Diccionary Automata

In previous practices, it has been studied a non-deterministic approach to pattern matching. It has also been seen that the non-deterministic approach has a better time complexity bound with respect to the *naive* approach to the problem. This practice is devoted to obtain a deterministic automaton which allows us to further improve the complexity of the solution.

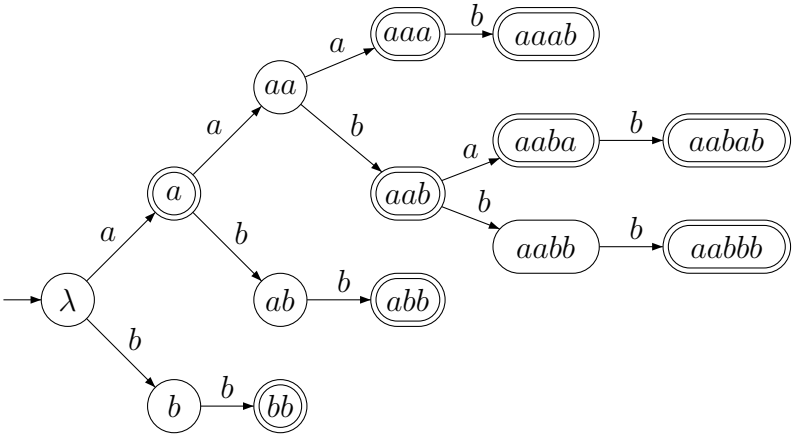
The method to study is known as the *dictionary automaton* for a given set of patterns. Thus, given any set of strings M over an alphabet Σ, the dictionary automaton $DA_M = (Q, \Sigma, \delta, q_0, F)$ is defined as follows:

- $Q = \{x \in \Sigma \ : \ x \in Pref(M)\}$
- $q_0 = \lambda$
- $F = Pref(M) \cap \Sigma^* M$
- $\delta(x, a) = h(xa)$ where, for any string u, $h(u)$ is the longest suffix of u which belongs to $Pref(M)$.

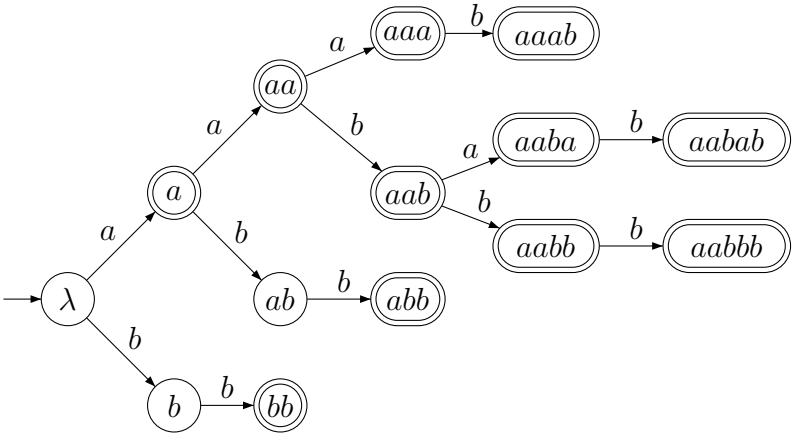
Given any set M, and taking into account the definitions of both automata, it is worth to note here the similarities among the dictionary automaton (AD_M) and the prefix tree acceptor ($AAP(M)$). The first difference refers to the definition of the final set of states. The second one is the definition of the transition function. Despite these differences, every transition in $AAP(M)$, and every final state as well, is also in AD_M. In the following, we will take into account the set of patterns of the previous practice:

$$M = \left\{ \begin{array}{l} p_1 = a,\ p_2 = bb,\ p_3 = aaa,\ p_4 = aab,\ p_5 = abb, \\ p_6 = aaab,\ p_7 = aaba,\ p_8 = aabab,\ p_9 = aabbb \end{array} \right\}$$

Note that an identifier has been added to have an easy way to refer to each one of these patterns. In order to further emphasize the simililarities among the $PTA(M)$ and the DA_M, we will compute the $PTA(M)$ as the first step to obtain the DA_M. Thus, the $PTA(M)$ is shown below:

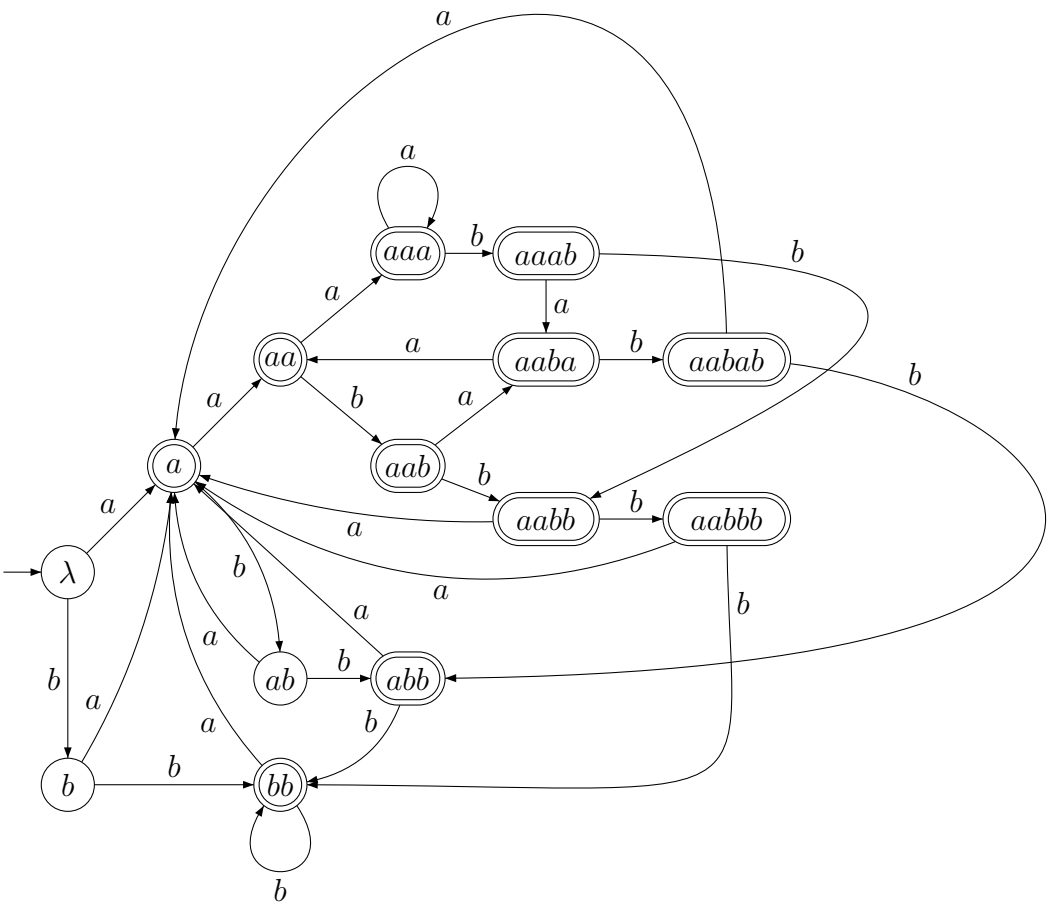


We now modify the set of final states. Please, note that the state aa, which is not final in $PTA(M)$, is a final state in DA_M (it has a suffix in M). No other state is added, thus, this first modification leads us to the following automaton:



We now consider the undefined transitions of the automaton, for instance, transitions from bb with symbols a and b, or transition from state ab with symbol a.

From the definition of the transition function for DA_M, $\delta(ab, a) = h(aba)$, which is the longest suffix of aba in $Pref(M)$, that is a. Therefore $\delta(ab, a) = a$. The result to apply this process is the DA_M which is shown below:

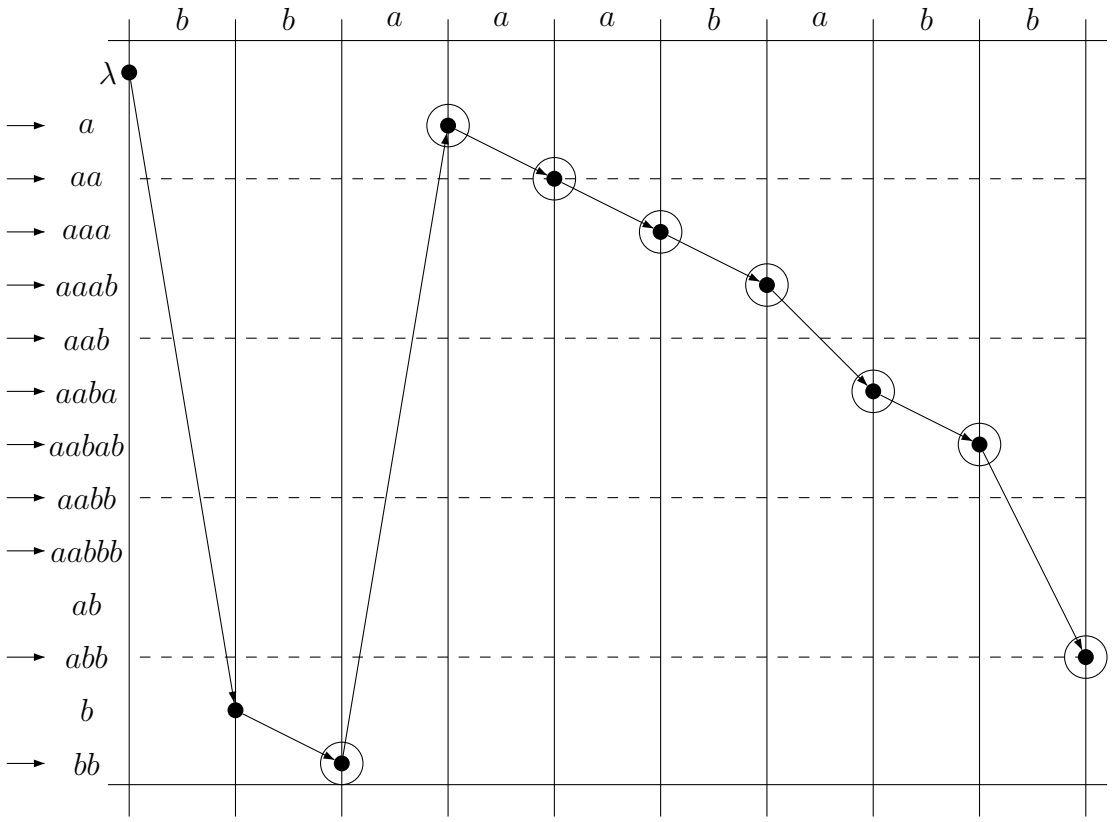


In the same way we did in the previous practice, we have obtained an automaton that recognizes the language $\Sigma^* M$. Thus, while any text is analyzed, whenever a final state is reached it can be assured that a pattern in M has been found. In fact, the patterns detected are those which are suffix of the string that denotes the final state.

To easily process the text, we modify the automaton to annotate in each final state u which patterns in M are suffix of the string u. This information is summarized in the following table:

state	a	aa	bb	aaa	aab	abb	$aaab$	$aaba$	$aabb$	$aabab$	$aabbb$
patterns	p_1	p_1	p_2	$p_1,\ p_3$	p_4	$p_2,\ p_5$	p_4, p_6	$p_1,\ p_7$	p_2, p_5	p_8	p_2, p_9

Now, it is possible to carry out a string matching process by a deterministic parsing over the automaton. Whenever a final state is found, those patterns associated to the state are found. For instance, taking into account the text $x = bbaaababb$, the deterministic parsing is represeted as a trellis below:



The final states reached during the analisys are circle-marked. Note that, once the second symbol of the text is analized the state bb is reached. This implies that the pattern p_2 has been found (pattern bb). In the same way, once the prefix $bbaaa$ has been analyzed, state aaa is reached, thus patterns a y aaa have been found.

Exercises

Exercise 1

Implement a Mathematica module that, on input of a string u and a set of strings M, returns the longest suffix of u which is in M.

Exercise 2

Implement a Mathematica module that, on input of a set of strings M, outputs the dictionary automaton for M.

Exercise 3

Implement a Mathematica module that, on input of a dictionary automaton of a set of patterns M and a text x, outputs the set of positions of x where a pattern in M can be found.

Bibliography

Maxime Crochemore, Christophe Hancart and Thierry Lecroq ALGORITHMS ON STRINGS. *Cambridge University Press*, 2007.