Introduction

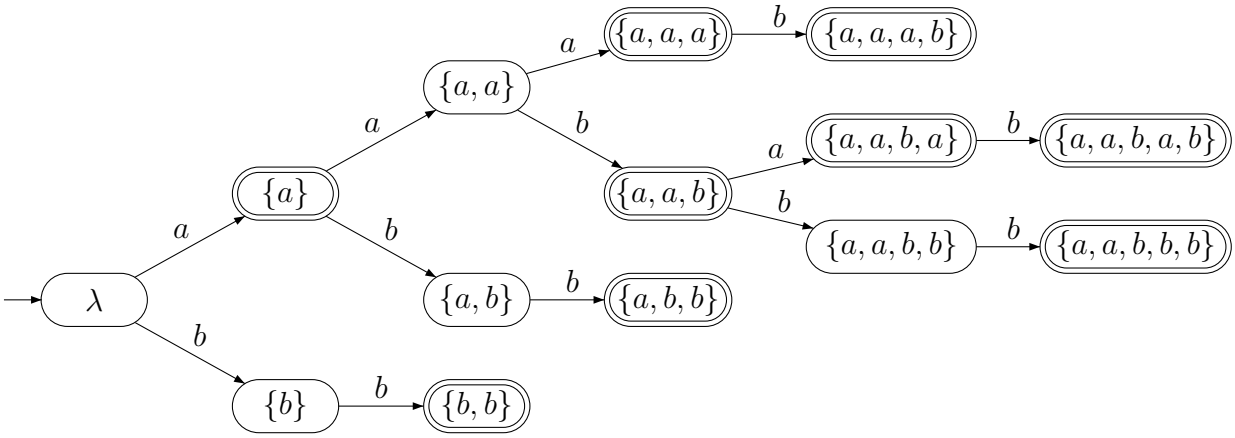
Given any text x, the task of detecting the positions of x where a set of patterns can be found is usually known as *pattern matching* or *string matching*. This problem is of algorithmic interest and plays an important role in many fields, such as Molecular Biology or Genetics, because it allows fast processing of biologic sequences.

A first *naive* approximation to the problem is to carry out an individual search of each pattern in the text. This leads to a complexity of $\mathcal{O}(n \cdot |p| \cdot |x|)$, where n is the number of patterns to locate, $|p|$ is the length of the longest pattern and $|x|$ is the length of the text. In this practice we will study how a non-deterministic approach can help to solve the problem.

To do so, for any given set M of words over some alphabet Σ, we consider the *prefix tree acceptor* for M ($PTA(M)$), which is defined as the DFA that accepts exactly the language M. For instance, taking into account the set:

$$M = \{\{a\}, \{b,b\}, \{a,a,a\}, \{a,a,b\}, \{a,b,b\}, \{a,a,a,b\}, \{a,a,b,a\}, \{a,a,b,a,b\}, \{a,a,b,b,b\}\}$$

the $PTA(M)$ would be:

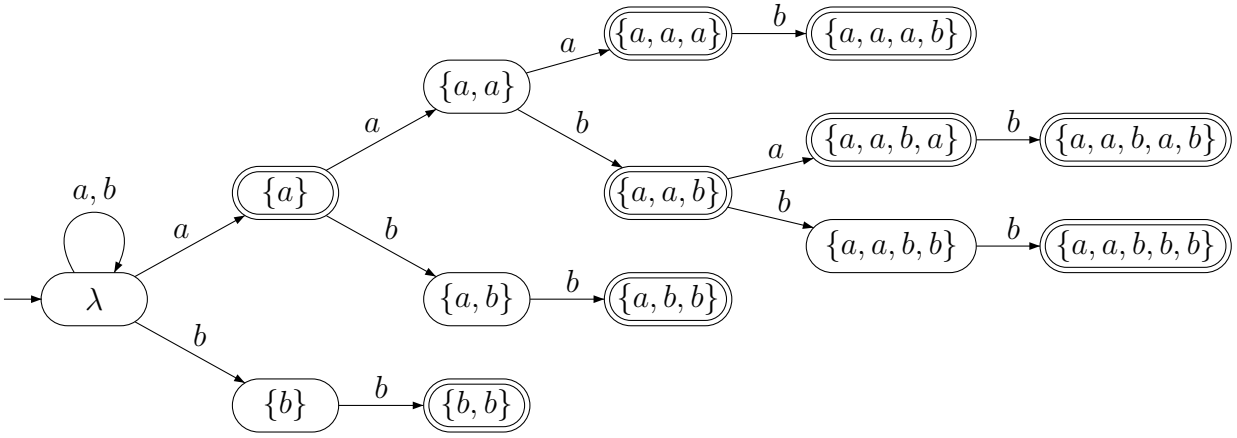


We note that it is easy to build the PTA of a set of words M if we consider the prefixes of the words in M. Intuitively, note that, in order to accept any given word, it is necesary to consider all the symbols in that word and in the correct order.

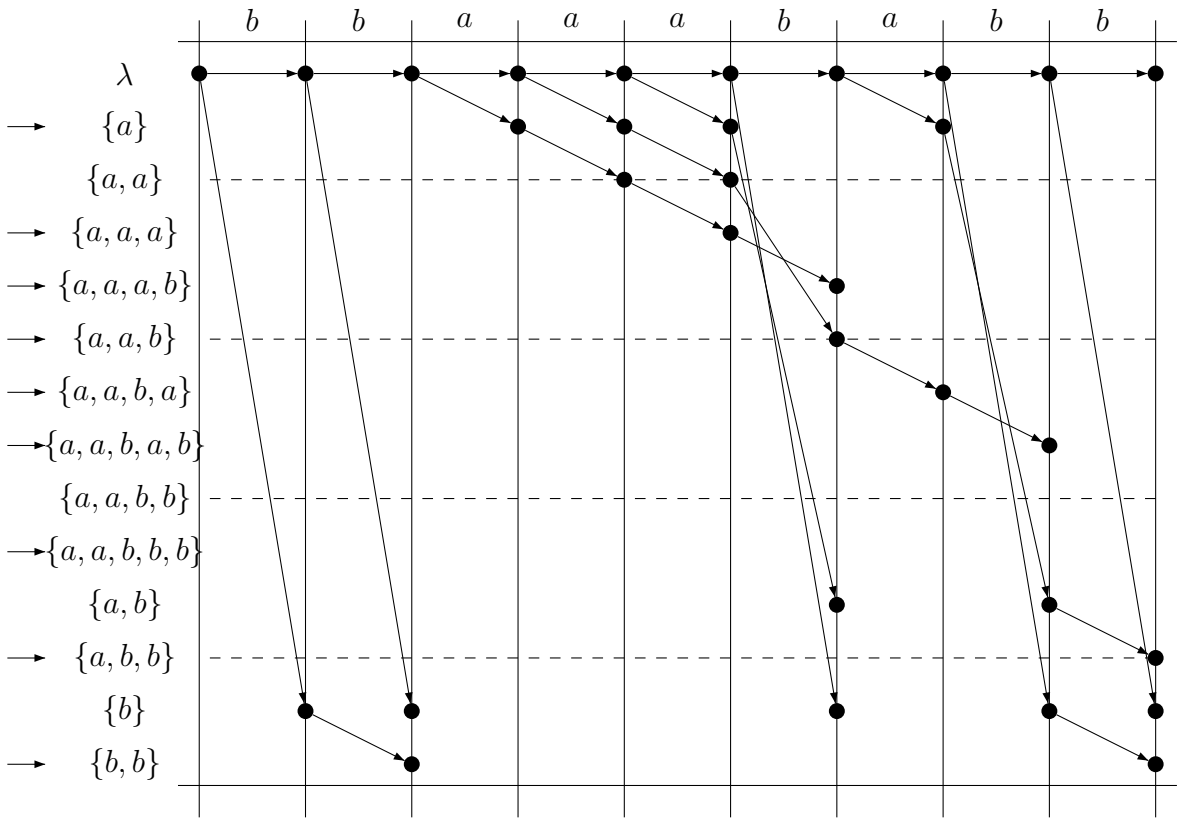
More formally, the $PTA(M)$ is defined as the automaton $A = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = \{x \in \Sigma \ : \ x \in Pref(M)\}$
- $q_0 = \lambda$
- $F = M$
- $\delta(x, a) = xa$ if $xa \in Q$, and undefined otherwise

This automaton can be easily modified to obtain a NFA to accept the language $\Sigma^* M$. To carry out such modification, note that it is enough to add a loop on the initial state with every symbol in the alphabet. Taking into account the previous automaton, the result of the operation is shown below.



This automaton is useful to locate each position of a text x where a string in a set M can be found. Note that, to do so, it is enough to carry out a, slightly modified, non-deterministic parsing of x. Whenever a final state is reached it is possible to say that a pattern has been found. For instance, let us consider the text $x = \{b, b, a, a, a, b, a, b, b\}$. The analysis is represented in the trellis below:



The diagram allows to detect, for instance, that once the second symbol has been processed, states $\{b\}$ and $\{b, b\}$ are reached. State $\{b, b\}$ is final, thus, a pattern in M has been found (pattern bb). In the same way: the analysis of bba and $bbaa$ lead to reach the state $\{a\}$ (as well as some other states), which means that pattern a has been found; and, the analysis of $bbaaa$ lead to reach the states $\{a\}$ and $\{a, a, a\}$, both finals, or in other words, the patterns a and aaa have been found.

Exercises

Exercise 1

Implement a Mathematica module that, on input of a set of strings M, returns the prefix tree acceptor of M

Exercise 2

Design a Mathematica module that, on input of a set of strings M, return a NFA that accepts the language $\Sigma^* M$

Exercise 3

Design a Mathematica module that, on input of a set of patterns M and a text x, returns the set of positions of x where each pattern in M can be found

Example: Given:

$$x = \{b, a, b, a, a, b, b, a, b, b, b, a, b, b, a, b, a, a, a, a, a, b, b, a, a, b, b, a, b, a\}$$
$$M = \{\{b, b\}, \{a, b, b, b\}, \{b, b, a, b\}, \{a, a, a, a\}\}$$

the module should return:

$$\{\{6, \{b, b\}\}, \{6, \{b, b, a, b\}\}, \{9, \{b, b\}\}, \{10, \{b, b\}\}, \{10, \{b, b, a, b\}\}, \{8, \{a, b, b, b\}\},$$
$$\{13, \{b, b\}\}, \{13, \{b, b, a, b\}\}, \{17, \{a, a, a, a\}\}, \{18, \{a, a, a, a\}\}, \{22, \{b, b\}\},$$
$$\{26, \{b, b\}\}, \{26, \{b, b, a, b\}\}\}$$

Hint: It is recommended to use and modify the exercise in Practice 2 that carries out the analysis of a word in a NFA.

Bibliography

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