UDC 004.635.3, DOI: 10.2298/CSIS1002331F

Subtree Matching by Pushdown Automata*

Tomáš Flouri[1], Jan Janoušek[2], and Bořivoj Melichar[2]

[1] Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
flourtom@fel.cvut.cz,
[2] Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University in Prague
Kolejní 550/2, 160 00 Prague 6, Czech Republic
{Jan.Janousek,Borivoj.Melichar}@fit.cvut.cz

Abstract. Subtree matching is an important problem in Computer Science on which a number of tasks, such as mechanical theorem proving, term-rewriting, symbolic computation and nonprocedural programming languages are based on. A systematic approach to the construction of subtree pattern matchers by deterministic pushdown automata, which read subject trees in prefix and postfix notation, is presented. The method is analogous to the construction of string pattern matchers: for a given pattern, a nondeterministic pushdown automaton is created and is then determinised. In addition, it is shown that the size of the resulting deterministic pushdown automata directly corresponds to the size of the existing string pattern matchers based on finite automata.

Keywords: subtree, subtree matching, pushdown automata.

1. Introduction

The theory of formal string (or word) languages [2, 16, 24] and the theory of formal tree languages [6, 8, 14] are important parts of the theory of formal languages [23]. While the models of computation of the theory of string languages are finite automata, pushdown automata, linear bounded automata and Turing machines, the most famous models of computation of the theory of tree languages are various kinds of tree automata [6, 8, 14]. Trees, however, can also be seen as strings, for example in their prefix (also called preorder) or postfix (also called postorder) notation. Recently it has been shown that the deterministic pushdown automaton (PDA) is an appropriate model of computation for labelled, ordered, ranked trees in postfix notation and that the trees in postfix notation, acceptable by deterministic PDA, form a proper superclass of the class of regular tree languages, which are accepted by finite tree automata [18].

* This research has been partially supported by the Ministry of Education, Youth and Sports under research program MSM 6840770014, and by the Czech Science Foundation as project No. 201/09/0807.

Trees represent one of the fundamental data structures used in Computer Science and thus tree pattern matching, the process of finding occurrences of subtrees in trees, is an important problem with many applications, such as compiler code selection, interpretation of non-procedural languages or various tree finding and tree replacement systems.

Tree pattern matching is often declared to be analogous to the problem of string pattern matching [6]. One of the basic approaches used for string pattern matching can be represented by finite automata constructed for the pattern, which means that the pattern is preprocessed. Examples of these automata are the string matching automata [9, 10, 22, 26]. Given a pattern P of size m, the string matching automaton can be constructed for the pattern P in time linear to m. The constructed string matching automaton accepts the set of words containing pattern P as a suffix, and thus it can find all occurrences of string P in a given text T. The main advantage of this kind of finite automata is that the deterministic string matching automaton can be constructed in time linear to the size of the given pattern P, and the search phase is in time linear to the input text. A generalization of the mentioned string matching problem can be the string matching problem with multiple patterns [1, 22, 26]. Given a set of patterns $P = \{p_1, p_2, \ldots, p_m\}$, the string matching automaton can be constructed in time linear to the number of symbols of patterns in set P. The constructed string matching automaton accepts the set of words having any of the patterns in P as a suffix, and thus it can find all occurrences of strings $p_1, \ldots, p_m$ in a given text T.

Although there are many tree pattern matching methods (see [5–7, 11, 15, 25] for these methods), they fail to present a simple and systematic approach with a linear time searching phase which would also be directly analogous to the basic string pattern matching method.

This paper, being an extended version of [12], presents a new kind of PDAs for trees in prefix and postfix notations called subtree matching PDAs, which are directly analogous to string matching automata and their properties. A subtree matching PDA, constructed from a given tree s, can find all occurrences of subtree s within a given tree t in time $\mathcal{O}(n)$, where n is the number of nodes of t. Subtree matching, as with string matching, can also be generalized to subtree matching with multiple patterns. Subtree matching PDAs can be constructed from a set of trees $P = \{t_1, t_2, \ldots, t_m\}$ in the same manner as string matching automata, retaining their property of linear searching phase $\mathcal{O}(n)$, where n is the number of nodes of the subject tree t.

Moreover, the presented subtree matching PDAs have the following two other properties. First, they are input–driven PDAs [28], which means that each pushdown operation is determined only by the input symbol. The input–driven PDAs can be always determinised [28]. Second, their pushdown symbol alphabets contain just one pushdown symbol and therefore their pushdown store can be implemented by a single integer counter. This means that the presented PDAs can be transformed to counter automata [4, 27], which is a weaker and simpler model of computation than the PDA.

The rest of the paper is organised as follows. Basic definitions are given in section 2. Some properties of subtrees in prefix notation are discussed in the third section. Sections 4 and 5 deal with the subtree matching PDA constructed over a single and multiple patterns, respectively. Section 6 shows the dual principle for the postfix notation and the last section is the conclusion.

2. Basic Notions

2.1. Ranked alphabet, tree, prefix notation, postfix notation, subtree matching

We define notions on trees similarly as they are defined in [2, 6, 8, 14].

We denote the set of natural numbers by $\mathbb{N}$. A *ranked alphabet* is a finite, nonempty set of symbols, each of which has a unique nonnegative *arity* (or *rank*). Given a ranked alphabet $\mathcal{A}$, the arity of a symbol $a \in \mathcal{A}$ is denoted by $Arity(a)$. The set of symbols of arity p is denoted by $\mathcal{A}_p$. Elements of arity $0, 1, 2, \ldots, p$ are respectively called nullary (constants), unary, binary, $\ldots$, p-ary symbols. We assume that $\mathcal{A}$ contains at least one constant. In the examples we use numbers at the end of identifiers for a short declaration of symbols with arity. For instance, $a2$ is a short declaration of a binary symbol a.

Based on concepts from graph theory (see [2]), a labelled, ordered, ranked tree over a ranked alphabet $\mathcal{A}$ can be defined as follows:

An *ordered directed graph* G is a pair (N, R), where N is a set of nodes and R is a set of linearly ordered lists of edges such that each element of R is of the form $((f, g_1), (f, g_2), \ldots, (f, g_n))$, where $f, g_1, g_2, \ldots, g_n \in N$, $n \geq 0$. This element would indicate that, for node f, there are n edges leaving f, the first entering node g_1, the second entering node g_2, and so forth.

A sequence of nodes $(f_0, f_1, \ldots, f_n)$, $n \geq 1$, is a *path* of length n from node f_0 to node f_n if there is an edge which leaves node f_{i-1} and enters node f_i for $1 \leq i \leq n$. A *cycle* is a path $(f_0, f_1, \ldots, f_n)$, where $f_0 = f_n$. An ordered *dag* (dag stands for Directed Acyclic Graph) is an ordered directed graph that has no cycle. *Labelling* of an ordered graph $G = (A, R)$ is a mapping of A into a set of labels. In the examples we use a_f for a short declaration of node f , labelled by symbol a.

Given a node f, its *out-degree* is the number of distinct pairs $(f, g) \in R$, where $g \in A$. By analogy, *in-degree* of node f is the number of distinct pairs $(g, f) \in R$, where $g \in A$.

A *labelled, ordered, ranked and rooted tree* t over a ranked alphabet $\mathcal{A}$ is an ordered dag $t = (N, R)$ with a special node $r \in A$ called the *root* such that
(1) r has in-degree 0,
(2) all other nodes of t have in-degree 1,
(3) there is just one path from the root r to every $f \in N$, where $f \neq r$,
(4) every node $f \in N$ is labelled by a symbol $a \in \mathcal{A}$ and out-degree of a_f is *Arity*(a).

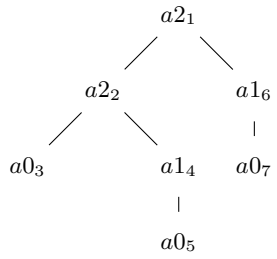
Nodes labelled by nullary symbols (constants) are called *leaves*.

Prefix notation pref(t) of a labelled, ordered, ranked and rooted tree t is obtained by applying the following *Step* recursively, beginning at the root of t:
Step: Let this application of *Step* be node a_f. If a_f is a leaf, list a and halt. If a_f is not a leaf, having direct descendants $a_{f_1}, a_{f_2}, \ldots, a_{f_n}$, then list a and subsequently apply *Step* to $a_{f_1}, a_{f_2}, \ldots, a_{f_n}$ in that order.
Postfix notation post(t) of t is formed by changing the last sentence of *Step* to read "Apply *Step* to $a_{f_1}, a_{f_2}, \ldots, a_{f_n}$ in that order and then list a."

Example 1. Consider a tree $t_1 = (\ \{a2_1, a2_2, a0_3, a1_4, a0_5, a1_6, a0_7\}, R\ )$ over $\mathcal{A} = \{a2, a1, a0\}$, where R is a set of the following ordered sequences of pairs:

$$((a2_1, a2_2), (a2_1, a1_6)),$$
$$((a2_2, a0_3), (a2_2, a1_4)),$$
$$((a1_4, a0_5)),$$
$$((a1_6, a0_7))$$

The prefix and postfix notations of tree t_1 are strings *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ and $post(t_1) = a0\ a0\ a1\ a2\ a0\ a1\ a2$, respectively. Trees can be represented graphically, and tree t_1 is illustrated in Fig. 1. □



pref$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$

Fig. 1. Tree t_1 from Example 1 and its prefix notation

The number of nodes of a tree t is denoted by $|t|$.

The height of a tree t, denoted by *Height(t)*, is defined as the maximal length of a path from the root of t to a leaf of t.

A subtree p *matches* an object tree t at node n if p is equal to the subtree of t rooted at n.

2.2. Alphabet, language, pushdown automaton

We define notions from the theory of string languages similarly as they are defined in [2, 16].

Let an *alphabet* be a finite nonempty set of symbols. A *string* x over a given alphabet is a finite, possibly empty sequence of symbols. A *language* over an alphabet $\mathcal{A}$ is a set of strings over $\mathcal{A}$. Set $\mathcal{A}^*$ denotes the set of all strings over $\mathcal{A}$ including the empty string, denoted by ε. Set $\mathcal{A}^+$ is defined as $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\varepsilon\}$. Similarly for string $x \in \mathcal{A}^*$, x^m, $m \geq 0$, denotes the m-fold concatenation of x with $x^0 = \varepsilon$. Set x^* is defined as $x^* = \{x^m : m \geq 0\}$ and $x^+ = x^* \setminus \{\varepsilon\} = \{x^m : m \geq 1\}$.

An (extended) *nondeterministic pushdown automaton* (nondeterministic PDA) is a seven-tuple $M = (Q, \mathcal{A}, G, \delta, q_0, Z_0, F)$, where Q is a finite set of *states*, $\mathcal{A}$ is the *input alphabet*, G is the *pushdown store alphabet*, δ is a mapping from $Q \times (\mathcal{A} \cup \{\varepsilon\}) \times G^*$ into a set of finite subsets of $Q \times G^*$, $q_0 \in Q$ is the initial state, $Z_0 \in G$ is the initial content of the pushdown store, and $F \subseteq Q$ is the set of final (accepting) states. The triplet $(q, w, x) \in Q \times \mathcal{A}^* \times G^*$ denotes the configuration of a pushdown automaton. In this paper we will write the top of the pushdown store x on its left hand side. The initial configuration of a pushdown automaton is a triplet (q_0, w, Z_0) for the input string $w \in \mathcal{A}^*$.

The relation $\vdash_M \subset (Q \times \mathcal{A}^* \times G^*) \times (Q \times \mathcal{A}^* \times G^*)$ is a *transition* of a pushdown automaton M. It holds that $(q, aw, \alpha\beta) \vdash_M (p, w, \gamma\beta)$ if $(p, \gamma) \in \delta(q, a, \alpha)$. The k-th power, transitive closure, and transitive and reflexive closure of the relation $\vdash_M$ is denoted $\vdash_M^k$, $\vdash_M^+$, $\vdash_M^*$, respectively. A pushdown automaton M is a *deterministic* pushdown automaton (deterministic PDA), if it holds:

1. $|\delta(q, a, \gamma)| \leq 1$ for all $q \in Q$, $a \in \mathcal{A} \cup \{\varepsilon\}$, $\gamma \in G^*$.
2. If $\delta(q, a, \alpha) \neq \emptyset$, $\delta(q, a, \beta) \neq \emptyset$ and $\alpha \neq \beta$ then α is not a suffix of β and β is not a suffix of α.
3. If $\delta(q, a, \alpha) \neq \emptyset$, $\delta(q, \varepsilon, \beta) \neq \emptyset$, then α is not a suffix of β and β is not a suffix of α.

A pushdown automaton is *input–driven* if its each pushdown operation is determined only by the input symbol.
A language L accepted by a pushdown automaton M is defined in two distinct ways:

1. *Accepting by final state:*

$$L(M) = \{x : \delta(q_0, x, Z_0) \vdash_M^* (q, \varepsilon, \gamma) \wedge x \in \mathcal{A}^* \wedge \gamma \in G^* \wedge q \in F\}.$$

2. *Accepting by empty pushdown store:*

$$L_\varepsilon(M) = \{x : (q_0, x, Z_0) \vdash_M^* (q, \varepsilon, \varepsilon) \wedge x \in \mathcal{A}^* \wedge q \in Q\}.$$

If a PDA accepts the language by empty pushdown store then the set F of final states may be the empty set. The subtree PDAs accept the languages by empty pushdown store.

In the rest of the text, we use the following notation for labelling edges when illustrating transition diagrams of various PDAs: For each transition rule $\delta_1(p, a, \alpha) = (q, \beta)$ from the transition mapping δ of a PDA, we label its edge leading from state p to state q by the triplet of the form $a|\alpha \mapsto \beta$.

For more details on pushdown automata see [2, 16].

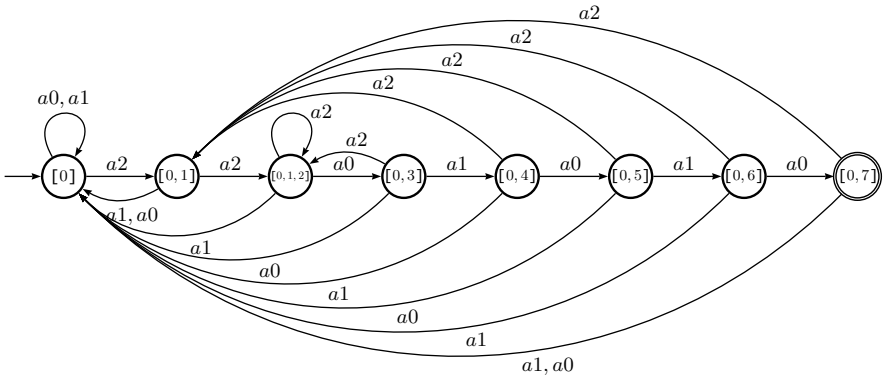


Fig. 2. Transition diagram of deterministic string matching automaton for pattern $x = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 2

2.3. Examples of string matching automaton

Example 2. The transition diagram of the deterministic string matching automaton constructed for string $a2\ a2\ a0\ a1\ a0\ a1\ a0$ is illustrated in Fig. 2. □

Example 3. The transition diagram of the deterministic string matching automaton constructed for a set of strings $P = \{a2\ a2\ a0\ a0\ b0, a2\ b1\ a0\ a0, a2\ a0\ a0\}$ is illustrated in Fig. 3. □

See [2, 9, 22] for definitions of finite automata and construction of the deterministic string matching automaton.

3. Properties of subtrees in prefix notation

In this section we describe some general properties of the prefix notation of a tree and of its subtrees. These properties are important for the construction of the subtree matching PDA, which is described in the next two sections.

Example 4. Consider tree t_1 in prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 1, which is illustrated in Fig. 1. Tree t_1 contains only subtrees shown in Fig. 4.

Generally, for any tree, the following theorem holds.

Theorem 1. *Given a tree t and its prefix notation pref(t), all subtrees of t in prefix notation are substrings of pref(t).*

Proof. By induction on the height of the subtree.

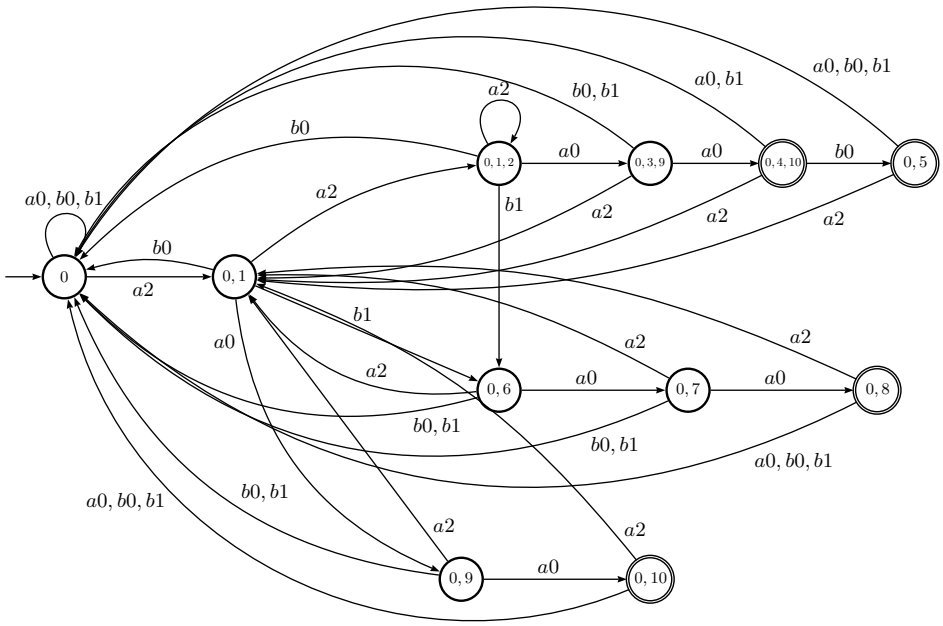
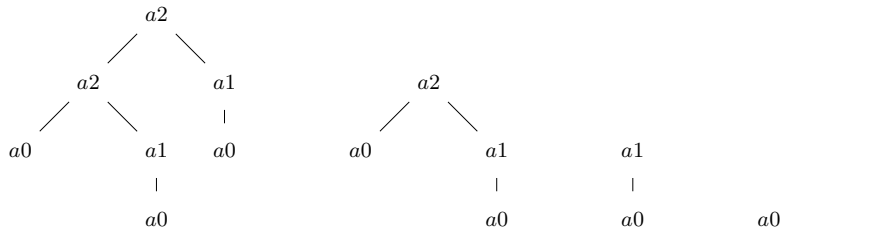


Fig. 3. Transition diagram of deterministic string matching automaton (Aho-Corasick) for patterns $\{a2\ a2\ a0\ a0\ b0,\ a2\ b1\ a0\ a0,\ a2\ a0\ a0\}$

1. If a subtree t' has just one node a, where *Arity*$(a) = 0$, then *Height*$(t') = 0$, *pref*$(t') = a$ and the claim holds for that subtree.
2. Assume that the claim holds for subtrees $t_1, t_2, \ldots, t_p$, where $p \geq 1$ and *Height*$(t_1) \leq m$, *Height*$(t_2) \leq m$, ..., *Height*$(t_p) \leq m$, $m \geq 0$. We have to prove that the claim holds also for each subtree $t' = at_1t_2 \ldots t_p$, where *Arity*$(a) = p$ and *Height*$(t') = m + 1$:
   As *pref*$(t') = a$ *pref*(t_1) *pref*$(t_2) \ldots$ *pref*(t_p), the claim holds for the subtree t'.

Thus, the theorem holds. □

However, not every substring of a tree in prefix notation is its subtree in prefix notation. This can be easily seen on the fact that for a given tree with n nodes in prefix notation, there can be $\mathcal{O}(n^2)$ distinct substrings but there is just n subtrees – each node of the tree is the root of just one subtree. Just those substrings which themselves are trees in prefix notation are those which are the subtrees in prefix notation. This property is formalised by the following definition and theorem.



pref$(t) = a2\ a2\ a0\ a1\ a0\ a1\ a0$	*pref*$(t) = a2\ a0\ a1\ a0$	*pref*$(t) = a1\ a0$	*pref*$(t) = a0$
post$(t) = a0\ a0\ a1\ a2\ a0\ a1\ a2$	*post*$(t) = a0\ a0\ a1\ a2$	*post*$(t) = a0\ a1$	*post*$(t) = a0$

Fig. 4. All subtrees of tree t_1 from Example 1, and their prefix and postfix notations

Definition 1. *Let $w = a_1 a_2 \ldots a_m$, $m \geq 1$, be a string over a ranked alphabet $\mathcal{A}$. Then, the* arity checksum *ac*$(w) =$ *Arity*$(a_1)+$*Arity*$(a_2)+\ldots+$*Arity*$(a_m)-m+1=\sum_{i=1}^{m}$ *Arity*$(a_i) - m + 1$.

Theorem 2. *Let pref*(t) *and w be a tree t in prefix notation and a substring of pref*(t)*, respectively. Then, w is the prefix notation of a subtree of t, if and only if $ac(w) = 0$, and $ac(w_1) \geq 1$ for each w_1, where $w = w_1 x$, $x \neq \varepsilon$.*

Proof. It is easy to see that for any two subtrees st_1 and st_2 it holds that *pref*(st_1) and *pref*(st_2) are either two different strings or one is a substring of the other. The former case occurs if the subtrees st_1 and st_2 are two different trees with no shared part and the latter case occurs if one tree is a subtree of the other tree. No partial overlapping of subtrees is possible in ranked ordered trees. Moreover, for any two neighbouring subtrees it holds that their prefix notations are two adjacent substrings.

- *If:* By induction on the height of a subtree st, where $w =$ *pref*(st):
  1. We assume that *Heigth*$(st) = 1$, which means we consider the case $w = a$, where *Arity*$(a) = 0$. Then, *ac*$(w) = 0$. Thus, the claim holds for the case *Height*$(st) = 1$.
  2. Assume that the claim holds for the subtrees $st_1, st_2, \ldots, st_p$ where $p \geq 1$, *Height*$(st_1) \leq m$, *Height*$(st_2) \leq m$, ..., *Height*$(st_p) \leq m$ and *ac*(*pref*$(st_1)) = 0$, *ac*(*pref*$(st_2)) = 0$, ..., *ac*(*pref*$(st_p)) = 0$.
     We are to prove that it holds also for a subtree of height $m+1$. Assume $w = a$ *pref*(st_1) *pref*(st_2) ... *pref*(st_p), where *Arity*$(a) = p$. Then *ac*$(w) = p+$*ac*(*pref*$(st_1))+$*ac*(*pref*$(st_2))+\ldots+$*ac*(*pref*$(st_p))-(p+1)+1 = 0$ and *ac*$(w_1) \geq 1$ for each w_1 , where $w = w_1 x$, $x \neq \varepsilon$.
     Thus, the claim holds for the case *Height*$(st) = m + 1$.
- *Only if*: Assume *ac*$(w) = 0$, and $w = a_1 a_2 \ldots a_k$, where $k \geq 1$, *Arity*$(a_1) = p$. Since *ac*$(w_1) \geq 1$ for each w_1, where $w = w_1 x$, $x \neq \varepsilon$, none of the substrings w_1 can be a subtree in prefix notation. This means that the only possibility

for $ac(w) = 0$ is that w is of the form $w = a$ *pref*(t_1) *pref*$(t_2) \ldots$ *pref*(t_p), where $p \geq 0$, and $t_1, t_2 \ldots t_p$ are neighbouring subtrees. In such case, *ac*$(w) = p + 0 - (p + 1) + 1 = 0$.
No other possibility of the form of w for *ac*$(w) = 0$ is possible. Thus, the claim holds.

Thus, the theorem holds. □

We note that in subtree matching PDAs, the arity checksum is computed by pushdown operations, where the contents of the pushdown store represents the corresponding arity checksum. For example, the empty pushdown store means that the corresponding arity checksum is equal to 0.

4. Subtree Matching pushdown automaton

This section deals with the subtree matching PDA for trees in prefix notation: algorithms and theorems are given and the subtree matching PDA and its construction are demonstrated with an example.

Problem 1 (Subtree Matching). Given two trees s and t, find all occurrences of tree s in tree t.

Definition 2. *Let s and pref(s) be a tree and its prefix notation, respectively. Given an input tree t, a subtree pushdown automaton constructed over pref(s) accepts all matches of tree s in the input tree t by final state.*

First, we start with a PDA which accepts the whole subject tree in prefix notation. The construction of the PDA accepting a tree in prefix notation is described by Alg. 1. The constructed PDA is deterministic.

Algorithm 1. Construction of a PDA accepting a tree t in prefix notation *pref*(t).
Input: A tree t over a ranked alphabet $\mathcal{A}$; prefix notation *pref*$(t) = a_1 a_2 \ldots a_n$, $n \geq 1$.
Output: PDA $M_p(t) = (\{0, 1, 2, \ldots, n\}, \mathcal{A}, \{S\}, \delta, 0, S, \{n\})$.
Method:

1. For each state i, where $1 \leq i \leq n$, create a new transition $\delta(i-1, a_i, S) = (i, S^{Arity(a_i)})$, where $S^0 = \varepsilon$. □

Example 5. The PDA constructed by Alg. 1, accepting the prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ of tree t_1 from Example 1, is the deterministic PDA $M_p(t_1) = (\{0, 1, 2, 3, 4, 5, 6, 7\}, \mathcal{A}, \{S\}, \delta_1, 0, S, \{n\}))$, where the mapping δ_1 is a set of the following transitions:

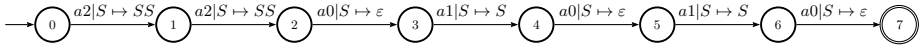


Fig. 5. Transition diagram of deterministic PDA $M_p(t_1)$ accepting tree t_1 in prefix notation *pref*$(t_1) = a2\ a0\ a2\ a0\ a0\ a0$ from Example 5

$$\begin{aligned}
\delta_1(0, a2, S) &= (1, SS)\\
\delta_1(1, a2, S) &= (2, SS)\\
\delta_1(2, a0, S) &= (3, \varepsilon)\\
\delta_1(3, a1, S) &= (4, S)\\
\delta_1(4, a0, S) &= (5, \varepsilon)\\
\delta_1(5, a1, S) &= (6, S)\\
\delta_1(6, a0, S) &= (7, \varepsilon)
\end{aligned}$$

The transition diagram of deterministic PDA $M_p(t_1)$ is illustrated in Fig. 5. Fig. 6 shows the sequence of transitions (trace) performed by deterministic PDA $M_p(t_1)$ for tree t_1 in prefix notation. □

State	Input	Pushdown Store
0	$a2\ a2\ a0\ a1\ a0\ a1\ a0$	S
1	$a2\ a0\ a1\ a0\ a1\ a0$	$S\ S$
2	$a0\ a1\ a0\ a1\ a0$	$S\ S\ S$
3	$a1\ a0\ a1\ a0$	$S\ S$
4	$a0\ a1\ a0$	$S\ S$
5	$a1\ a0$	S
6	$a0$	S
7	ε	ε
accept		

Fig. 6. Trace of deterministic PDA $M_p(t_1)$ from Example 5 for tree t_1 in prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$

Theorem 3. *Let $M = (\{Q, \mathcal{A}, \{S\}, \delta, 0, S, F)$ be an input–driven PDA whose each transition from δ is of the form $\delta(q_1, a, S) = (q_2, S^i)$, where $i =$ Arity(a). Then, if $(q_3, w, S) \vdash_M^+ (q_4, \varepsilon, S^j)$, then $j =$ ac(w).*

Proof. By induction on the length of w:

1. Assume $w = a$. Then, $(q_3, a, S) \vdash_M (q_4, \varepsilon, S^j)$, where $j =$ *Arity*$(a) = ac(a)$. Thus, the claim holds for the case $w = a$.
2. Assume that the claim holds for a string $w = a_1 a_2 \ldots a_k$, where $k \geq 1$. This means that $(q_3, a_1 a_2 \ldots a_k, S) \vdash_M^k (q_4, \varepsilon, S^j)$, where $j =$ ac$(a_1 a_2 \ldots a_k)$. We have to prove that the claim holds also for $w = a_1 a_2 \ldots a_k\ a$.

It holds that $(q_3, a_1a_2 \dots a_k a, S) \vdash_M^k (q_4, a, S^j) \vdash_M (q_5, \varepsilon, S^l)$, where $l = j + \mathit{Arity}(a) - 1 = ac(w) + \mathit{Arity}(a) - 1 = \mathit{Arity}(a_1) + \mathit{Arity}(a_2) + \ldots + \mathit{Arity}(a_k) - k + 1 + \mathit{Arity}(a) - 1 = \mathit{ac}(a_1a_2 \dots a_k a)$.
Thus, the claim holds for the case $w = a_1a_2 \dots a_k\ a$.

Thus, the theorem holds. □

The correctness of the deterministic PDA constructed by Alg. 1, which accepts trees in prefix notation, is described by the following lemma.

Lemma 1. *Given a tree t and its prefix notation $\mathit{pref}(t)$, the PDA $M_p(t) = (\{0, 1, 2, \dots, n\}, \mathcal{A}, \{S\}, \delta, 0, S, F)$, where $n = |t|$, constructed by Alg. 1, accepts $\mathit{pref}(t)$.*

Proof. By induction on the height of the tree t:

1. If tree t has just one node a, where $\mathit{Arity}(a) = 0$, then $\mathit{Height}(t) = 0$, $\mathit{pref}(t) = a$, $\delta(0, a, S) = (1, \varepsilon) \in \delta$, $(0, a, S) \vdash_{M_p(t)} (1, \varepsilon, \varepsilon)$ and the claim holds for that tree.
2. Assume that claim holds for trees $t_1, t_2, \dots, t_p$, where $p \geq 1$, $\mathit{Height}(t_1) \leq m$, $\mathit{Height}(t_2) \leq m$, ..., $\mathit{Height}(t_p) \leq m$, $m \geq 0$.
   We have to prove that the claim holds also for each tree t such that $\mathit{pref}(t) = a\ \mathit{pref}(t_1)\mathit{pref}(t_2) \dots \mathit{pref}(t_p)$, $\mathit{Arity}(a) = p$, and $\mathit{Height}(t) \geq m + 1$:
   Since $\delta(0, a, S) = (1, S^p) \in \delta$, and $(0, a\ \mathit{pref}(t_1)\mathit{pref}(t_2) \dots \mathit{pref}(t_p), S)$
   $\vdash_{M_p(t)} (1, \mathit{pref}(t_1)\mathit{pref}(t_2) \dots \mathit{pref}(t_p), S^p)$
   $\vdash^*_{M_p(t)} (i, \mathit{pref}(t_2) \dots \mathit{pref}(t_p), S^{p-1})$
   $\vdash^*_{M_p(t)} \dots$
   $\vdash^*_{M_p(t)} (j, \mathit{pref}(t_p), S)$
   $\vdash^*_{M_p(t)} (k, \varepsilon, \varepsilon)$,
   the claim holds for that tree.

Thus, the lemma holds. □

We present the construction of the deterministic subtree matching PDA for trees in prefix notation. The construction consists of two steps. First, a nondeterministic subtree matching PDA is constructed by Alg. 2. This nondeterministic subtree matching PDA is an extension of the PDA accepting trees in prefix notation, which is constructed by Alg. 1. Second, the constructed nondeterministic subtree matching PDA is transformed to the equivalent deterministic subtree matching PDA. In spite of the fact that the determinisation of a nondeterministic PDA is not possible generally, the constructed nondeterministic subtree matching PDA is an input–driven PDA and therefore can be determinised [28].

Algorithm 2. Construction of a nondeterministic subtree matching PDA for a tree t in prefix notation $\mathit{pref}(t)$.
Input: A tree t over a ranked alphabet $\mathcal{A}$; prefix notation $\mathit{pref}(t) = a_1a_2 \dots a_n$, $n \geq 1$.

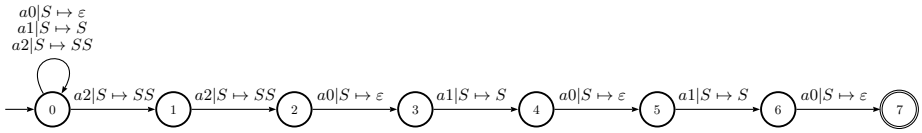


Fig. 7. Transition diagram of nondeterministic subtree matching PDA $M_p(t_1)$ for tree t_1 in prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 6

Output: Nondeterministic subtree matching PDA $M_{nps}(t) = (\{0, 1, 2, \ldots, n\}, \mathcal{A}, \{S\}, \delta, 0, S, \{n\})$.
Method:

1. Create PDA $M_{nps}(t)$ as PDA $M_p(t)$ by Alg. 1.
2. For each symbol $a \in \mathcal{A}$ create a new transition $\delta(0, a, S) = (0, S^{\mathit{Arity}(a)})$, where $S^0 = \varepsilon$.

Example 6. The subtree matching PDA, constructed by Alg. 2 from tree t_1 having prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$, is the nondeterministic PDA $M_{nps}(t_1) = (\{0, 1, 2, 3, 4, 5, 6, 7\}, \mathcal{A}, \{S\}, \delta_2, 0, S, \{7\}))$, where mapping δ_2 is a set of the following transitions:

$$\begin{array}{ll}
\delta_2(0, a2, S) = (1, SS) & \\
\delta_2(1, a2, S) = (2, SS) & \delta_2(0, a2, S) = (0, SS) \\
\delta_2(2, a0, S) = (3, \varepsilon) & \delta_2(0, a1, S) = (0, S) \\
\delta_2(3, a1, S) = (4, S) & \delta_2(0, a0, S) = (0, \varepsilon) \\
\delta_2(4, a0, S) = (5, \varepsilon) & \\
\delta_2(5, a1, S) = (6, S) & \\
\delta_2(6, a0, S) = (7, \varepsilon) &
\end{array}$$

The transition diagram of the nondeterministic PDA $M_{nps}(t_1)$ is illustrated in Fig. 7. □

Theorem 4. *Given a tree t and its prefix notation pref(t), the PDA $M_{nps}(t)$ constructed by Alg. 2 is a subtree matching PDA for pref(t).*

Proof. According to Theorem 2, given an input tree t, each subtree in prefix notation is a substring of *pref*(t). Since the PDA $M_{nps}(s)$ has just states and transitions equivalent to the states and transitions, respectively, of the string matching automaton, the PDA $M_{nps}(t)$ accepts all matches of subtree s in tree t by final state. □

For the construction of deterministic subtree PDA, we use the transformation described by Alg. 3. This transformation is based on the well known transformation of nondeterministic finite automaton to an equivalent deterministic one, which constructs the states of the deterministic automaton as subsets of states

of the nondeterministic automaton and selects only a set of accessible states (i.e. subsets) [16]. Again, states of the resulting deterministic PDA correspond to subsets of states of the original nondeterministic PDA.

Algorithm 3. Transformation of an input–driven nondeterministic PDA to an equivalent deterministic PDA.
Input: Input–driven nondeterministic PDA $M_{nx}(t) = (\{0, 1, 2, \ldots, n\}, \mathcal{A}, \{S\}, \delta, 0, S, F)$
Output: Equivalent deterministic PDA $M_{dx}(t) = (Q', \mathcal{A}, \{S\}, \delta', q_I, S, F')$.
Method:

1. Initially, $Q' = \{\{0\}\}$, $q_I = \{0\}$ and $\{0\}$ is an unmarked state.
2. (a) Select an unmarked state q' from Q'.
   (b) For each input symbol $a \in \mathcal{A}$:
      i. $q'' = \{q : \ \delta(p, a, \alpha) = (q, \beta)$ for all $p \in q'\}$.
      ii. Add transition $\delta'(q', a, S) = (q'', S^{Arity}(a))$.
      iii. If $q'' \notin Q$ then add q'' to Q and set it as unmarked state.
   (c) Set state q' as marked.
3. Repeat step 2 until all states in Q' are marked.
4. $F' = \{\ q' \mid q' \in Q' \wedge q' \cap F \neq \emptyset\ \}$. □

The deterministic subtree matching automaton $M_{dps}(t)$ for a tree t with prefix notation *pref*(t) is demonstrated by the following example.

Example 7. The deterministic subtree matching PDA for tree t_1 in prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 1 , which has been constructed by Alg. 3 from nondeterministic subtree matching PDA $M_{nps}(t_1)$ from Example 6, is the deterministic PDA $M_{dps}(t_1) = (\{[0], [0, 1], [0, 1, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7]\}, \mathcal{A}, \{S\}, \delta_3, [0], S, \{[0, 7]\})$, where its transition diagram is illustrated in Fig. 9.

We note that the deterministic subtree matching PDA $M_{dps}(t_1)$ has a very similar transition diagram to the deterministic string matching automaton constructed for *pref*(t_1) [9, 22], as can be seen by comparing Figs. 2 and 9. The only difference between the two types of automata are the pushdown operations appearing in the subtree matching PDA, which ensure the validity of the input tree. The input tree is valid only if the pushdown store of the subtree PDA is emptied after the last symbol from the prefix notation of the input tree is read.

Fig. 8 shows the sequence of transitions (trace) performed by the deterministic subtree PDA $M_{dps}(t_1)$ for an input tree t_2 in prefix notation *pref*$(t_2) = a2\ a2\ a2\ a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$. The accepting state is $\{0, 7\}$. Fig. 10 depicts the pattern subtree t_1 and input tree t_2. □

Theorem 5. *Given a nondeterministic input–driven PDA $M_{nx}(t) = (Q, \mathcal{A}, \{S\}, \delta, q_0, S, F)$, the deterministic PDA $M_{dx}(t) = (Q', \mathcal{A}, \{S\}, \delta', \{q_0\}, S, F')$ which is constructed by Alg. 3 is equivalent to PDA $M_{nx}(t)$.*

State	Input	PDS
$\{0\}$	$a2\ a2\ a2\ a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$	S
$\{0,1\}$	$a2\ a2\ a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$	SS
$\{0,1,2\}$	$a2\ a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$	SSS
$\{0,1,2\}$	$a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$	$SSSS$
$\{0,3\}$	$a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$	SSS
$\{0,4\}$	$a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$	SSS
$\{0,5\}$	$a1\ a0\ a1\ a1\ a2\ a0\ a0$	SS
$\{0,6\}$	$a0\ a1\ a1\ a2\ a0\ a0$	SS
$\{0,7\}$	$a1\ a1\ a2\ a0\ a0$ match	S
$\{0\}$	$a1\ a2\ a0\ a0$	S
$\{0\}$	$a2\ a0\ a0$	S
$\{0,1\}$	$a0\ a0$	SS
$\{0\}$	$a0$	S
$\{0\}$	ε	ε

Fig. 8. Trace of deterministic subtree PDA $M_{dps}(t_1)$ from Example 7 for an input subtree t_2 in prefix notation *pref*$(t_2) = a2\ a2\ a2\ a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$

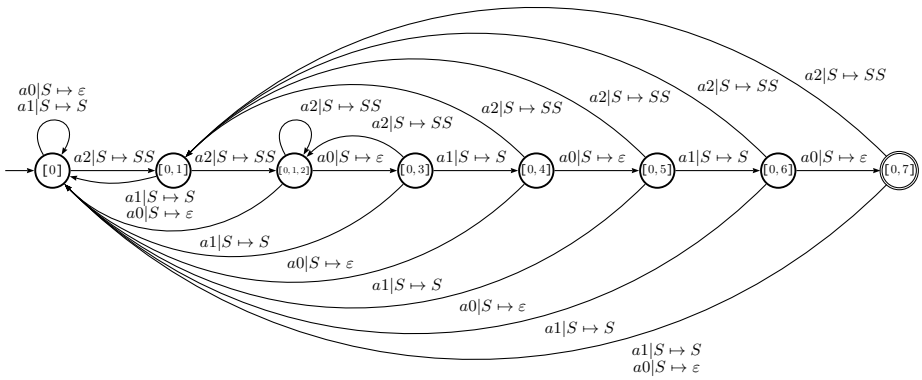
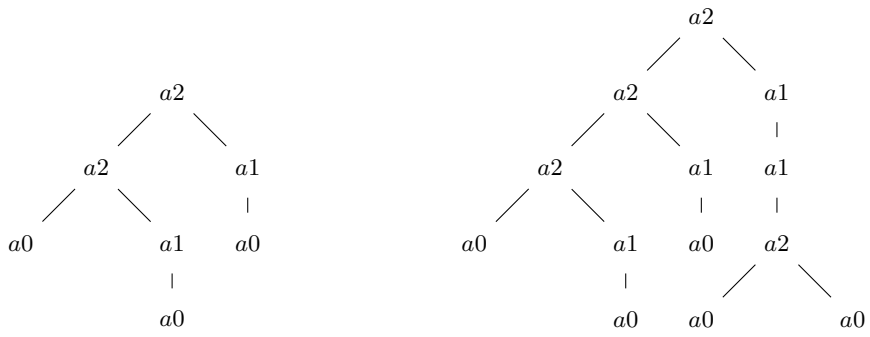


Fig. 9. Transition diagram of deterministic PDA $M_{dps}(t_1)$ for tree t_1 in prefix notation *pref*$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 7

Proof. First, we prove the following claim by induction on i:
(*): $(q'_1, w, S) \vdash^i_{M_{dx}(t)} (q'_2, \varepsilon, S^j)$ if and only if
$q'_2 = \{p : (q, w, S) \vdash^i_{M_{nx}(t)} (p, \varepsilon, S^j) \text{ for some } q \in q'_1\}$.

1. Assume i=1.
   - *if*: if $(q'_1, a, S) \vdash_{M_{dx}(t)} (q'_2, \varepsilon, S^j)$, then there exists a state $q \in q'_1$, where $(q, a, S) \vdash_{M_{nx}(t)} (p, \varepsilon, S^j)$, $p \in q'_2$.
   - *only if*: if $(q, a, S) \vdash_{M_{nx}(t)} (p, \varepsilon, \beta)$, then for each $q'_1 \in Q'$, where $q \in q'_1$, it holds that $(q'_1, a, S) \vdash_{M_{dx}(t)} (q'_2, \varepsilon, S^j)$, where $p \in q'_2$.
2. Assume that claim (*) holds for $i = 1, 2, \ldots, k$, $k \geq 1$.
   This means that $(q'_1, w, S) \vdash^k_{M_{dx}(t)} (q'_2, \varepsilon, S^j)$ if and only if
   $q'_2 = \{p : (q, S, w) \vdash^k_{M_{nx}(t)} (p, \varepsilon, S^j) \text{ for some } q \in q'_1\}$. We have to prove that claim (*) holds also for $i = k + 1$.
   - *if*: if $(q'_1, w, S) \vdash^k_{M_{dx}(t)} (q'_2, a, S^l) \vdash_{M_{dx}(t)} (q'_3, \varepsilon, S^j)$, then there exists a state $q \in q'_2$, where $(q, a, S^l) \vdash_{M_{nx}(t)} (p, \varepsilon, S^j)$, $p \in q'_3$.
   - *only if*: if $(q_0, pref(t), S) \vdash^k_{M_{nx}(t)} (q, a, S^l) \vdash_{M_{nx}(t)} (p, \varepsilon, S^j)$, then for each $q'_1 \in Q'$, where $q \in q'_1$, it holds that $(q'_1, a, S^l) \vdash_{M_{dx}(t)} (q'_2, \varepsilon, S^j)$, where $p \in q'_2$.

As a special case of claim (*), $(\{q_0\}, pref(t), S) \vdash^i_{M_{dx}(t)} (q', \varepsilon, \varepsilon)$ if and only if $(q_0, S, pref(t)) \vdash^i_{M_{nx}(t)} (q_1, \varepsilon, \varepsilon)$. Thus, the theorem holds.



pref$(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ *pref*$(t_2) = a2\ a2\ a2\ a0\ a1\ a0\ a1\ a0\ a1\ a1\ a2\ a0\ a0$
post$(t_1) = a0\ a0\ a1\ a2\ a0\ a1\ a2$ *post*$(t_2) = a0\ a0\ a1\ a2\ a0\ a1\ a2\ a0\ a0\ a2\ a1\ a1\ a2$

Fig. 10. Trees t_1 and t_2 from Example 7 along with their prefix and postfix notations

Theorem 6. *Given a tree t with n nodes in its prefix or postfix notation, the deterministic subtree matching PDA $M_{pds}(t)$ constructed by Alg. 2 and 3 is made of exactly $n + 1$ states, one pushdown symbol and $|\mathcal{A}|(n + 1)$ transitions.*

Proof. Let $M_{nps}(t) = (\{0, 1, 2, \ldots, n\}, \mathcal{A}, \{S\}, \delta, 0, S, \{n\}$ be an automaton constructed from tree t with a prefix notation ***pref***$(t) = a_1\ a_2\ \ldots\ a_n$ over ranked alphabet $\mathcal{A}$ by Alg. 2. We will prove that this automaton is directly analogous to the string matching automaton and accepts the same language if we ignore the pushdown operations, which actually do not affect the process of determinisation as M_{pds} is an input–driven automaton. From Alg 2 and 3, $M_{nps}(t)$ has transitions $\delta(0, a, S) = (0, S^{\textit{Arity}(a)})$ for all $a \in \mathcal{A}$ and $\delta(i-1, a_i, S) = (i, \varepsilon, S^{\textit{Arity}(a_i)})$. The proof is a mutual induction of the following $n+1$ statements:
(1) $\delta^*(0, w, S) = (0, \varepsilon, S^{\textit{ac}(w)})$, $w \in \mathcal{A}^*$.
(2) $\delta^*(0, w, S) = (1, \varepsilon, S^{\textit{ac}(w)})$ if and only if $w = w_1 a_1$, $w_1 \in \mathcal{A}^*$
(i) $\delta^*(0, w, S) = (i-1, \varepsilon, S^{\textit{ac}(w)})$ if and only if $w = w_1 a_1 a_2 \ldots a_{i-1}$, $w_1 \in \mathcal{A}^*$

1. Assume that $|w| = 0$, which means $w = \varepsilon$. Statement (1) holds, since $\delta^*(0, \varepsilon, S) = (0, \varepsilon, S)$. Statements ($i$), $1 < i \leq n+1$, do not hold as $\delta^*(0, \varepsilon, S)$ contains, from its basic definition, only $(0, \varepsilon, S)$.
2. Assume $w = w_1 a$, where $w_1 \in \mathcal{A}^k$, that is $|w_1| = k$ and $a \in \mathcal{A}$. We may assume that statements (i) $1 < i \leq n+1$ hold for w_1, and we need to prove them for w. We assume the inductive hypothesis for k and prove it for $k+1$.
   (a) There exists a series of transitions $(0, w_1, S) \vdash^* (0, \varepsilon, S^{\textit{ac}(w_1)})$, since $\delta(0, a, S) = (0, \varepsilon, S^{\textit{Arity}(a)})$ are transitions of automaton M_{nps}.Thus statement (1) is proved for w.
   (b) We now prove statements i, where $1 < i \leq n+1$:
      – *If:* Assume that $w_1 = w_2 a_1 a_2 \ldots a_{i-2}$, where $w_2 \in \mathcal{A}^*$ and $a = a_{i-1}$. By statement $(i-1)$ applied to w_1, we know from our induction hypothesis that there exists a series of transitions $(0, w_1, S) \vdash^* (i-2, \varepsilon, S^{\textit{ac}(w_1)})$. Since for all $1 \leq j \leq n$ there exists a transition $\delta(j-1, a_j, S) = (j, S^{\textit{Arity}(a_j)})$, we conclude that $\delta^*(0, w, S) = (i-1, \varepsilon, S^{\textit{ac}(w)})$.
      – *Only if:* Suppose there exists a series of transitions $(0, w, S) \vdash^* (i-1, \varepsilon, S^{\textit{ac}(w)})$. From the inductive assumption we know that there exists a series of transitions $(0, w_1, S) \vdash^* (i-2, \varepsilon, S^{\textit{ac}(w_1)})$. By statement $(i-1)$ applied to w_1, we know that $w_1 = w_2 a_1 a_2 \ldots a_{i-2}$. Thus $w = w_2 a_1 a_2 \ldots a_{i-1}$, and we have proved statement (i).

Thus, from statements $1, \ldots, n+1$, if we ignore the pushdown operations, M_{pds} accepts the language $L = \{w.\textit{pref}(t)\}$, where $w \in \mathcal{A}^*$. Since the subtree matching PDA is directly analogous to the string matching automaton, we can use the proof from [10, 22] for space and time complexities. □

Theorem 7. *Given an input tree t with n nodes, the searching phase of the deterministic subtree matching automaton constructed by Algs. 2 and 3 is $\mathcal{O}(n)$.*

Proof. The searching phase consists of reading tree t once, symbol by symbol from left to right. The appropriate transition is taken each time a symbol is read, resulting in exactly n transitions. Each transition consumes a constant time because the time of each pushdown operation is limited by the maximal arity of nodes. Occurrences of the subtree to find are matched by transitions leading to the final states. □

Finally, we note that trees having structure *pref*$(t) = (a1)^{n-1}a0$ represent strings. The deterministic subtree matching PDA for such trees has the same number of states and transitions as the deterministic string matching automaton constructed for *pref*(t) and accepts the same language.

5. Multiple subtree matching

In this section we present a generalization of Problem 1. We deal with the construction of subtree matching PDA over a finite set of trees. The whole concept is demonstrated with an example.

Problem 2 (Multiple Subtree Matching). Given a tree t and a set of m trees $P = \{t_1, t_2, \ldots, t_m\}$, find all occurrences of trees $t_1, t_2, \ldots, t_m$ in tree t.

Definition 3. *Let $P = \{t_1, t_2, \ldots, t_m\}$ be a set of m trees and pref$(t_i), 1 \leq i \leq m$ be the prefix notation of the i-th tree in P. Given an input tree t, a subtree pushdown automaton constructed over set P accepts all matches of subtrees $t_1, t_2, \ldots, t_m$ in the input tree t by final state.*

Similarly as in Section 4, our method begins with a PDA which accepts trees $t_1, t_2, \ldots, t_m$ in their prefix notation. The construction of this PDA is described by Alg. 4

Algorithm 4. Construction of a PDA accepting a set of trees $P = \{t_1, t_2, \ldots, t_m\}$ in their prefix notation.
Input: A set of trees $P = \{t_1, t_2, \ldots, t_m\}$ over a ranked alphabet $\mathcal{A}$; prefix notation *pref*$(t_i) = a_1 a_2 \ldots a_{n_i}$, $1 \leq i \leq m$, $n_i \geq 1$.
Output: PDA $M_p(P) = (\{0, 1, 2, \ldots, q\}, \mathcal{A}, \{S\}, \delta, 0, S, F)$.
Method:

1. Let $q \leftarrow 0$ and $F \leftarrow \emptyset$
2. For each tree $t_i = a_1^i \; a_2^i \; \ldots \; a_{|t_i|}^i$, $1 \leq i \leq m$, do
   (a) Let $l \leftarrow 0$
   (b) For $j = 1$ to $|t_i|$ do
      i. If the transition $\delta(l, a_j^i, S)$ is not defined then
         A. Let $q \leftarrow q + 1$
         B. Create a transition $\delta(l, a_j^i, S) \leftarrow (q, S^{\mathit{Arity}(a_j^i)})$
         C. Let $l \leftarrow q$
      ii. Else if transition $\delta(l, a_j^i, S)$ is defined
         A. $l \leftarrow p$ where $(p, \gamma) \leftarrow \delta(l, a_j, S)$
   (c) $F \leftarrow F \cup \{l\}$

Example 8. Consider a set of trees $P = \{t_1, t_2, t_3\}$, with their prefix notations being *pref*$(t_1) = a2 \; a2 \; a0 \; a0 \; b0$, *pref*$(t_2) = a2 \; b1 \; a0 \; a0$ and *pref*$(t_3) = a2 \; a0 \; a0$. The deterministic PDA constructed by Alg. 4 accepting the prefix notation of trees in P is $M_p(P) = (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, \mathcal{A}, \{S\}, \delta_1, 0, S, \{5, 8, 10\}))$, where mapping δ_1 is a set of the following transitions:

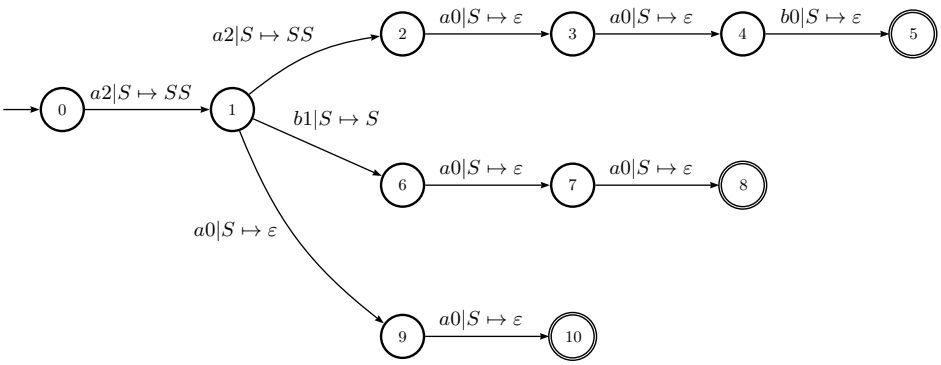


Fig. 11. Transition diagram of deterministic PDA $M_p(P)$ accepting the trees with prefix notation $\{a2\ a2\ a0\ a0\ b0, a2\ b1\ a0\ a0, a2\ a0\ a0\}$ from Example 8

$$
\begin{aligned}
&\delta_1(0, a2, S) = (1, SS)\\
&\delta_1(1, a2, S) = (2, SS)\\
&\delta_1(2, a0, S) = (3, \varepsilon)\\
&\delta_1(3, a0, S) = (4, \varepsilon)\\
&\delta_1(4, b0, S) = (5, \varepsilon)\\
&\delta_1(1, b1, S) = (6, S)\\
&\delta_1(6, a0, S) = (7, \varepsilon)\\
&\delta_1(7, a0, S) = (8, \varepsilon)\\
&\delta_1(1, a0, S) = (9, \varepsilon)\\
&\delta_1(9, a0, S) = (10, \varepsilon)
\end{aligned}
$$

The transition diagram of deterministic PDA $M_p(P)$ is illustrated in Fig. 11.

Fig. 12 shows the sequence of transitions (trace) performed by deterministic PDA $M_p(P)$ for trees $t_1, t_2, t_3 \in P$ in prefix notation. □

The correctness of the deterministic PDA constructed by Alg. 4, which accepts trees in prefix notation, is described by the following lemma.

Lemma 2. *Given a set of k trees $P = \{t_1, t_2, \ldots, t_m\}$ and their prefix notation pref(t_i), $1 \leq i \leq m$, the PDA $M_p(P) = (\{0, 1, 2, \ldots, n\}, \mathcal{A}, \{S\}, \delta, 0, S, F)$, where $1 + min(|t_1|, |t_2|, \ldots, |t_m|) \leq n \leq 1 + \sum_{j=1}^{k} |t_j|$, constructed by Alg. 4 accepts pref(t_i), where $1 \leq t_i \leq m$.*

Proof. By induction on the height of trees $t_1, t_2, \ldots, t_m$:

1. If trees $t_1, t_2, \ldots, t_m$ have just one node, $a_1, a_2, \ldots, a_k$ respectively, where *Arity*$(a_i) = 0$, for all $1 \leq i \leq k$, then *Height*$(t_i) = 0$, *pref*$(t_i) = a_i$, $\delta(0, a_i, S) = (i, \varepsilon) \in \delta$, $(0, a_i, S) \vdash_{M_p(P)} (i, \varepsilon, \varepsilon)$ for all $1 \leq i \leq k$ and the claim holds.

State	Input	Pushdown Store
0	$a2\ a2\ a0\ a0\ b0$	S
1	$a2\ a0\ a0\ b0$	$S\ S$
2	$a0\ a0\ b0$	$S\ S\ S$
3	$a0\ b0$	$S\ S$
4	$b0$	S
5	ε	ε
accept		
0	$a2\ b1\ a0\ a0$	S
1	$b1\ a0\ a0$	$S\ S$
6	$a0\ a0$	$S\ S$
7	$a0$	S
8	ε	ε
accept		
0	$a2\ a0\ a0$	S
1	$a0\ a0$	$S\ S$
9	$a0$	S
10	ε	ε
accept		

Fig. 12. Trace of deterministic PDA $M_p(P)$ from Example 8 for trees in prefix notation $\{a2\ a2\ a0\ a0\ b0,\ a2\ b1\ a0\ a0,\ a2\ a0\ a0\}$

2. Assume that the claim holds for trees $t_1^1, t_2^1, \ldots, t_{p_1}^1, t_1^2, t_2^2, \ldots, t_{p_2}^2, \ldots, t_1^k, t_2^k, \ldots, t_{p_k}^k$ where $p_i \geq 1$ for all $1 \leq i \leq k$, *Height*$(t_1^i) \leq m$, *Height*$(t_2^i) \leq m$, ..., *Height*$(t_p^i) \leq m$, $m \geq 0$, for all $1 \leq i \leq k$.
   We have to prove that the claim holds also for each tree t_i, $1 \leq i \leq k$, such that
   pref$(t_i) = a_i$ *pref*(t_1^i)*pref*$(t_2^i) \ldots$ *pref*$(t_{p_i}^i)$, *Arity*$(a_i) = p_i$, and *Height*$(t_i) \geq m+1$:
   Since $\delta(0, a_i, S) = (i, S^p) \in \delta$, and $(0, a$ *pref*(t_1^i)*pref*$(t_2^i) \ldots$ *pref*$(t_{p_i}^i), S)$
   $\vdash_{M_p(t_i)} (i,$ *pref*(t_1^i)*pref*$(t_2^i) \ldots$ *pref*$(t_{p_i}^i), S^p)$
   $\vdash^*_{M_p(t_i)} (j^i,$ *pref*$(t_2^i) \ldots$ *pref*$(t_{p_i}^i), S^{p_i - 1})$
   $\vdash^*_{M_p(t_i)} \cdots$
   $\vdash^*_{M_p(t_i)} (\ell^i,$ *pref*$(t_{p_i}^i), S)$
   $\vdash^*_{M_p(t_i)} (f^i, \varepsilon, \varepsilon)$
   the claim holds for that tree.

Thus, the lemma holds. □

The deterministic subtree matching PDA for multiple tree patterns in prefix notation can be constructed in a similar fashion to the subtree matching PDA for a single pattern. First, the PDA accepting a set of trees in their prefix notations, constructed by Alg. 4, is used to construct a nondeterministic subtree matching PDA by Alg. 5. The constructed nondeterministic subtree matching PDA is then transformed to the equivalent deterministic subtree matching PDA.

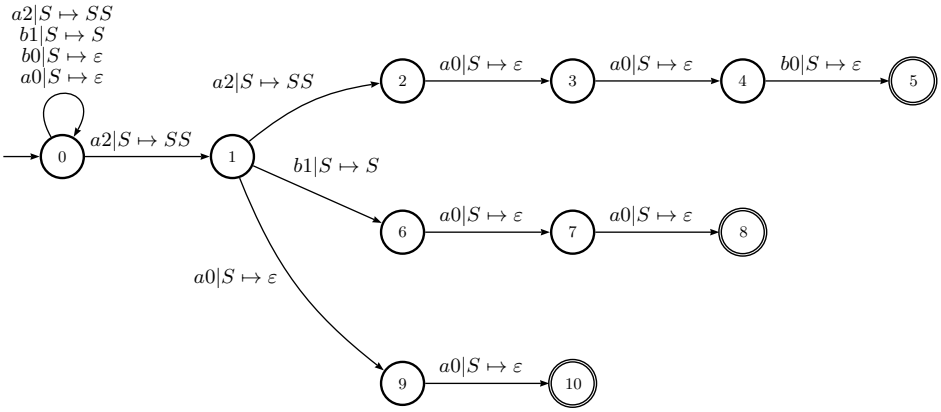


Fig. 13. Transition diagram of nondeterministic subtree matching PDA $M_p(P)$ constructed over trees in set P from Example 9

Algorithm 5. Construction of a nondeterministic subtree matching PDA for a set of trees $P = \{t_1, t_2, \ldots, t_m\}$ in their prefix notation.
Input: A tree t over a ranked alphabet $\mathcal{A}$; prefix notation *pref*$(t) = a_1 a_2 \ldots a_n$, $n \geq 1$.
Output: Nondeterministic subtree matching PDA $M_{nps}(t) = (Q, \mathcal{A}, \{S\}, \delta, 0, S, F)$.
Method:

1. Create PDA $M_{nps}(t)$ as PDA $M_p(t) = (Q, \mathcal{A}, \{S\}, \delta, 0, S, F)$ by Alg. 4.
2. For each symbol $a \in \mathcal{A}$ create a new transition $\delta(0, a, S) = (0, S^{\mathit{Arity}(a)})$, where $S^0 = \varepsilon$.

□

Example 9. The subtree matching PDA constructed by Alg. 2 over the set of trees P from Example 8 is the nondeterministic PDA $M_{nps}(P) = (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, \mathcal{A}, \{S\}, \delta_2, 0, S, \{5, 8, 10\}))$, where mapping δ_2 is a set of the following transitions:

$$\begin{array}{ll}
\delta_2(0, a2, S) = (1, SS) & \\
\delta_2(1, a2, S) = (2, SS) & \delta_2(0, a2, S) = (0, SS) \\
\delta_2(2, a0, S) = (3, \varepsilon) & \delta_2(0, b1, S) = (0, S) \\
\delta_2(3, a0, S) = (4, \varepsilon) & \delta_2(0, b0, S) = (0, \varepsilon) \\
\delta_2(4, b0, S) = (5, \varepsilon) & \delta_2(0, a0, S) = (0, \varepsilon) \\
\delta_2(1, b1, S) = (6, S) & \\
\delta_2(6, a0, S) = (7, \varepsilon) & \\
\delta_2(7, a0, S) = (8, \varepsilon) & \\
\delta_2(1, a0, S) = (9, \varepsilon) & \\
\delta_2(9, a0, S) = (10, \varepsilon) &
\end{array}$$

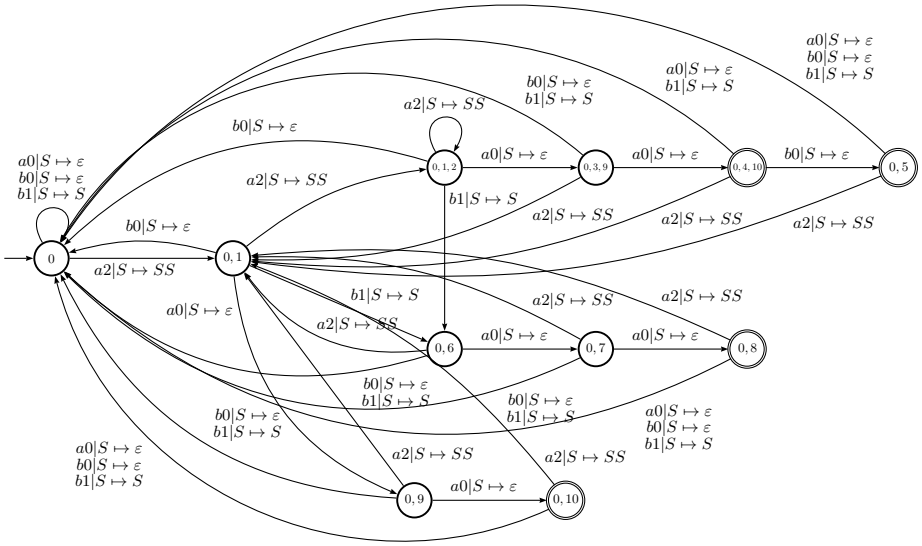


Fig. 14. Transition diagram of deterministic PDA $M_{dps}(P)$ constructed over trees in set P from Example 10

The transition diagram of nondeterministic PDA $M_{nps}(P)$ is illustrated in Fig. 13. □

Theorem 8. *Given a set of m trees $P = \{t_1, t_2, \ldots, t_m\}$ and their prefix notation pref(t_i), $1 \leq i \leq m$, the PDA $M_{nps}(P)$ constructed by Alg. 5 is a subtree matching PDA for tree patterns $t_1, t_2, \ldots, t_m$.*

Proof. According to Theorem 2, given an input tree t, each subtree in prefix notation is a substring of *pref*(t). Since the PDA $M_{nps}(P)$ has just states and transitions equivalent to the states and transitions, respectively, of the Aho-Corasick string matching automaton , the PDA $M_{nps}(P)$ accepts all matches of subtrees $t_1, t_2, \ldots, t_m$ in tree t by final state. □

For the construction of deterministic subtree PDA, we use the transformation described by Alg. 3 from Section 4.

The deterministic subtree matching automaton $M_{dps}(P)$ for a set of trees $P = \{t_1, t_2, \ldots, t_m\}$ with prefix notations *pref*$(t_i), 1 \leq i \leq k$ is demonstrated by the following example.

Example 10. The deterministic subtree matching PDA for the set of trees P from Example 8, constructed by Alg. 3 from the nondeterministic subtree matching PDA $M_{nps}(P)$ from Example 9, is $M_{dps}(P) = (\{[0], [0,1], [0,1,2], [0,3,9], [0,4,10], [0,5], [0,6], [0,7], [0,8], [0,9], [0,10]\}, \mathcal{A}, \{S\}, \delta_3, [0], S, \{[0,4,10], [0,5], [0,8], [0,10]\})$, with its transition diagram illustrated in Fig. 14.

We note that the deterministic subtree matching PDA $M_{dps}(P)$ has a very similar transition diagram to the Aho-Corasick string matching automaton constructed for the strings representing the prefix notations of trees in set P from Example 8 (see also [1, 9, 22]), as can be seen by comparing Figs. 4 and 14.

Fig. 15 shows the sequence of transitions (trace) performed by the deterministic subtree PDA $M_{dps}(P)$ for the input tree t having prefix notation *pref*$(t) = a2\ a2\ a2\ a0\ a0\ a2\ a2\ a0\ a0\ b0\ \ a2\ b1\ a0\ a0$. The final states are $\{[0, 4, 10], [0, 5], [0, 8], [0, 10]\}$. Fig. 16 depicts the pattern subtrees from set P and the input tree t. □

State	Input		PDS
$\{0\}$	$a2\ a2\ a2\ a0\ a0\ a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$		S
$\{0, 1\}$	$a2\ a2\ a0\ a0\ a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$		SS
$\{0, 1, 2\}$	$a2\ a0\ a0\ a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$		SSS
$\{0, 1, 2\}$	$a0\ a0\ a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$		$SSSS$
$\{0, 3, 9\}$	$a0\ a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$		SSS
$\{0, 4, 10\}$	$a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$	match	SS
$\{0, 1\}$	$a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$		SSS
$\{0, 1, 2\}$	$a0\ a0\ b0\ a2\ b1\ a0\ a0$		$SSSS$
$\{0, 3, 9\}$	$a0\ b0\ a2\ b1\ a0\ a0$		SSS
$\{0, 4, 10\}$	$b0\ a2\ b1\ a0\ a0$	match	SS
$\{0, 5\}$	$a2\ b1\ a0\ a0$	match	S
$\{0, 1\}$	$b1\ a0\ a0$		SS
$\{0, 6\}$	$a0\ a0$		SS
$\{0, 7\}$	$a0$		S
$\{0, 8\}$	ε	match	ε

Fig. 15. Trace of deterministic subtree PDA $M_{dps}(P)$ from Example 10 for tree t_2 in prefix notation *pref*$(t) = a2\ a2\ a2\ a0\ a0\ a2\ a2\ a0\ a0\ b0\ a2\ b1\ a0\ a0$.

Theorem 9. *Given a set of m trees $P = \{t_1, t_2, \ldots, t_m\}$ over a ranked alphabet $\mathcal{A}$, the deterministic subtree matching PDA $M_{pds}(P)$ is constructed by Alg. 5 and 3 in time $\Theta(|\mathcal{A}|s)$, requires $\Theta(|\mathcal{A}|s)$ storage, where $s = \sum_{i=1}^{m} |t_i|$, and its pushdown store alphabet consists of one symbol.*

Proof. Since the subtree matching PDA for multiple patterns is directly analogous to the Aho-Corasick string matching automaton (this can be proved from proof of Theorem 6), we can use the proof from [1] and [26]. □

Theorem 10. *Given an input tree t with n nodes, the searching phase of the deterministic subtree matching automaton constructed by Algs. 2 and 3 over a set of m trees P is $\mathcal{O}(n)$.*

Proof. The searching phase consists of reading tree t once, symbol by symbol from left to right. The appropriate transition is taken each time a symbol is read,

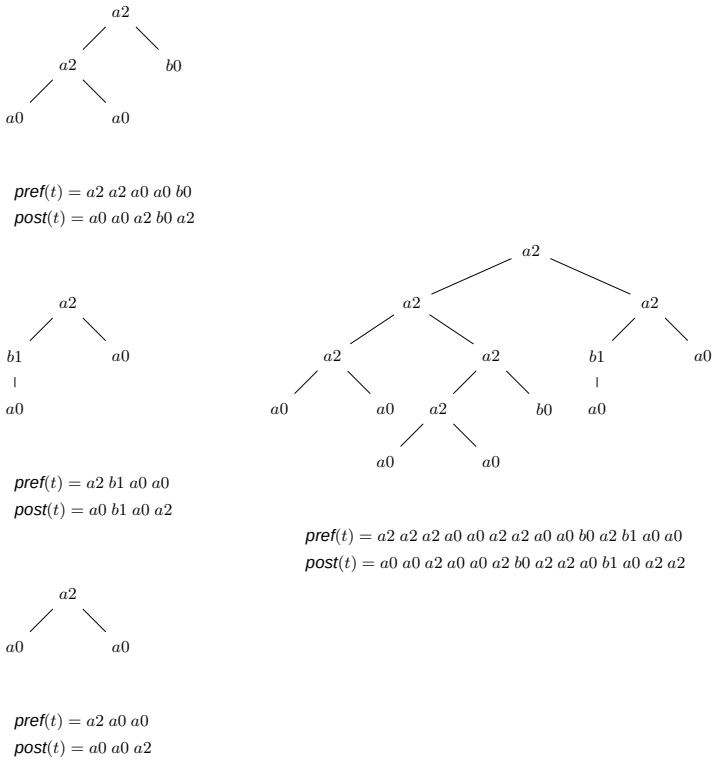


Fig. 16. Pattern subtrees from set P and the input tree from Example 10 along with their prefix and postfix notations

resulting in exactly n transitions. Each transition consumes a constant time because the time of each pushdown operation is limited by the maximal arity of nodes. Occurrences of the subtree to find are matched by transitions leading to the final states. □

6. Subtree matching in postfix notation

In this section we show the dual principle for the postfix notation. Theorems 11 and 12 present the direct analogy of properties of the prefix and postfix notations. Theorem 13 is analogous to Theorem 3.

Theorem 11. *Given a tree t and its postfix notation post(t), all subtrees of t in postfix notation are substrings of post(t).*

Theorem 12. *Let post(t) and w be a tree t in postfix notation and a substring of post(t), respectively. Then, w is the postfix notation of a subtree of t, if and only if $ac(w) = 0$, and $ac(w_1) \leq -1$ for each w_1, where $w = xw_1$, $x \neq \varepsilon$.*

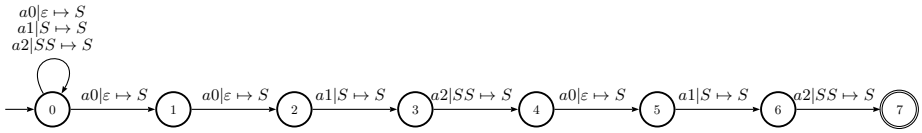


Fig. 17. Transition diagram of nondeterministic subtree matching PDA $M_p(t_1)$ for tree t_1 in postfix notation *post*$(t_1) = a0\ a0\ a1\ a2\ a0\ a1\ a2$ from Example 6

Theorem 13. *Let $M = (\{Q, \mathcal{A}, \{S\}, \delta, 0, S, F)$ be an input–driven PDA whose each transition from δ is of the form $\delta(q_1, a, S^i) = (q_2, S)$, where $i =$ Arity(a). Then, if $(q_3, w, \varepsilon) \vdash_M^+ (q_4, \varepsilon, S^j)$, then $j = -ac(w) + 1$.*

From the above Theorems, we can easily transform Algorithms 1-5 to work with the postfix notation of trees. The only change required is in the pushdown operations. All transitions of the form $\delta(q, a, S) = (p, S^{\text{Arity}(a_i)})$ must be changed to the form $\delta(q, a, S^{\text{Arity}(a_i)}) = (p, S)$. The subtree matching PDA also requires no initial pushdown store symbol, while after processing a valid tree in postfix notation, the pushdown store contains a single symbol 'S'.

Fig. 17 illustrates the nondeterministic subtree matching PDA $M_p(t_1)$ constructed from the postfix notation of the tree from Example 6.

Fig. 18 illustrates the deterministic subtree matching PDA $M_dps(t_1)$ constructed from the postfix notation of the tree from Example 6.

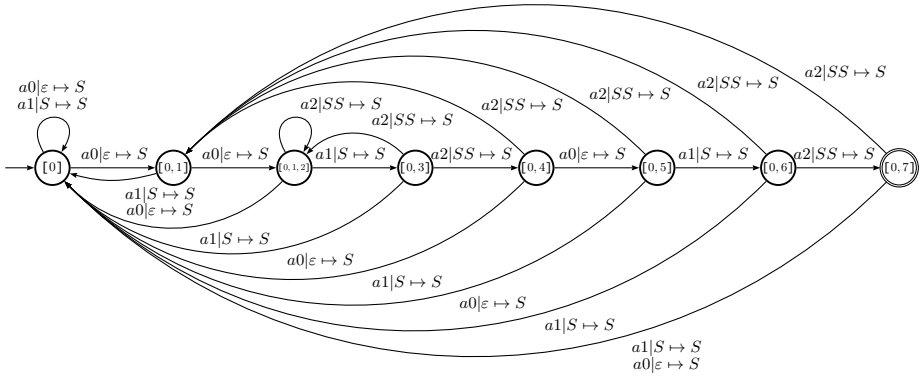


Fig. 18. Transition diagram of deterministic PDA $M_{dps}(t_1)$ for tree t_1 in postfix notation *post*$(t_1) = a0\ a0\ a1\ a2\ a0\ a1\ a2$ from Example 7

7. Conclusion

We have introduced a new kind of pushdown automata: subtree matching PDAs for trees in prefix and postfix notations. These pushdown automata are in their properties analogous to string matching automata, which are widely used in stringology [9, 10, 22, 26].

Regarding specific tree algorithms whose model of computation is the standard deterministic pushdown automaton, we have recently introduced principles of other three new algorithms. First, the tree pattern matching PDA [13, 21] which is an extension of the subtree matching PDA presented in this paper. Second, the subtree and tree pattern PDAs, which represent a complete index of a given tree by preprocessing it. Searching for all occurrences of a subtree or a tree pattern of size m is then performed in time linear to m and not depending on the size of the preprocessed tree [17, 19, 21]. These automata representing indexes of trees are analogous in their properties to the string suffix and factor automata [9, 10, 22, 26]. Third, a method on how to find all repeats of connected subgraphs in trees with the use of subtree or tree pattern PDAs [21, 20]. More details on these results and related information can also be found on [3].

References

1. Aho, A.V., Corasick, M.J.: Efficient string matching: an aid to bibliographic search. Commun. ACM 18(6), 333–340 (1975)
2. Aho, A.V., Ullman, J.D.: The theory of parsing, translation, and compiling. Prentice-Hall Englewood Cliffs, N.J. (1972)
3. Arbology www pages. Available on: http://www.arbology.org/ (2009), december 2009
4. Berstel, J.: Transductions and Context-Free Languages. Teubner Studienbucher, Stuttgart (1979)
5. Chase, D.R.: An improvement to bottom-up tree pattern matching. In: POPL. pp. 168–177 (1987)
6. Cleophas, L.: Tree Algorithms. Two Taxonomies and a Toolkit. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven (2008)
7. Cole, R., Hariharan, R., Indyk, P.: Tree pattern matching and subset matching in deterministic ($\log^3$)-time. In: SODA. pp. 245–254 (1999)
8. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata (2007), release October, 12th 2007
9. Crochemore, M., Hancart, C.: Automata for matching patterns. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 2 Linear Modeling: Background and Application, chap. 9, pp. 399–462. Springer–Verlag, Berlin (1997)
10. Crochemore, M., Rytter, W.: Jewels of Stringology. World Scientific, New Jersey (1994)
11. Dubiner, M., Galil, Z., Magen, E.: Faster tree pattern matching. J. ACM 41(2), 205–213 (1994)
12. Flouri, T., Janoušek, J., Melichar, B.: Subtree matching by deterministic pushdown automata. In: Ganzha, M., Paprzycki, M. (eds.) Proceedings of the IMCSIT, Vol. 4. pp. 659–666. IEEE Computer Society Press (2009)

13. Flouri, T., Janoušek, J., Melichar, B.: Tree pattern matching by deterministic pushdown automata (2009), draft
14. Gecseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3 Beyond Words. Handbook of Formal Languages, pp. 1–68. Springer–Verlag, Berlin (1997)
15. Hoffmann, C.M., O'Donnell, M.J.: Pattern matching in trees. J. ACM 29(1), 68–95 (1982)
16. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley, Boston, 2nd edn. (2001)
17. Janoušek, J.: String suffix automata and subtree pushdown automata. In: Holub, J., Žďárek, J. (eds.) Proceedings of the Prague Stringology Conference 2009. pp. 160–172. Czech Technical University in Prague, Czech Republic (2009), available on: http://www.stringology.org/event/2009
18. Janoušek, J., Melichar, B.: On regular tree languages and deterministic pushdown automata. Acta Inf. 46(7), 533–547 (2009)
19. Janoušek, J., Melichar, B.: Subtree and tree pattern pushdown automata for trees in prefix notation (2009), submitted for publication
20. Janoušek, J., Melichar, B.: Finding repeats of subtrees in a tree using pushdown automata (2010), submitted for publication
21. London stringology days 2009 conference presentations. Available on: http://www.dcs.kcl.ac.uk/events/LSD&LAW09/, King's College London, London (2009)
22. Melichar, B., Holub, J., Polcar, J.: Text searching algorithms. Available on: http://stringology.org/athens/ (2005), release November 2005
23. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer–Verlag, Berlin (1997)
24. Rozenberg, G., Salomaa, A. (eds.): Vol. 1: Word, Language, Grammar, Handbook of Formal Languages. Springer–Verlag, Berlin (1997)
25. Shankar, P., Gantait, A., Yuvaraj, A.R., Madhavan, M.: A new algorithm for linear regular tree pattern matching. Theor. Comput. Sci. 242(1-2), 125–142 (2000)
26. Smyth, B.: Computing Patterns in Strings. Addison-Wesley-Pearson Education Limited, Essex, England (2003)
27. Valiant, L.G., Paterson, M.: Deterministic one-counter automata. In: Automaten theorie und Formale Sprachen. pp. 104–115 (1973)
28. Wagner, K., Wechsung, G.: Computational Complexity. Springer–Verlag, Berlin (2001)

Tomáš Flouri graduated at the Department of Computer Science and Engineering, Faculty of Electrical Engineering of the Czech Technical University in Prague in 2008. Since 2008 he has been a Ph.D. student at the same department. His scientific research focuses on tree algorithms using pushdown automata.

Jan Janoušek graduated the Department of Computer Science and Engineering, Faculty of Electrical Engineering of the Czech Technical University in Prague in 1994. He received his Ph.D. in the field of parsing and translation in 2001. His research interests include tree algorithms, parsing and translation algorithms

and attribute grammars. Since July 2009 he has been working as an assistant professor at the Department of Theoretical Computer Science, Faculty of Information Technology of the Czech Technical University in Prague.

Bořivoj Melichar graduated the Faculty of Electrical Engineering of the Czech Technical University in Prague in 1964. During 1964 - 2009 he worked at the Department of Computer Science and Engineering, Faculty of Electrical Engineering of the Czech Technical University in Prague. His interests include parsing and translation algorithms, attribute grammars, and text and tree algorithms. Since the beginning of 2010 he has been working as a full professor at the Department of Theoretical Computer Science, Faculty of Information Technology of the Czech Technical University in Prague.

Received: November 16, 2009; Accepted: December 25, 2009.