Graphs and Automata*

Extended abstract

Bořivoj Melichar

Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University
Thakurova 9, 160 00 Prague 6, Czech Republic
melichar@fit.cvut.cz

The well known Chomsky classification concerns the classification of grammars and automata. It is sketched in the following table.

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

Algorithm processing some nonlinear structure (tree, matrix, n-dimensional array, graph, ...), contains, in some form, a statement like this:

Do traversing the structure and perform the 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.

The question which can be asked is about properties of linear notations of some type of structures. Some of properties of such linear notations can be used for design of algorithms for their processing. The following table shows types of automata suitable as models for processing linear notations of different types of graphs.

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	?

"Linear" graphs are representations of texts. The use of finite automata for this case has been described in many publications [3].

Linear notations of trees are context-free languages. Therefore, pushdown automata can serve as good models for algorithms in arbology [1,2]. Next examples show how to transfer the knowledge from stringology to arbology.

Example 1. Pattern matching

Given string $t = a2a2a0a1a0a1a0$. The nondeterministic finite automaton for matching the string t has the transition diagram depicted in Figure 1.

* This research has been supported by the Czech Science Foundation (GAČR) as project No. 13-03253S.

Bořivoj Melichar: Graphs and Automata, pp. 1–6.
Proceedings of PSC 2013, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-05330-0

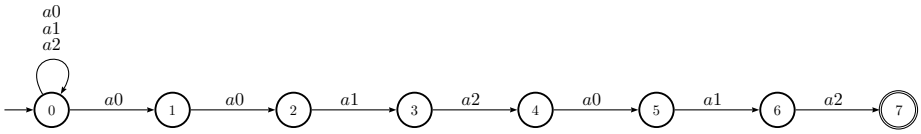
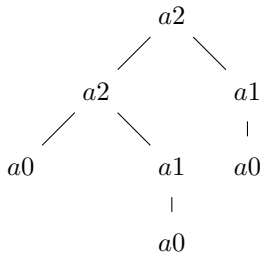


Figure 1. Transition diagram of nondeterministic finite automaton for string t from Example 1

Example 2.
String t from Example 1 is in fact the prefix notation of the tree depited in Figure 2. The transition diagram of nondeterministic subtree matching automaton is shown in Figure 3.



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

Figure 2. Tree t from Example 2 and its prefix notation

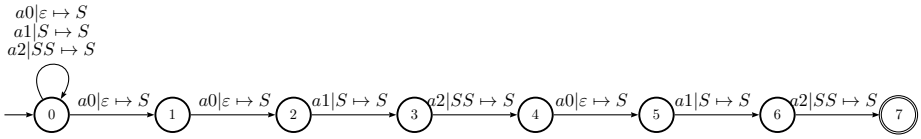


Figure 3. Transition diagram of the pushdown automaton from Example 2

Example 3.
It can be seen that shapes of transition diagrams of both automata, finite and pushdown ones, in Figures 1 and 3 are the same. They differ by pushdown operations contained in pushdown automata. The pushdown atomaton from Figure 3 is input-driven and therefore it can be determinised in the same way as finite automaton. Figure 4 shows transition diagrams of both, determistic finite and deterministic pushdown automata.

Next example shows the similarity of shapes of factor automaton and subtree pushdown automaton accepting all factors of a string and all subtrees of a tree, respectively.

Example 4. Factor and subtree automata
Figure 5 shows transition diagrams of nondeterministic factor automaton and nondeterministic subtree automaton for string $t = a2a2a0a1a0a1a0$ which is the prefix notation of tree from Figure 2.

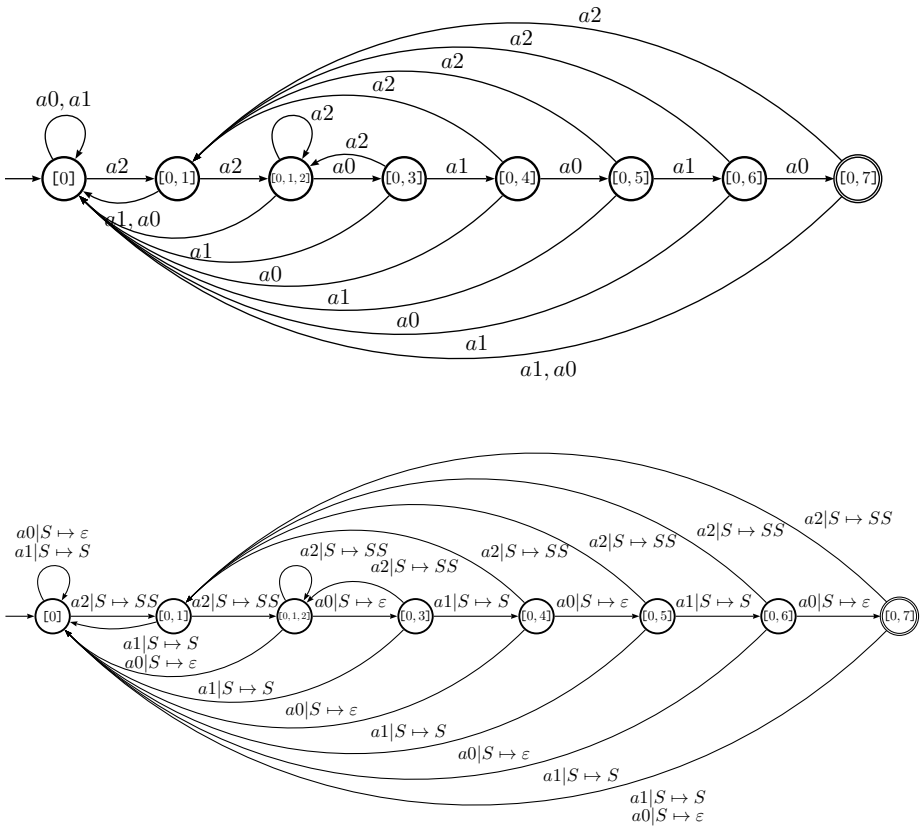


Figure 4. Transition diagrams of deterministic finite and pushdown automata from Example 2

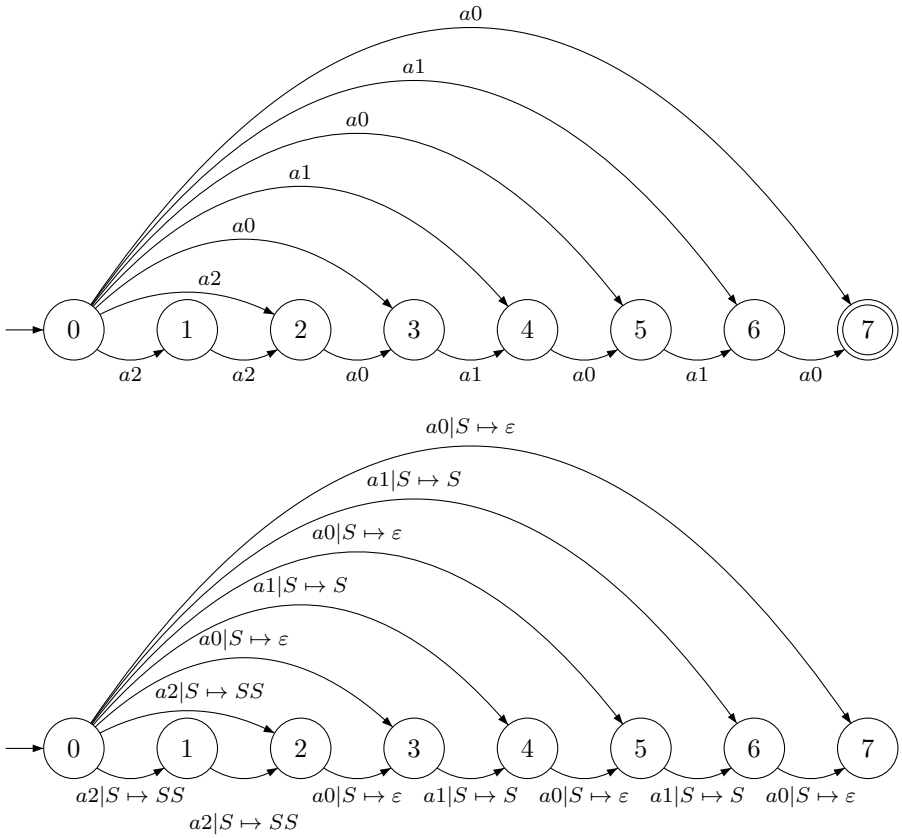


Figure 5. Transition diagram of nondeterministic factor and subtree pushdown automata for tree t in prefix notation $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 4

Transition diagrams of deterministic factor and deterministic subtree pushdown automata are depicted in Figures 6 and 7.

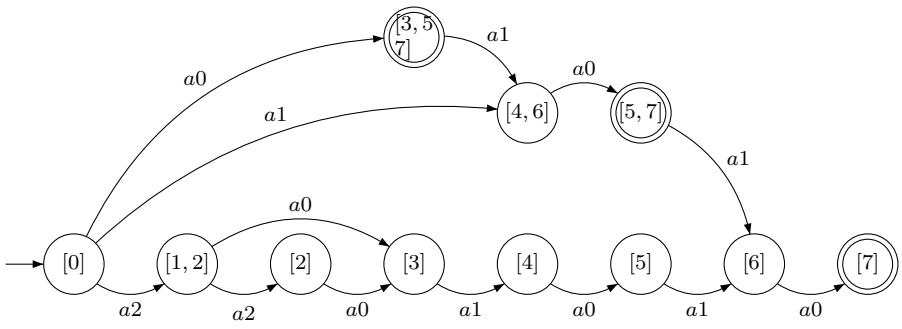


Figure 6. Transition diagram of deterministic suffix automaton for string $a2\ a2\ a0\ a1\ a0\ a1\ a0$

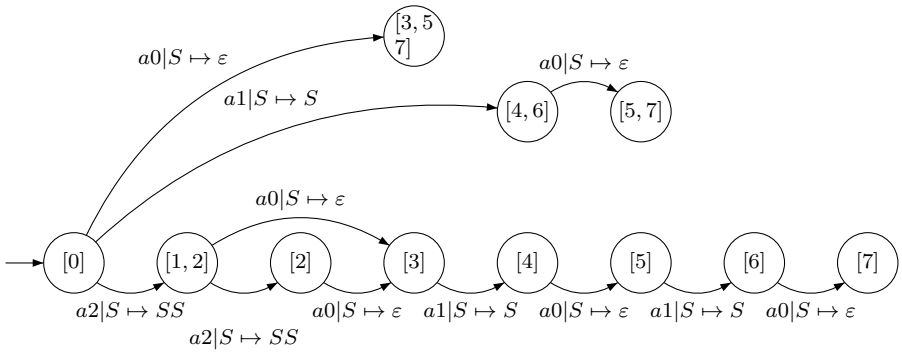


Figure 7. Transition diagram of deterministic subtree PDA $M_{dps}(t_1)$ for tree in prefix notation $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ from Example 5

There is a difference between these two deterministic automata. The factor automaton accepts all factors without any additional conditions. On the other hand, the subtree automaton accepts linear notations of subtrees only. Therefore transitions from states $[3, 5, 7]$ and $[5, 7]$ to states $[4, 6]$ and $[6]$ for input symbol $a1$, respectively, are omitted in the subtree automaton. The reason is that linear notations of subtrees are either $a0$ or $a1a0$ in these cases. If they are exteded by any symbol, they are no more linear notations of subtrees. The end of linear notation of a subtree is found using pushdown automaton by empty pushdown store.

Next example shows how to represent a matrix as an acyclic directed graph. Moreover, a linear representation of of this graph is shown which can be processed by linear bounded automaton. The principle can be easily extended to n-dimensional arrays [4].

Example 5. Representation of a matrix
A matrix is represented as a directed acyclic graph using relations to the right neighbour and lower neighbour. The next step is the construction of the spanning tree of this graph and adition of some pointers instead of missing edges.

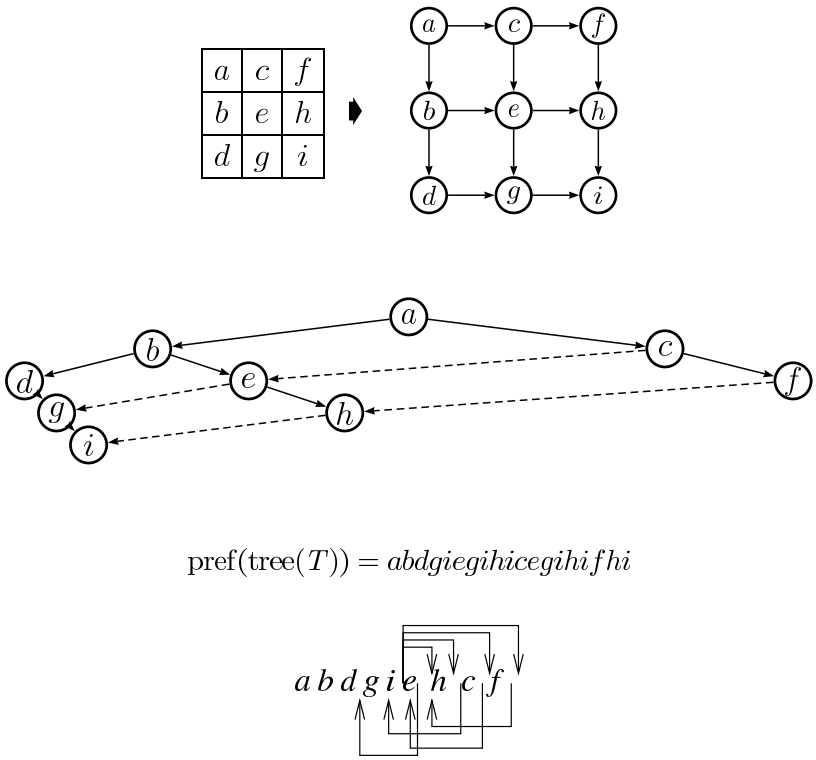


Figure 8. Representation of a matrix: A matrix is represented as a directed acyclic graph using relations to the right neighbour and lower neighbour. The next step is the construction of the spanning tree of this graph and adition of some pointers instead of missing edges

References

1. *Arbology www pages*: Available on: `http://www.arbology.org`, July 2009.
2. 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.
3. B. Melichar, J. Holub, and T. Polcar: *Text searching algorithms*. Available on: `http://www.stringology.org/athens/`, Nov. 2005.
4. 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`.