Languages, Automata and Computation

REGULAR EXPERSSION

AUTHOR: JAREMEK KATARZYNA

07.11.2021
WARSAW UNIVERSITY OF TECHNOLOGY
EiTI Faculty

1 FORMULATION OF THE PROBLEM

Regular expressions are used to describe languages which can help with finding words that fits the pattern. In my assignment I am going to examine following regular expression:

$$a \cdot (ab + a)^* \cdot ab$$

The task will cover:

- analysing language of the RE and providing examples of accepting strings
- using Tompson construction algorithm to build ϵ −NFA
- subset algorithm – transforming ϵ −NFA into DFA
- construction a minimal state DFA

2 LANGUAGE OF THE REGULAR EXPRESSION

It will be convenient to divide whole expression into three parts

$$a \cdot \qquad (ab + a)^* \qquad \cdot\, ab$$

We can see two operations of *concatenation* and power set of *or*. It may be concluded that each word described by this RE starts with 'a' and ends with 'ab'. Then the middle part consists of any number of combinations of "ab" or "a". The middle part may be empty string.

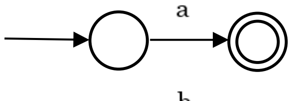
Thus, the shortest accepted string will be "aab". The regular expression accepts also:

- "aabab"
- "aaab"
- "aaaab"
- "aabaab"

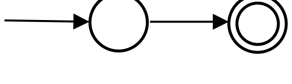
3 CONSTRUCTION OF NFA

The process of the construction is going to be done step by step starting from transitions for each element of the alphabet.

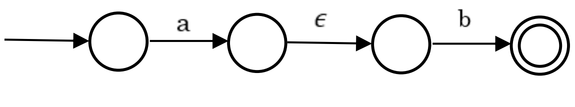
- a



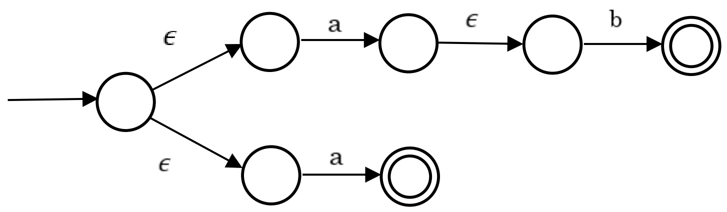
- b



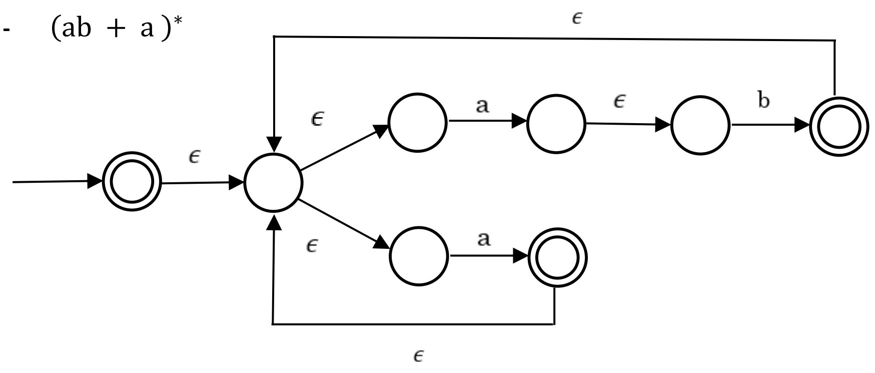
- ab



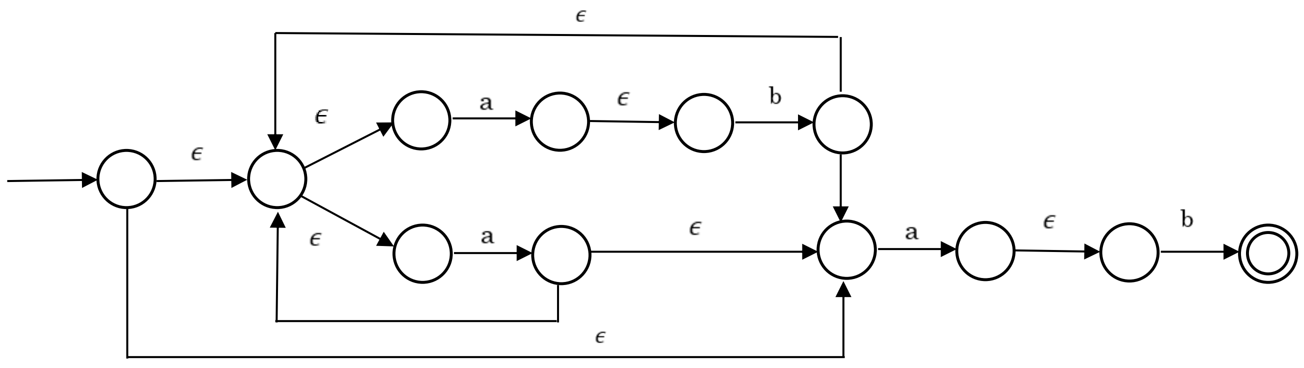
- $ab + a$



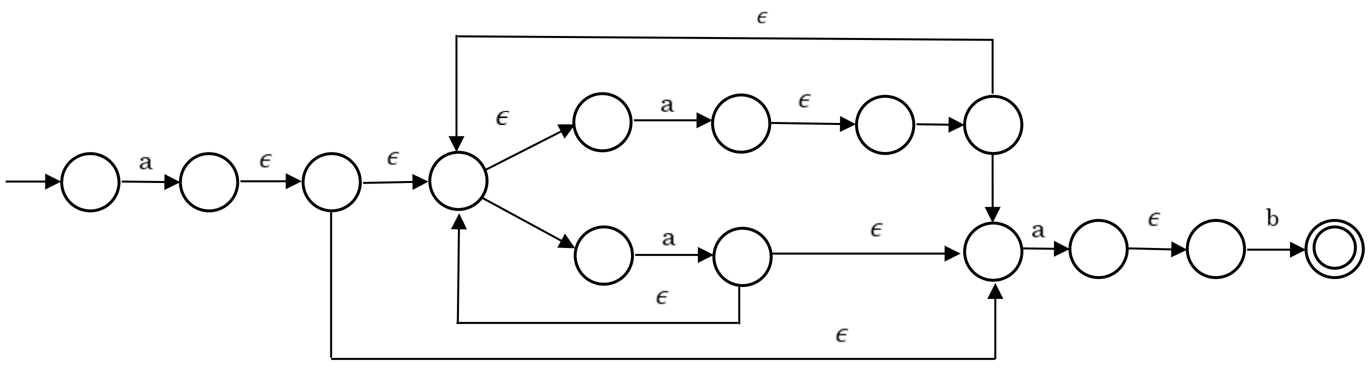
- $(\mathrm{ab} + \mathrm{a})^*$



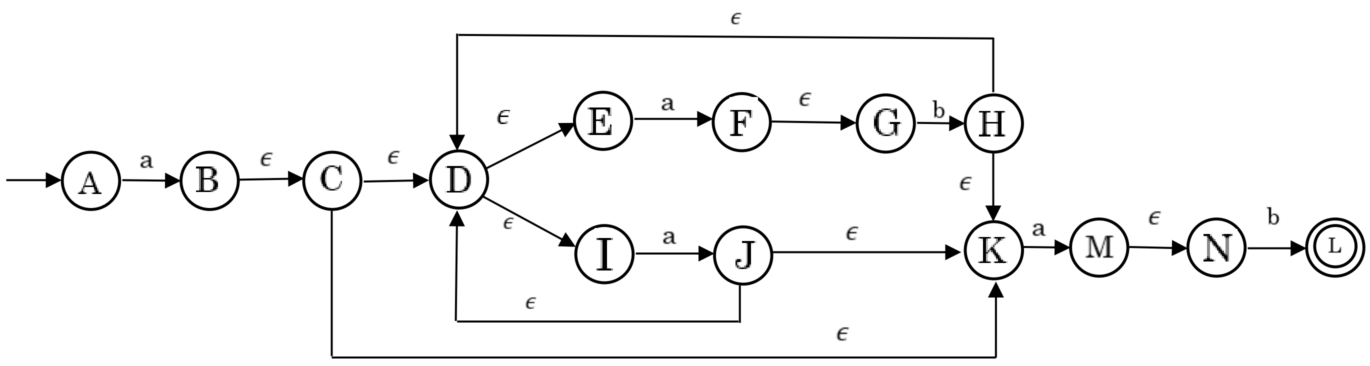
- $(\mathrm{ab} + \mathrm{a})^* \cdot ab$



- $a(\mathrm{ab} + \mathrm{a})^* \cdot ab$



So, the final ϵ −NFA with numbered states is presented below:



And it has following transition table:

State	**a**	**b**	**ϵ**
A	B	∅	∅
B	∅	∅	C
C	∅	∅	D, K
D	∅	∅	E, I
E	F	∅	∅
F	∅	∅	G
G	∅	H	∅
H	∅	∅	D, K
I	J	∅	∅
J	∅	∅	D, K
K	M	∅	∅
M	∅	∅	N
N	∅	L	∅
L	∅	∅	∅

4 SUBSET ALGORITHM – DFA

The DFA may be obtained from transformation of a transition table of ϵ −NFA. Firstly, we consider the starting state and all arcs coming from this state. However, in the DFA we do not have transitions on *epsilon*. To remove these transitions, we calculate the *closure of q*, where *q* is state of the NFA**.**

closure(q) CL (q) = set of states that can be reached from q following transitions on $\boldsymbol{\epsilon}$

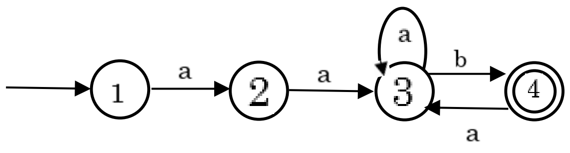
For example, CL(B) = CL(C) = CL(D, K) = {E, I, K} and after this we can check results of transitions on *a* and *b*.

A final state of DFA is the one that contains at least one final state of the NFA.

Transitions of all reachable states are considered.

New State	*State*	a	b	CL(STATE)
➔ *1*	*A*	B	∅	∅
2	*B*	F, J, M	∅	F, J, M
3	F, J, M	F, J, M	H, L	H, F, J, M, L
**4*	H, L	F, J, M	∅	F, J, M

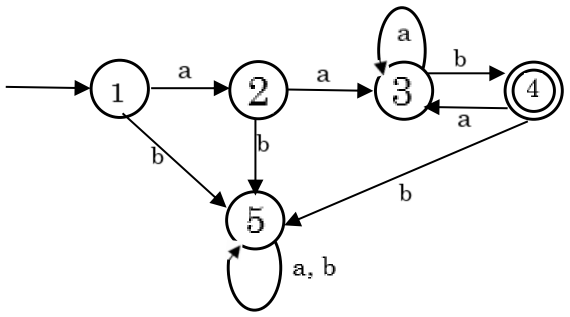
Below is presented the DFA which correspond to the given regular expression with its transition table:



State	*a*	*b*
➔ *1*	2	∅
2	3	∅
3	3	4
**4*	3	∅

5 MINIMAL STATE DFA

To perform minimization given DFA need to be complete, which means that there must be specified transition from each state on every input. So, there will be added state 5, which can be treated as a reject state. If the automata go to the state 5, it remains there, and input is not accepted. The DFA presented below is complete version:



Now the minimalization named the table filing based on Myhill-Nerode’s theorem can be performed.

Firstly, we draw a table of all pairs of states (P, Q) and mark all pairs where P is final, and Q is not.

1				
	2			
		3		
X	X	X	**4**	
			X	**5**

Then if there are any unmarked pairs such that [δ(P, x), δ(Q, x)] is marked then marked (P, Q). This step is repeated until no more pairs can be marked.

- (1, 2)
  [δ(1, a), δ(2, a)] = (2,3) -unmarked, [δ(1, b), δ(2, b)] = (5, 5) unmarked
- (2, 3)
  [δ(2, a), δ(3, a)] = (3,3) -unmarked, [δ(2, b), δ(3, b)] = (5, 4) marked so (2,3) is also marked
- (1, 3)
  [δ(1, a), δ(3, a)] = (2,3) - marked so (1,3) is also marked, [δ(1, b), δ(3, b)] = (5, 4) marked
- (1, 5)
  [δ(1, a), δ(5, a)] = (2,5) - unmarked, [δ(1, b), δ(5, b)] = (5, 5) unmarked
- (2, 5)
  [δ(2, a), δ(5, a)] = (3,5) - unmarked, [δ(2, b), δ(5, b)] = (5, 5) unmarked
- (3, 5)
  [δ(3, a), δ(5, a)] = (3,5) - unmarked, [δ(3, b), δ(5, b)] = (4, 5) marked so (3,5) is also marked

All new marked states can be put into table:

1				
	2			
X	X	**3**		
X	X	X	**4**	
		X	X	**5**

After next two iterations the table is as follows:

1				
X	**2**			
X	X	**3**		
X	X	X	**4**	
X	X	X	X	**5**

The next step of algorithm is to combine all unmarked states into one. Since all the states are marked, we may stay with the initial incomplete DFA since the algorithm prove that it is minimal.