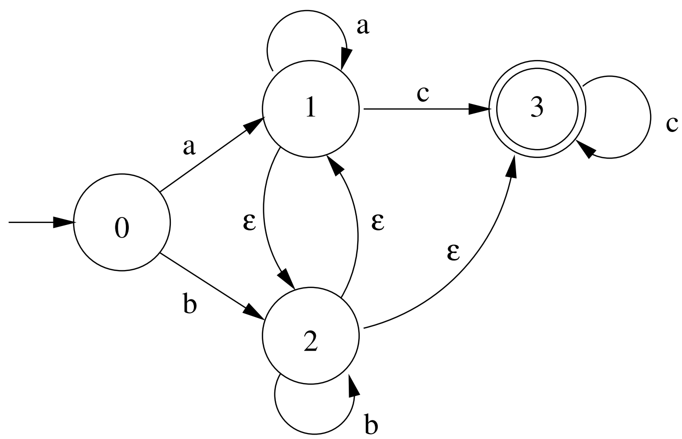
Formal Languages and Compilers
Exercises on Lexical Analysis I with Solutions

MSc in Computer Science, University of Camerino
prof. Luca Tesei

Note Regular expressions are written with the usual precedence order: operator * has precedence on concatenation, which has precedence on $|$. Moreover, the usual shorthands $^+$ and ? may be used.

Exercise 1

Write a regular expression denoting the language accepted by the following automaton:



Solution

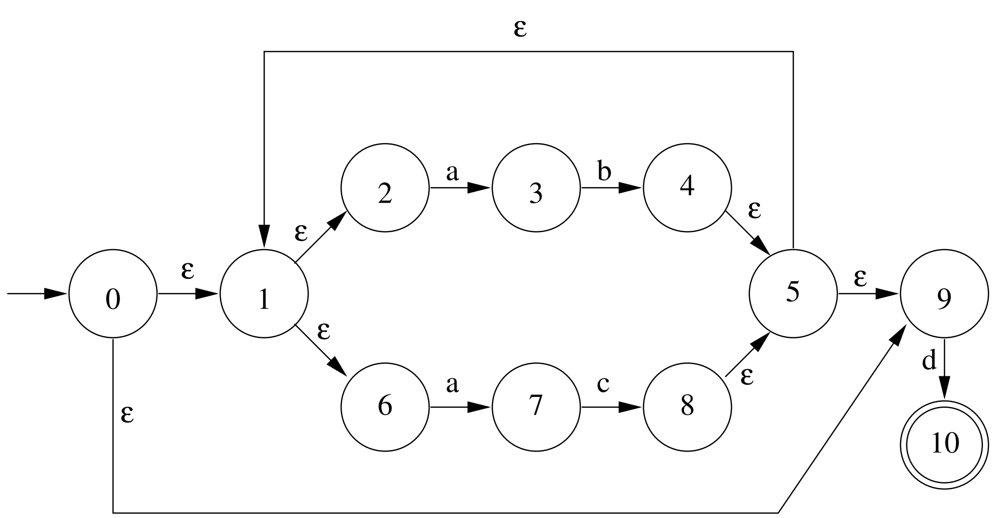
The expression is $(a|b)^+c^*$.

Exercise 2

Use Thompson algorithm to construct an NFA accepting the language denoted by $(ab|ac)^*d$.

Solution

The syntax tree of the regexp is a concatenation between a star and a d, then the star is of a union between two concatenations. Following the inductive definitions of the Thomposon algorithm the following NFA is obtained:

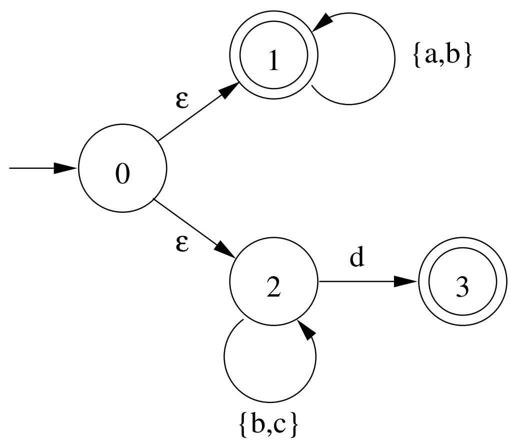


Exercise 3

Write a minimal automaton for the language $(a|b)^* \mid (b|c)^*d$.

Solution

Let us first use non-determinism to easily define an NFA for the language:



Now we can use the subset construction algorithm to find an equivalent DFA. The *move* table of the obtained DFA is the following one where accepting states are $\{A, B, C, E\}$:

State	a	b	c	d
$A = \{0, 1, 2\}$	B	C	D	E
$B = \{1\}$	B	B		
$C = \{1, 2\}$	B	C	D	E
$D = \{2\}$		D	D	E
$E = \{3\}$				

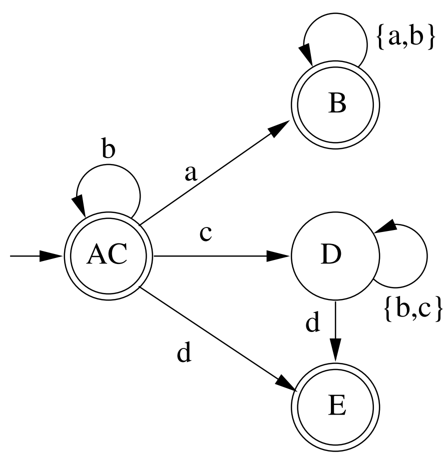
It is quite clear from the table that state A and state C are equivalent, while the rest of the states behave differently. However, for the sake of completeness, let us apply the minimisation algorithm.

First, let us complete the DFA by adding a dead state F to which we create a transition for every empty entry in the table.

The first partition to consider is $(ABCE), (DF)$. Consider the group (DF); we have that $move(D, d) = E$ and $move(F, d) = F$. We conclude that the two states are not equivalent because the input d sends the two states in different groups. Thus, the new partition to consider is $(ABCE), (D), (F)$.

The only group that can be refined is $(ABCE)$. We have $move(A, d) = E$, $move(B, d) = F$, $move(C, d) = E$, $move(E, d) = F$. Thus, the new partition is $(AC), (BE), (D), (F)$.

We have already observed that there are no differences between A and C. Let us then consider B and E. We have that $move(B, a) = B$ and $move(E, a) = F$. Thus they must be distinguished. We obtain the following automaton, which is minimal for the language and in which the dead state F is not represented:



Exercise 3

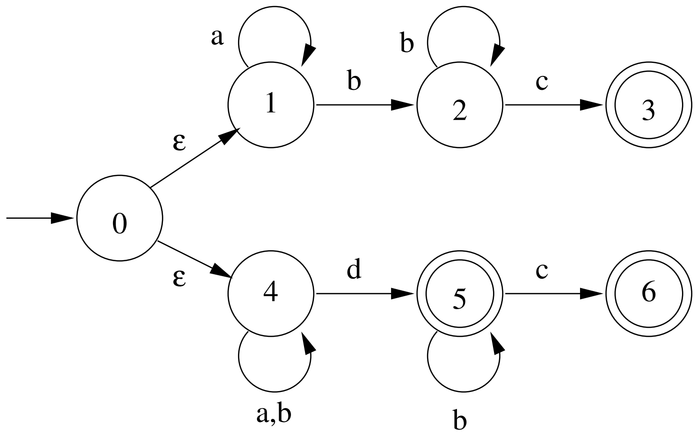
Define a **deterministic** automaton that accepts the following language:

$$a^*b^+c \mid (a|b)^*db^*(c|\epsilon)$$

Illustrate all the steps to reach the proposed solution.

Solution

We can start from a non-deterministic automaton directly obtained from the regular expression using a simplified version of the automaton that would be generated by the Thompson algorithm:



Let's apply the subset construction algorithm to get an equivalent deterministic automaton. The resulting *move* table is the following:

State	a	b	c	d
$A = \{0, 1, 4\}$	B	C		D
$B = \{1, 4\}$	B	C		D
$C = \{2, 4\}$	E	C	F	D
$D = \{5\}$		D	G	
$E = \{4\}$	E	E		D
$F = \{3\}$				
$G = \{6\}$				

The final states are F, D and G. Notice that states A and B are equivalent because they are both non-final and they behave in the same way. Thus, they can be identified.