Graphs and Automata

Bořivoj Melichar

melichar@fit.cvut.cz
http://www.arbology.org

Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University
Thakurova 9, 160 00 Prague 6, Czech Republic

Prague Stringology Conference 2013

Chomsky classification

Type of grammars	Type of automata
regular grammars	finite automata
context-free grammars	pushdown automata
context-sensitive grammars	linear bounded automata
unrestricted grammars	Turing machines

Linear notation

Statement:

Do traversing the structure and perform following operations. . .

Such statement leads to a linearisation of the structure in question. There is a possibility to divide such process into two parts:

1. Creating a linear notation of the structure
2. Processing the linear notation of the structure

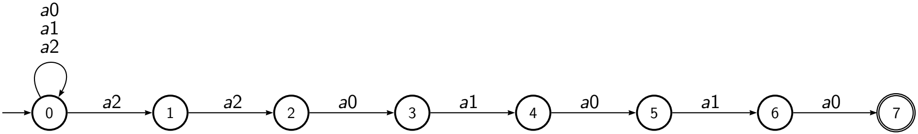
Graphs and automata

Type of graphs	Type of automata	Discipline
“linear” graphs	finite automata	stringology
trees	pushdown automata	arbology
directed acyclic graphs	linear bounded automata	dagology
general graphs	Turing machines	?

Pattern matching

Example

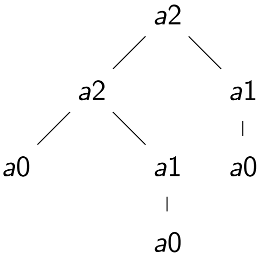
String: $t = a2\ a2\ a0\ a1\ a0\ a1\ a0$

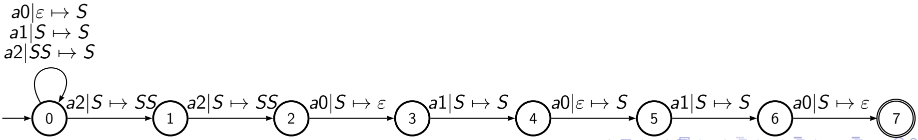


Pattern matching

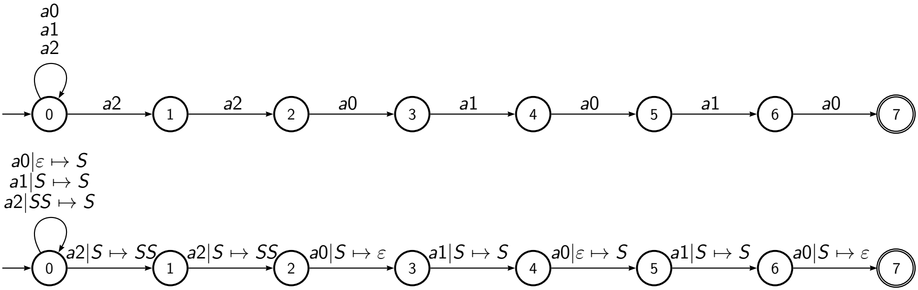
Example

String: $t = a2\ a2\ a0\ a1\ a0\ a1\ a0$ – prefix notation

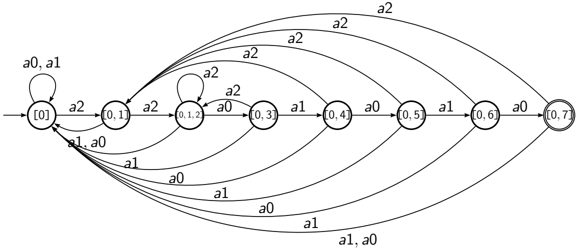


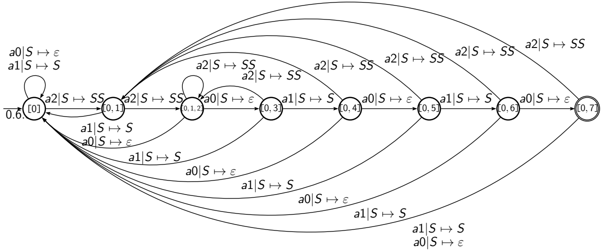


Pattern matching

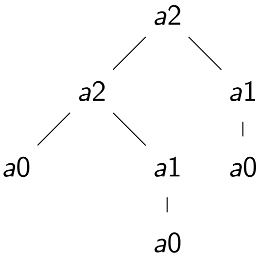


Deterministic finite and pushdown automata



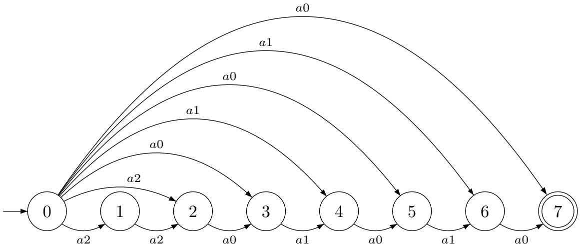


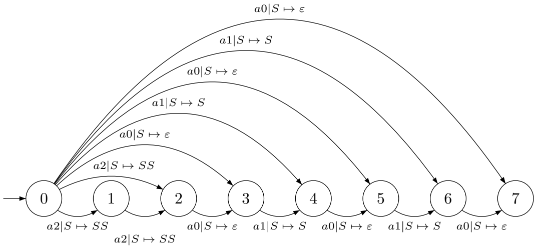
Indexing



Indexing

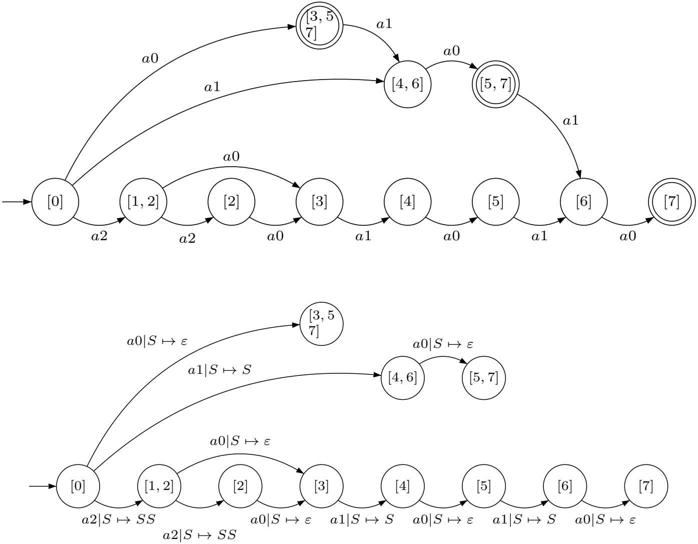
Nondeterministic factor and subtree PDA





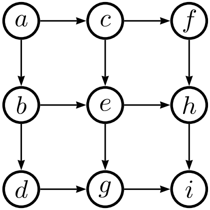
Indexing

Deterministic factor and subtree PDA



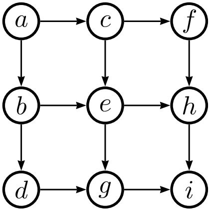
Directed acyclic graph and linear notation

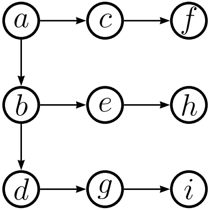
a	c	f
b	e	h
d	g	i

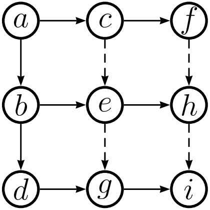


Directed acyclic graph and linear notation

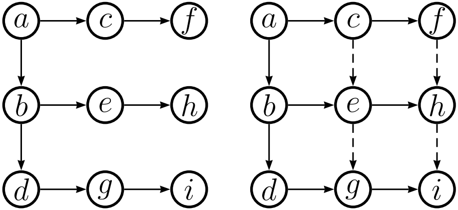
a	c	f
b	e	h
d	g	i



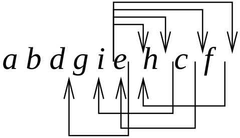




Directed acyclic graph and linear notation



$\mathrm{pref}(\mathrm{tree}(T)) = abdgiehcf$



References

Arbology www pages:
Available on: http://www.arbology.org, July 2009.

B. Melichar:
Arbology: Trees and pushdown automata, in Language and Automata Theory and Applications, A.-H. Dediu, H. Fernau, and C. Martín-Vide, eds., vol. 6031 of Lecture Notes in Computer Science, Springer-Verlag, Berlin/Heidelberg, 2010, pp. 32–49.

B. Melichar, J. Holub, and T. Polcar:
Text searching algorithms.
Available on: http://www.stringology.org/athens/, Nov. 2005.

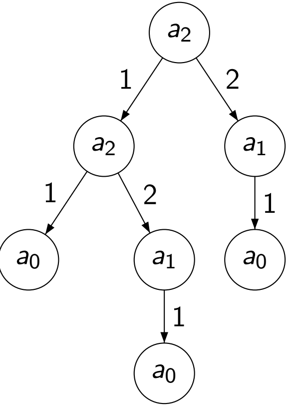
J. Žďárek:
Two-dimensional Pattern Matching Using Automata Approach, PhD thesis, Czech Technical University in Prague, 2010, Available on: http://www.stringology.org/papers/Zdarek-PhD_thesis-2010.pdf.

Stringology and Arbology

Every sequential algorithm must do some linearisation of a tree.

Parallel algorithm must do some linearisation of a tree “per partes”.

Linear notations of trees

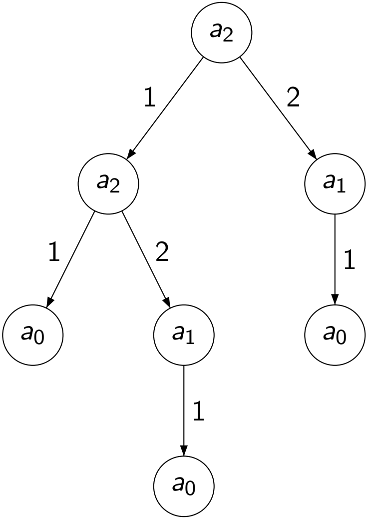


prefix	$a_2\ a_2\ a_0\ a_1\ a_0\ a_1\ a_0$
postfix	$a_0\ a_0\ a_1\ a_2\ a_0\ a_1\ a_2$
Euler	$a_2\ a_2\ a_0\ a_2\ a_1\ a_0\ a_1\ a_2\ a_2\ a_1\ a_0\ a_1\ a_2$
bracketted prefix	$[\ a_2\ [\ a_2\ [\ a_0\ ]\ [\ a_1\ [\ a_0\ ]\ ]\ ]\ [\ a_1\ [\ a_0\ ]\ ]\ ]$
bar prefix	$a_2\ a_2\ a_0\ \vert\ a_1\ a_0\ \vert\ \vert\ \vert\ a_1\ a_0\ \vert\ \vert\ \vert$
bracketted postfix	$[\ [\ [\ a_0\ ]\ [\ [\ a_0\ ]\ a_1\ ]\ a_2\ ]\ [\ [\ a_0\ ]\ a_1\ ]\ a_2\ ]$
bar postfix	$\vert\ \vert\ \vert\ a_0\ \vert\ \vert\ a_0\ a_1\ a_2\ \vert\ \vert\ a_0\ a_1\ a_2$

It holds that subtrees in a linear notation are substrings of the tree in the linear notation. This implies analogous pushdown automata for all such linear notations.

Tree – the simplest case

- Rooted
- Oriented
- Ordered
- Ranked



Prefix notation

Grammar for tree with nodes having rank 0, 1, 2:

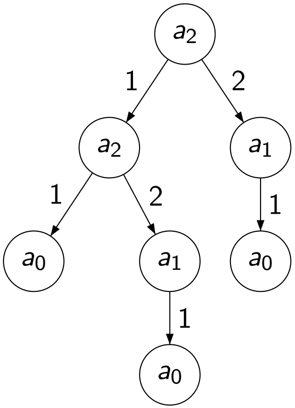
(1) $S \rightarrow a_0$

(2) $S \rightarrow a_1\ S$

(3) $S \rightarrow a_2\ S\ S$

(Greibach normal form, simple LL(1) grammar, LR(0) grammar)

Example: $a_2\ a_2\ a_0\ a_1\ a_0\ a_1\ a_0$



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

Postfix notation

Grammar for tree with nodes having rank 0, 1, 2:

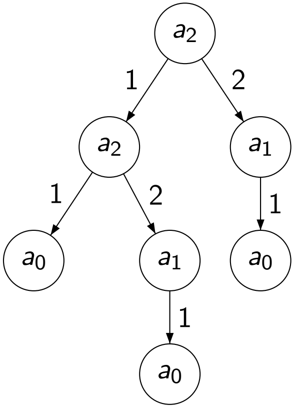
(1) $S \rightarrow a_0$

(2) $S \rightarrow S\ a_1$

(3) $S \rightarrow S\ S\ a_2$

(Reversed Greibach normal form, strong LR(1) grammar)

Example: $\underline{\underline{a_0\ \underline{\underline{a_0\ a_1}}\ a_2}\ \underline{a_0\ a_1}\ a_2}$



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

Well known principles

- Scanning of tree

recursive procedures

- Covering of tree
- Construction of tree
- Target program generation

} context-free parsing (top down, bottom up)

Pushdown automata

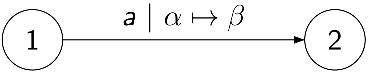
Stringology

Basic tool: Finite automata

Arbology

Basic tool: (Deterministic) Pushdown automata

Pushdown automaton – notation



transition from state 1 to state 2

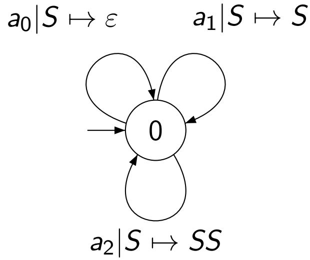
reading a, pop α, push β

Basic deterministic pushdown automaton for prefix notation of tree with nodes having rank 0, 1, 2:

(1) $S \to a_0$

(2) $S \to a_1\ S$

(3) $S \to a_2\ S\ S$



$$\delta(0, a_0, S) = (0, \varepsilon)$$
$$\delta(0, a_1, S) = (0, S)$$
$$\delta(0, a_2, S) = (0, SS)$$

$$\delta(q, a, S) = \{(q, \alpha) : S \to a\alpha \in P\}$$

Accept by empty pushdown store.

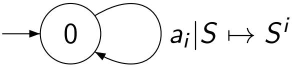
Basic deterministic pushdown automata for tree with nodes having rank 0, 1, 2, . . . , n:

Prefix notation:

$(i+1)\quad S \to a_i S^i$

where

$i = 0, 1, 2, \ldots, n.$



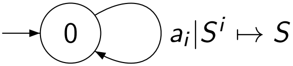
$$\delta(0, a_i, S) = (0, S^i)$$

Postfix notation:

$(i+1)\quad S \to S^i a_i$

where

$i = 0, 1, 2, \ldots, n.$



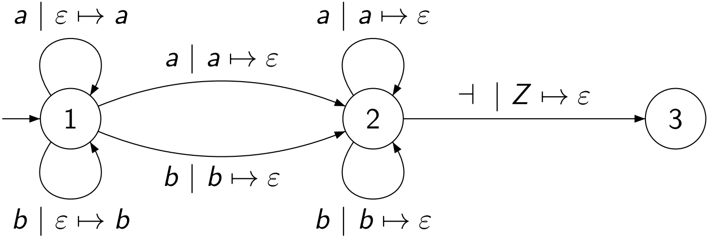
$$\delta(0, a_i, S^i) = (0, S)$$

$$S^0 = \varepsilon$$

Accept by empty pushdown store.

Determinisation of pushdown automata

Not always possible: $L = \{w\ w^R\ \dashv :\ w \in \{a,b\}^+\}$



Determinisation is possible for:

1. Input–driven pushdown automata

2. Visibly pushdown automata

3. Height–deterministic pushdown automata

Determinisation of input–driven PDA

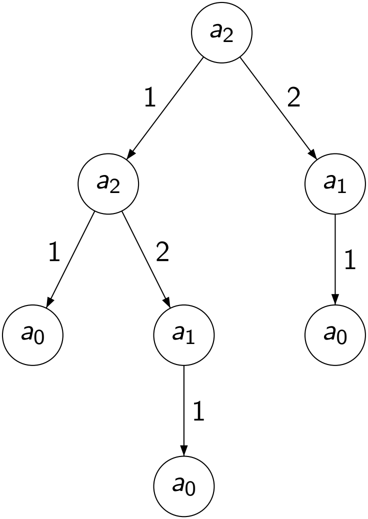
Input–driven PDA – pushdown store operations are determined by the input symbol.

Any nondeterministic input–driven PDA can be determinised similarly as in the case of finite automata – the states of the deterministic PDA correspond to subsets of states of the nondeterministic PDA (d–subsets).

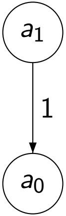
Moreover, nondeterministic acyclic input–driven PDA – the contents of the pushdown store can be precomputed, and only transitions and states with possible pushdown operations are selected.

Tree pattern matching

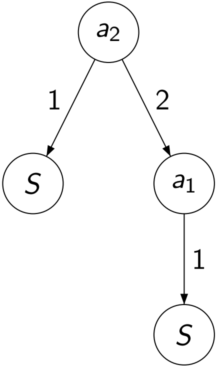
Object Tree



Subtree



Tree template



Prefix notation

$a_1 \; a_0$

Prefix notation

$a_2 \; S \; a_1 \; S$

2. Subtree and tree pattern pushdown automata – Indexing trees

Motivation

Stringology
String suffix and factor automata.
Properties:

1. Accept all occurences of an input suffix and an input factor, respectively, in a text of size n.
2. Search phase for all occurences of an input suffix or an input factor of size $\mathbf{m}$ in time $\mathcal{O}(\mathbf{m})$, and not depending on $\mathbf{n}$.
3. Although the number of factors in the text is $\mathcal{O}(\mathbf{n^2})$, the total size of the deterministic factor automaton is $\mathcal{O}(\mathbf{n})$.

Motivation

Arbology

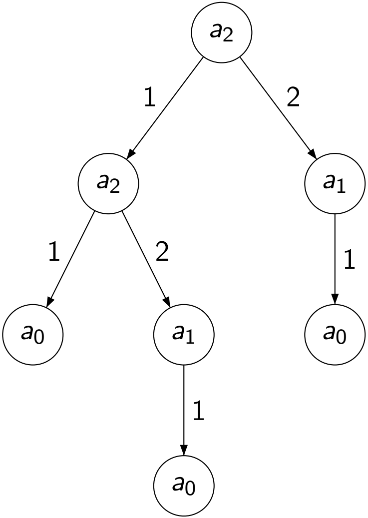
Subtree and tree pattern pushdown automata – analogous to string suffix and factor automata.

Properties:

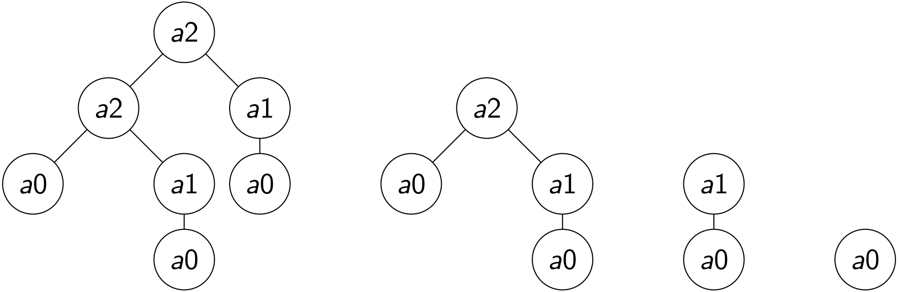
1. Accept all occurences of an input subtree and of subtrees matching an input tree pattern, respectively, in a tree of size n.
2. Search phase for all occurences of an input subtree or an input tree pattern of size $\mathbf{m}$ in time $\mathcal{O}(\mathbf{m})$, and not depending on $\mathbf{n}$.
3. Although the number of tree patterns matching the tree can be $\mathcal{O}(\mathbf{2^n})$, the total size of the deterministic tree pattern pushdown automaton is in specific cases $\mathcal{O}(\mathbf{n})$. This total size generally is an open question – we guess it is $\mathcal{O}(\mathbf{n^2})$.

Subtree PDA
Example 1

- Ranked alphabet
  $\mathcal{A} = \{a2, a1, a0\}$
- Tree t_1
  prefix notation is
  $pref(t_1) =$
  $a2\ a2\ a0\ a1\ a0\ a1\ a0$
- Different subtrees of t_1
  in prefix notation are:
  1. $a2\ a2\ a0\ a1\ a0\ a1\ a0$
  2. $a2\ a0\ a1\ a0$
  3. $a1\ a0$
  4. $a0$



All subtrees of tree t_1 and their prefix notation



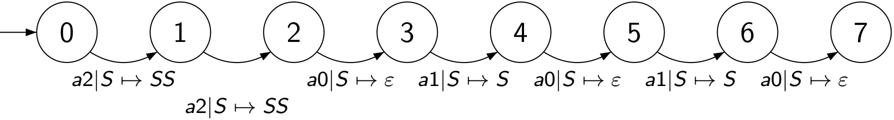
$a2\ a2\ a0\ a1\ a0\ a1\ a0$

$a2\ a0\ a1\ a0$

$a1\ a0$

$a0$

Transition diagram of deterministic PDA $M_p(t_1)$ accepting $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ by empty pushdown store



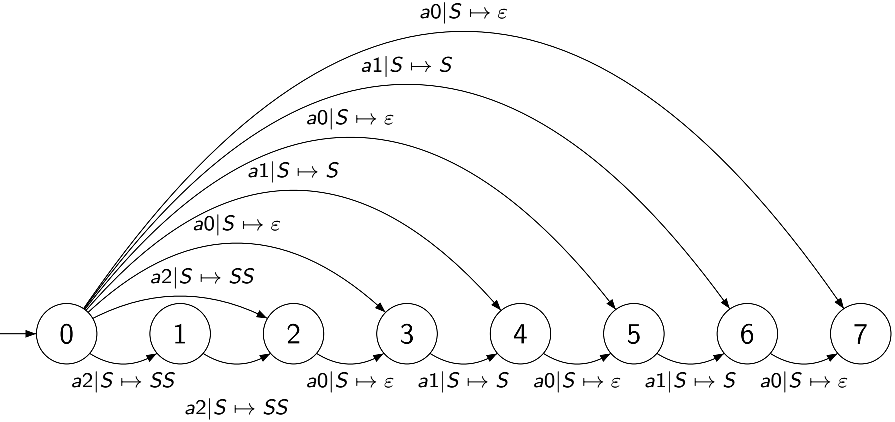
Initial contents of pushdown store is S.

Trace of deterministic PDA $M_p(t_1)$ for input string $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$

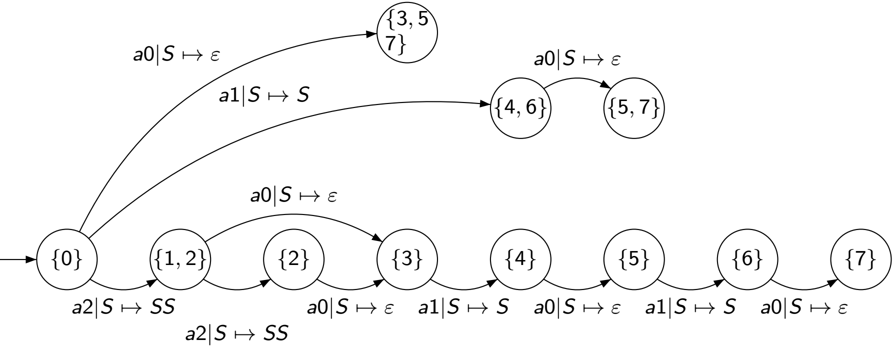
State	Pushdown Store	Input
0	S	$a2\ a2\ a0\ a1\ a0\ a1\ a0$
1	$S\ S$	$a2\ a0\ a1\ a0\ a1\ a0$
2	$S\ S\ S$	$a0\ a1\ a0\ a1\ a0$
3	$S\ S$	$a1\ a0\ a1\ a0$
4	$S\ S$	$a0\ a1\ a0$
5	S	$a1\ a0$
6	S	$a0$
7	ε	ε
accept		

accept by empty pushdown store

Nondeterministic subtree PDA $M_{nps}(t_1)$ for tree t_1 in prefix notation $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ (input–driven PDA)



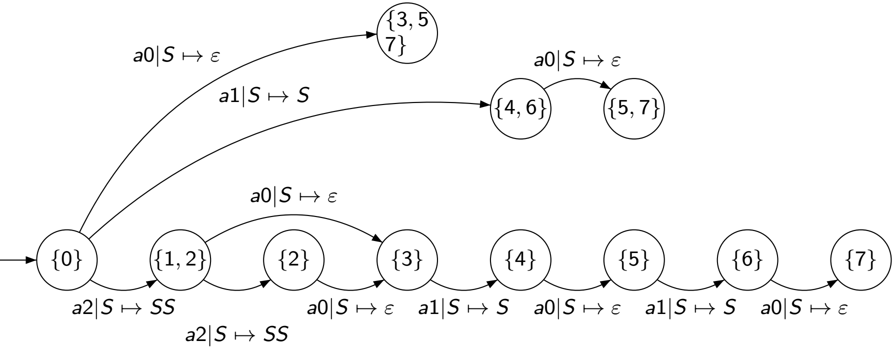
Deterministic subtree PDA $M_{dps}(t_1)$ for tree in prefix notation $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$



Given a tree t with n nodes and its prefix notation $pref(t)$, the total size of the deterministic subtree PDA $M_{dps}(t)$ is $\mathcal{O}(\mathbf{n})$. As for the case of the string factor automaton.

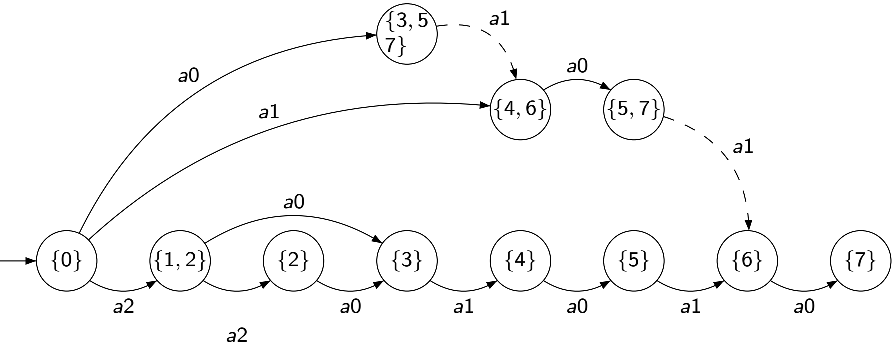
Deterministic subtree PDA $M_{dps}(t_1)$ vs. deterministic string factor automaton

Deterministic subtree PDA $M_{dps}(t_1)$

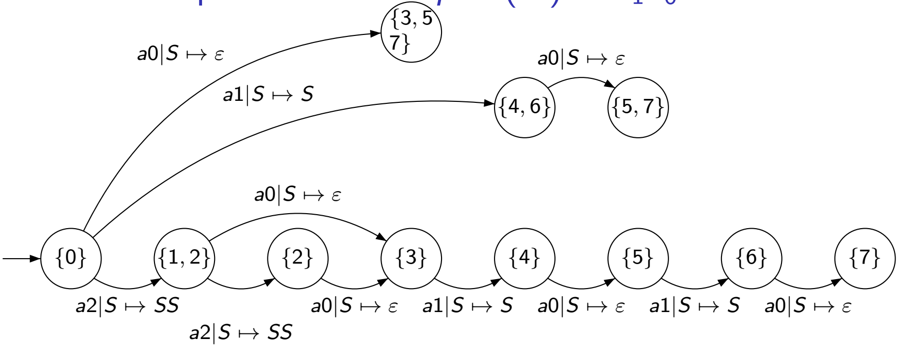


Deterministic subtree PDA $M_{dps}(t_1)$ vs. deterministic string factor automaton

Deterministic string factor automaton



Trace of deterministic subtree PDA $M_{dps}(t_1)$ for an input subtree *st* in prefix notation $pref(st) = a_1 a_0$

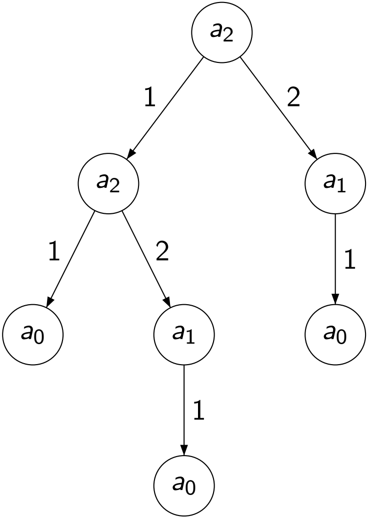


State	PDS	Input
$\{0\}$	S	$a1\ a0$
$\{4,6\}$	S	$a0$
$\{5,7\}$	ε	ε
accept		

Input subtree

Tree pattern PDA

List of some treetops of the tree

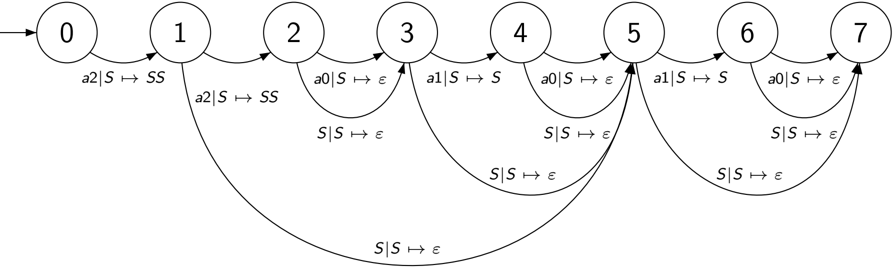


a2 S S
a2 a2 S S S
a2 a2 a0 S S
a2 a2 a0 a1 S S
a2 a2 a0 a1 a0 S
a2 a2 a0 a1 a0 a1 S
a2 a2 a0 a1 a0 a1 a0
a2 S a1 S
a2 S a1 a0
a2 a2 S S a1 S

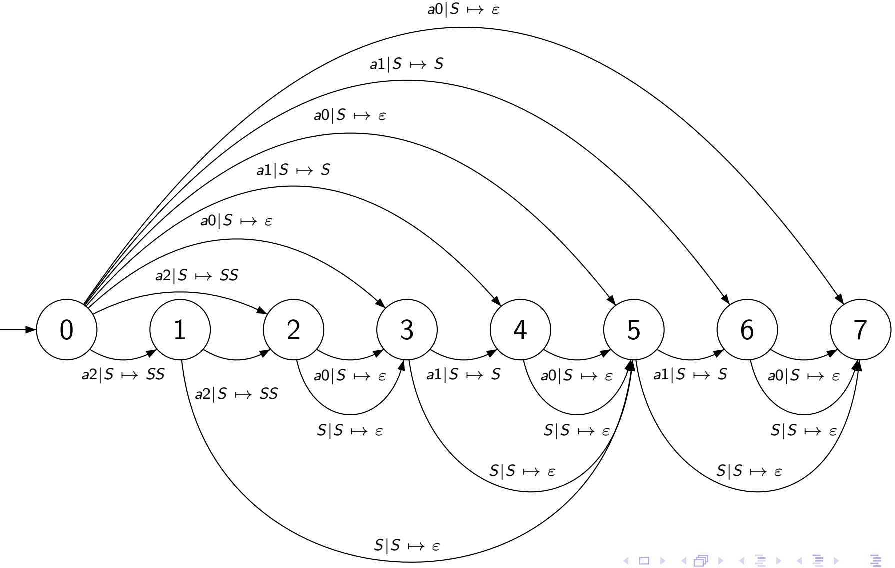
.
.
.

Deterministic treetop PDA $M_{pt}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$

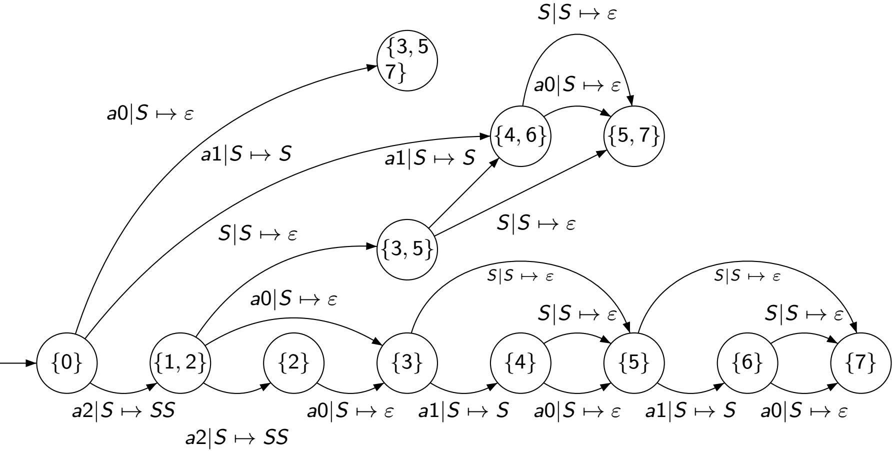
$srms = \{3, 5, 7\}$ Set of the Right-Most States



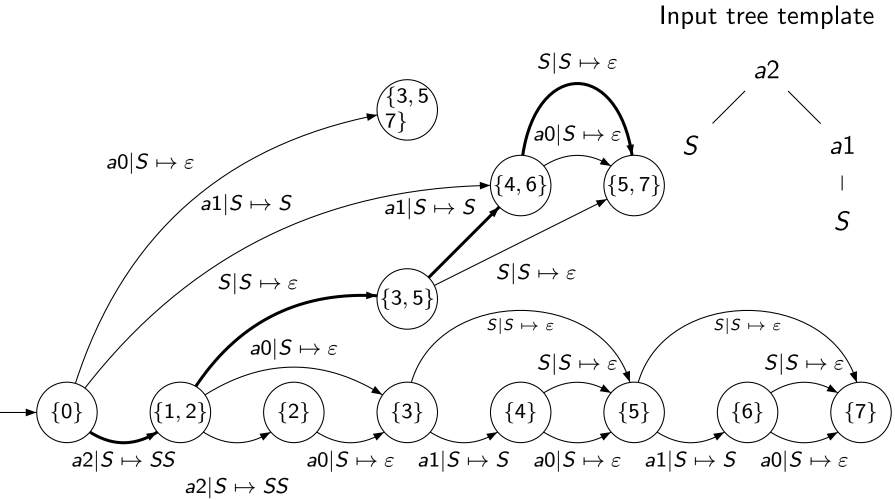
Nondeterministic tree pattern PDA $M_{npg}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$



Deterministic tree pattern PDA $M_{dpg}(t_1)$ for tree t_1 in prefix notation $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$



Trace of deterministic PDA M_{dpg} for prefix notation of tree pattern $a_2\ S\ a_1\ S$



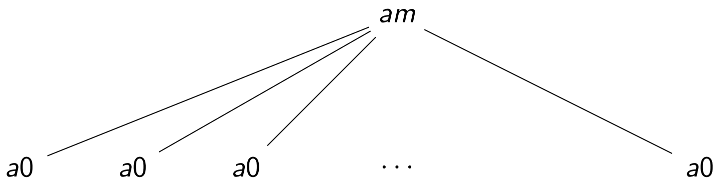
Complexity

Given a tree t with n nodes and its prefix notation $pref(t)$, the number of distinct tree templates matching the tree is less or equal $\mathbf{2^{n-1}}$.

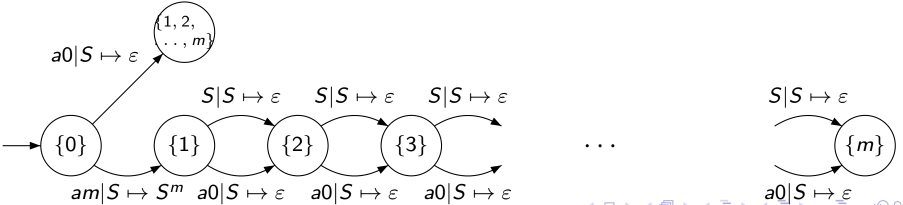
The total size of the deterministic tree pattern PDA $M_{dpg}(t)$ is in specific cases $\mathcal{O}(\mathbf{n})$. The total size generally is an open question – we guess it is $\mathcal{O}(\mathbf{n^2})$.

Example 2

tree t_2, $pref(t_2) = am\ a0^m$



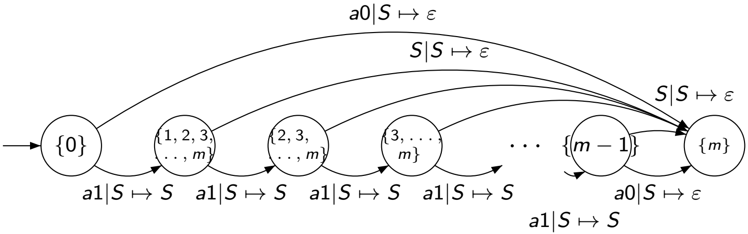
Deterministic tree pattern PDA for $pref(t_2)$:



Example 3

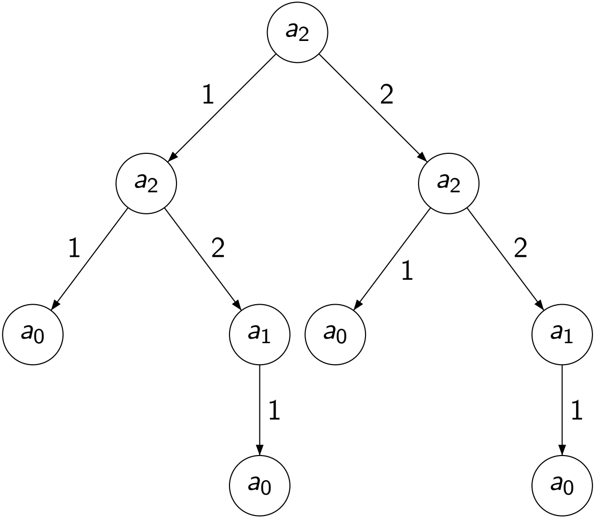
tree t_3, $pref(t_3) = a1^{m-1}\ a0$

Deterministic tree pattern PDA for $pref(t_3)$:



3. Repeats in Trees

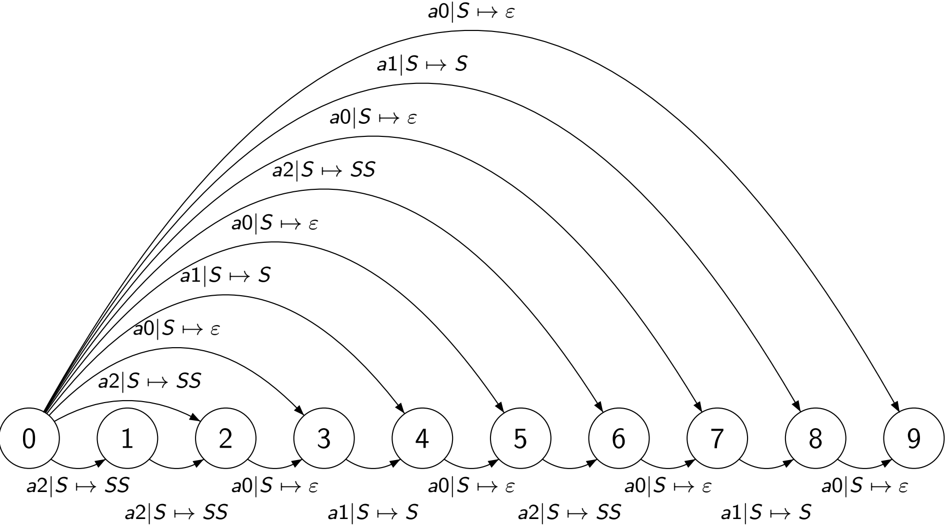
Subtree repeats



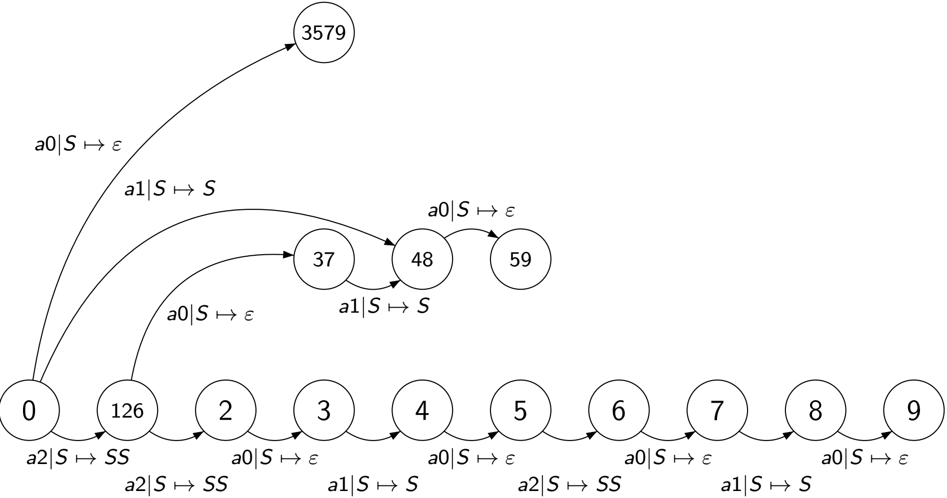
prefix notation

$a2\ a2\ a0\ a1\ a0\ a2\ a0\ a1 a0$

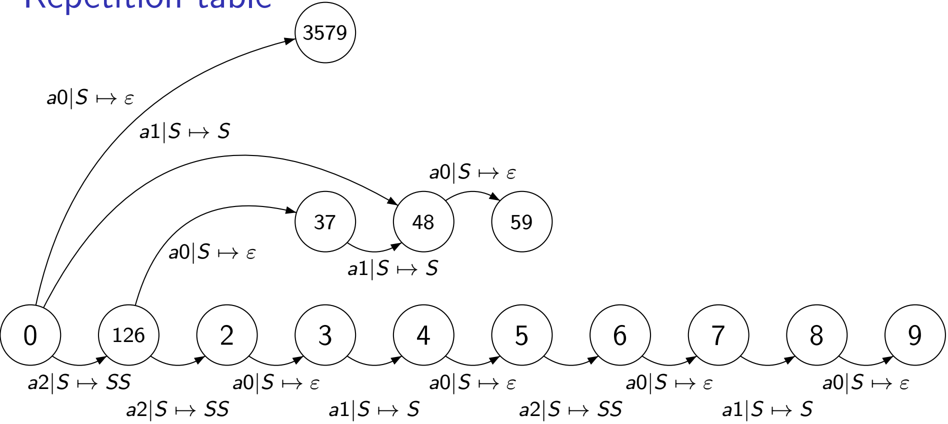
Nondeterministic subtree PDA $M_{nps}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a2\ a0\ a1a0$ (input–driven PDA)



Deterministic subtree PDA $M_{dps}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a2\ a0\ a1a0$



Repetition table

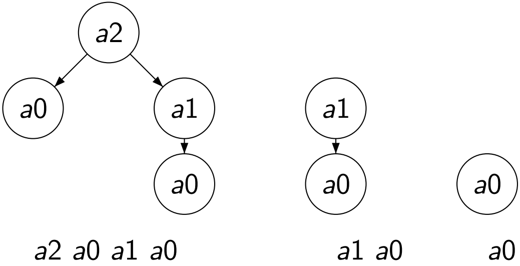


d-subset	Subtree	List of repeats
3579	$a0$	$(3, F), (5, G), (7, G), (9, G)$
59	$a1\ a0$	$(5, F), (9, G)$
	$a2\ a0\ a1\ a0$	$(5, F), (9, N)$

F first
G gap
N neighbour

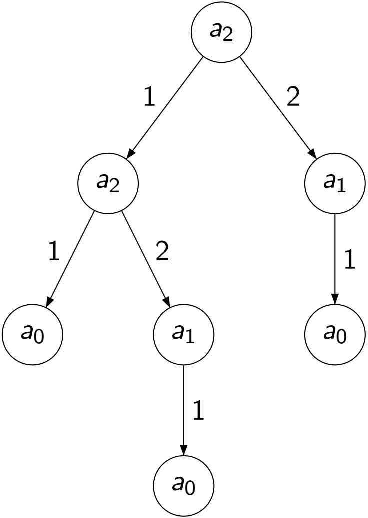
Repetition table

d-subset	Subtree	List of repeats
3579	$a0$	$(3, F), (5, G), (7, G), (9, G)$
59	$a1\ a0$	$(5, F), (9, G)$
	$a2\ a0\ a1\ a0$	$(5, F), (9, N)$



Overlapping is not possible, which follows from the basic property of tree!

Tree pattern repeats

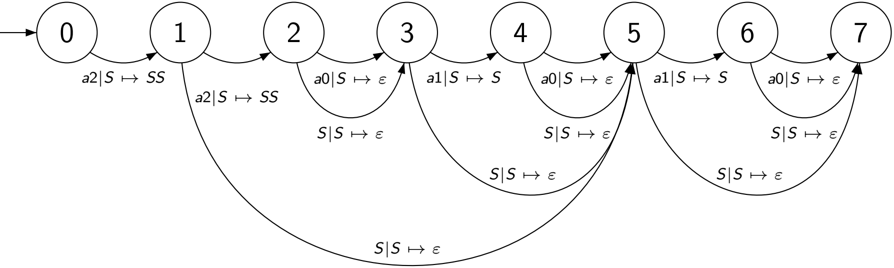


PREFIX NOTATION

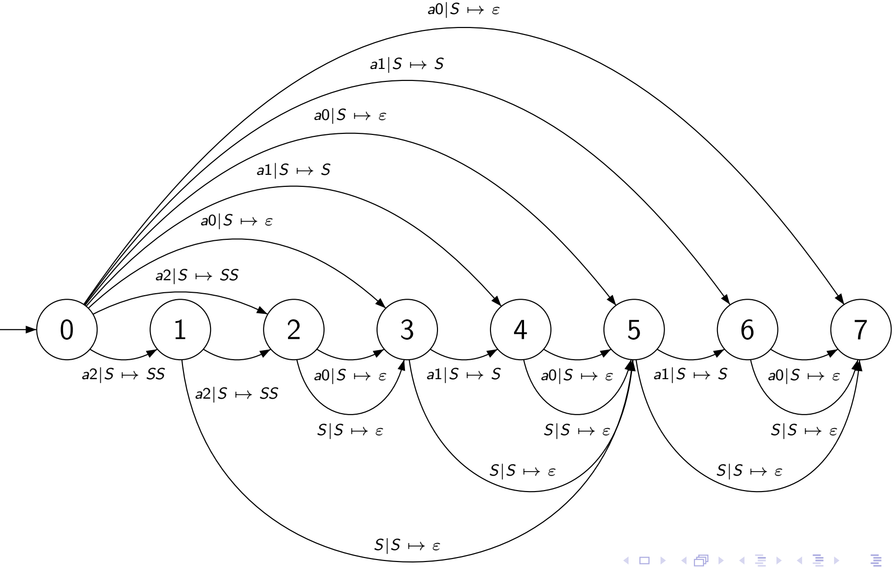
$a2\ a2\ a0\ a1\ a0\ a1\ a0$

Deterministic treetop PDA $M_{pt}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$

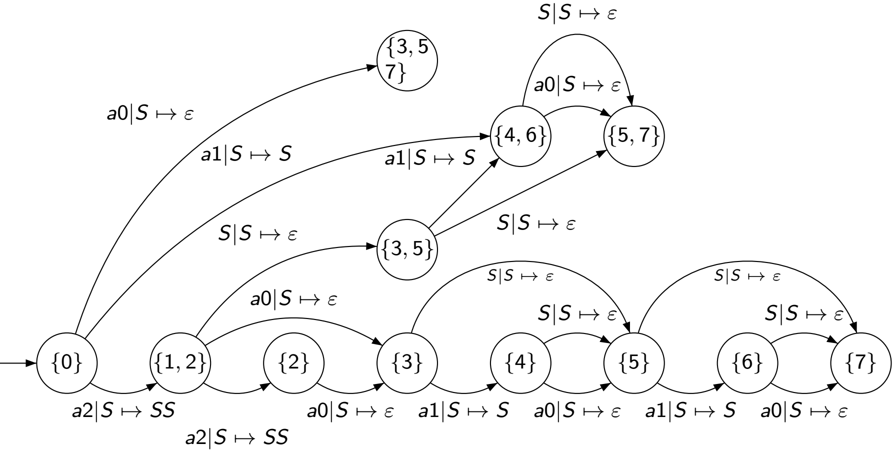
$srms = \{3, 5, 7\}$ Set of the Right-Most States



Nondeterministic tree pattern PDA $M_{npg}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ (input–driven PDA)



Deterministic tree pattern PDA $M_{dpg}(t_1)$ for $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$



Repetition table

d-subset	Subtree	List of repeats
357	$a0$	$(3, F), (5, G), (7, G)$
57	$a1\ a0$	$(5, F), (7, G)$
	$a1\ S$	$(5, F), (7, G)$
	$a2\ SS$	$(5, F), (7, O)$
	$a2\ S\ a1\ S$	$(5, F), (7, O)$
	$a2\ S\ a1\ a0$	$(5, F), (7, O)$

F first
G gap
N neighbour
O overlapping (inclusion)

Complexity

$$\mathcal{O}(n + r)$$

n	the number of nodes of the tree
r	the total size of repeating parts (subtrees, templates) of the tree (the size of repetition table)

$$r = \sum_p (rp * nr)$$

r is the total size of all pathes from the initial state to states with multiple subsets.

rp	size of repeating part
nr	number of repeats (size of d–subsets)
p	pathes

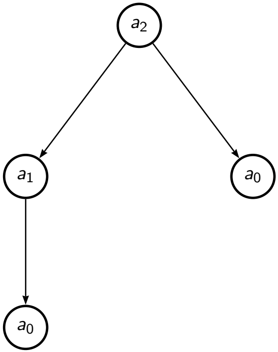
4. Tree pattern matching

Types of patterns

- EXACT PATTERNS
  $P = a_{x_1} a_{x_2} \dots a_{x_n}$
  EXAMPLE: $a_2 a_1 a_0 a_0$
- PATTERNS HAVING SUBTREES (TREE TEMPLATES)
  $P = a_{x_1} \dots a_{x_{k_1}} S^{p_1} a_{x_{k_1+1}} \dots a_{x_{k_m}} S^{p_m} a_{x_{k_m+1}} \dots a_{x_n}$
  EXAMPLE 1: $a_2 a_1 S a_0$
  EXAMPLE 2: $a_3 S S a_2 S a_0$

Exact pattern

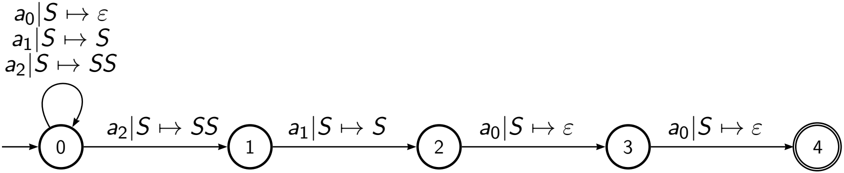
EXAMPLE:

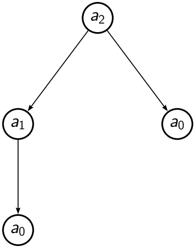


$$P = a_2 a_1 a_0 a_0$$

Exact pattern

Non–Deterministic Searching Pushdown Automaton

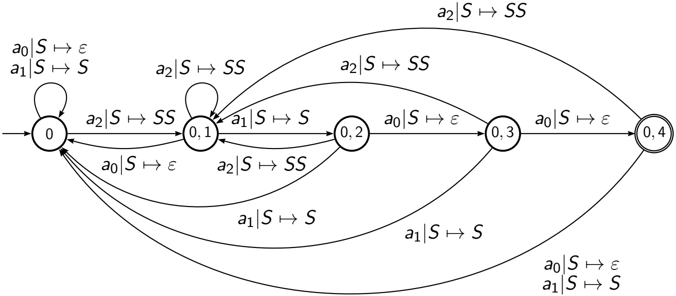


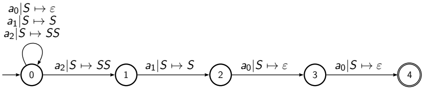


$$P = a_2 a_1 a_0 a_0$$

Exact pattern

Deterministic Searching Pushdown Automaton





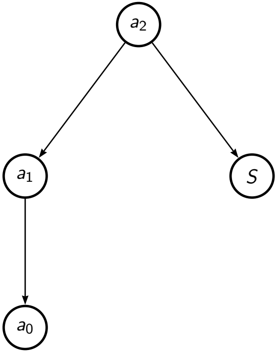
$$P = a_2 a_1 a_0 a_0$$

Tree template

Example

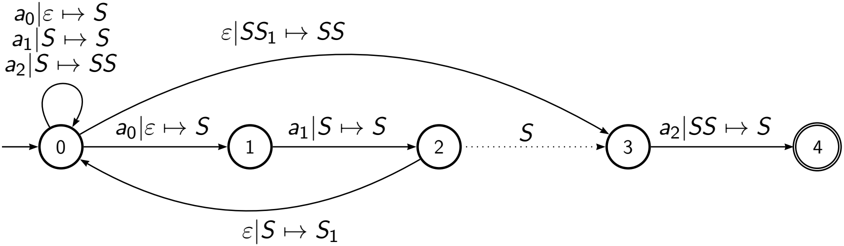
Template in postfix notation:

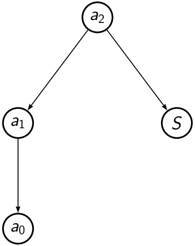
$$P = a_0 a_1 S a_2$$



Tree template

NON–DETERMINISTIC SEARCHING PUSHDOWN AUTOMATON



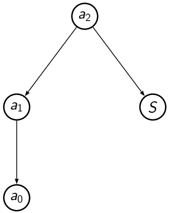


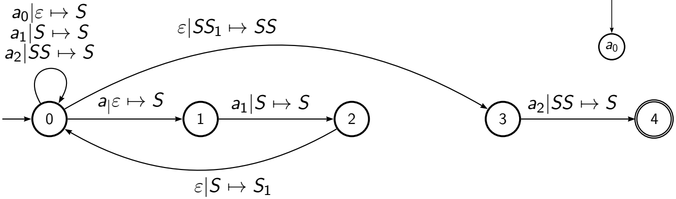
$$P = a_2 a_1 S a_0$$

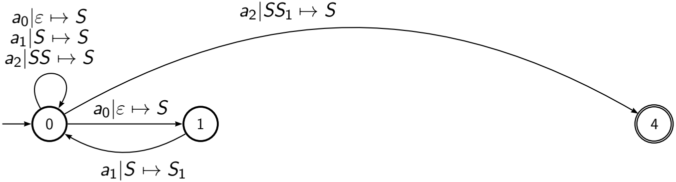
Tree template

Elimination of ε-transitions

Pattern $P = a_0 \ a_1 \ S \ a_2$

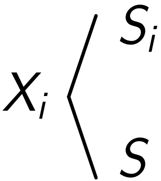






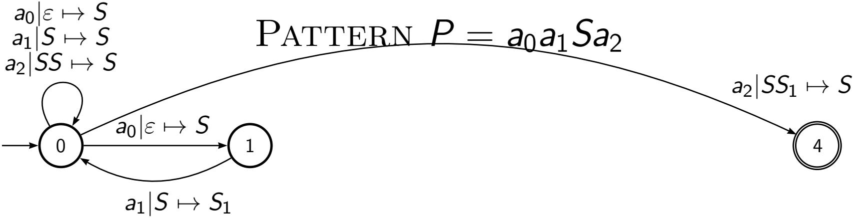
Tree template

USAGE OF NEW PUSHDOWN SYMBOLS...



...SIMULATING THE TWO CASES.

Tree template



	a_0	a_1	a_2
0	$0\|\varepsilon \mapsto S$ $1\|\varepsilon \mapsto S$	$0\|S \mapsto S$	$0\|SS \mapsto S$ $4\|SS_1 \mapsto S$
1		$0\|S \mapsto S_1$	
4			

	a_0	a_1	a_2
[0]	$[0,1]\|\varepsilon \mapsto S$	$[0]\|S \mapsto S$ $[0]\|X_1 \mapsto S$	$[0]\|SS \mapsto S$ $[0,4]\|SX_1 \mapsto S$ $[0]\|X_1S \mapsto S$ $[0,4]\|X_1X_1 \mapsto S$
[0, 1]	$[0,1]\|\varepsilon \mapsto S$	$[0]\|S \mapsto X_1$ $[0]\|X_1 \mapsto X_1$	$[0]\|SS \mapsto S$ $[0,4]\|SX_1 \mapsto S$ $[0]\|X_1S \mapsto S$ $[0,4]\|X_1X_1 \mapsto S$
[0, 4]	$[0,1]\|\varepsilon \mapsto S$	$[0]\|S \mapsto S$ $[0]\|X_1 \mapsto S$	$[0]\|SS \mapsto S$ $[0,4]\|SX_1 \mapsto S$ $[0]\|X_1S \mapsto S$ $[0,4]\|X_1X_1 \mapsto S$

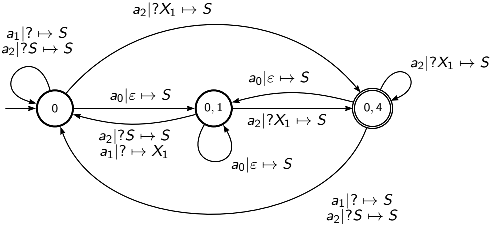
Tree template

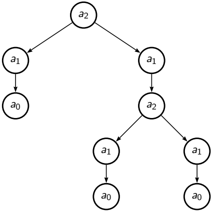
SIMPLIFICATION

	a_0	a_1	a_2
$[0]$	$[0,1]\|\varepsilon \mapsto S$	$[0]\|S \mapsto S$ $[0]\|X_1 \mapsto S$	$[0]\|SS \mapsto S$ $[0,4]\|SX_1 \mapsto S$ $[0]\|X_1S \mapsto S$ $[0,4]\|X_1X_1 \mapsto S$
$[0,1]$	$[0,1]\|\varepsilon \mapsto S$	$[0]\|S \mapsto X_1$ $[0]\|X_1 \mapsto X_1$	$[0]\|SS \mapsto S$ $[0,4]\|SX_1 \mapsto S$ $[0]\|X_1S \mapsto S$ $[0,4]\|X_1X_1 \mapsto S$
$[0,4]$	$[0,1]\|\varepsilon \mapsto S$	$[0]\|S \mapsto S$ $[0]\|X_1 \mapsto S$	$[0]\|SS \mapsto S$ $[0,4]\|SX_1 \mapsto S$ $[0]\|X_1S \mapsto S$ $[0,4]\|X_1X_1 \mapsto S$

	a_0	a_1	a_2
$[0]$	$[0,1]\|\varepsilon \mapsto S$	$[0]\|? \mapsto S$	$[0]\|?S \mapsto S$ $[0,4]\|?X_1 \mapsto S$
$[0,1]$	$[0,1]\|\varepsilon \mapsto S$	$[0]\|? \mapsto X_1$	$[0]\|?S \mapsto S$ $[0,4]\|?X_1 \mapsto S$
$[0,4]$	$[0,1]\|\varepsilon \mapsto S$	$[0]\|? \mapsto S$	$[0]\|?S \mapsto S$ $[0,4]\|?X_1 \mapsto S$

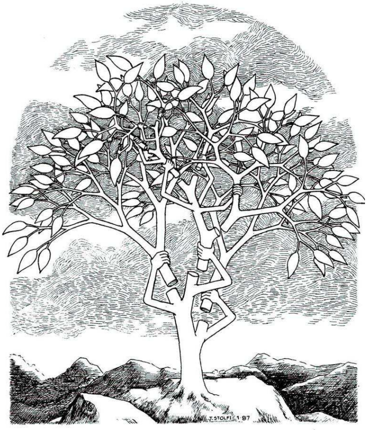
Tree template





$$P = a_0 \; a_1 \; S \; a_2$$
$$T = a_0 \; a_1 \; a_0 \; a_1 \; a_0 \; a_1 \; a_2 \; a_1 \; a_2$$

PDS	State	Node	
ε	[0]	a_0	
S	[0, 1]	a_1	
X_1	[0]	a_0	
SX_1	[0, 1]	a_1	
X_1X_1	[0]	a_0	
SX_1X_1	[0, 1]	a_1	
$X_1X_1X_1$	[0]	a_2	
SX_1	[0, 4]	a_1	match
SX_1	[0]	a_2	
S	[0, 4]		match



More information on web pages

`http://www.arbology.org`