Assignment 2: Theoretic exercises

Florian Thuin Cyril de Vogelaere

22 mars 2016

Table des matières

1 Theoretic exercices

1.1 Lexical analysis

Here is a regExp :

$$((a^*b) \mid (ab)^*)c$$

With this regExp, we can define an infinite number of sequences :

$$\{\epsilon, c, bc, abc, aabc, ababc, \ldots\}$$

1.1.1 NFA with Thompson construction

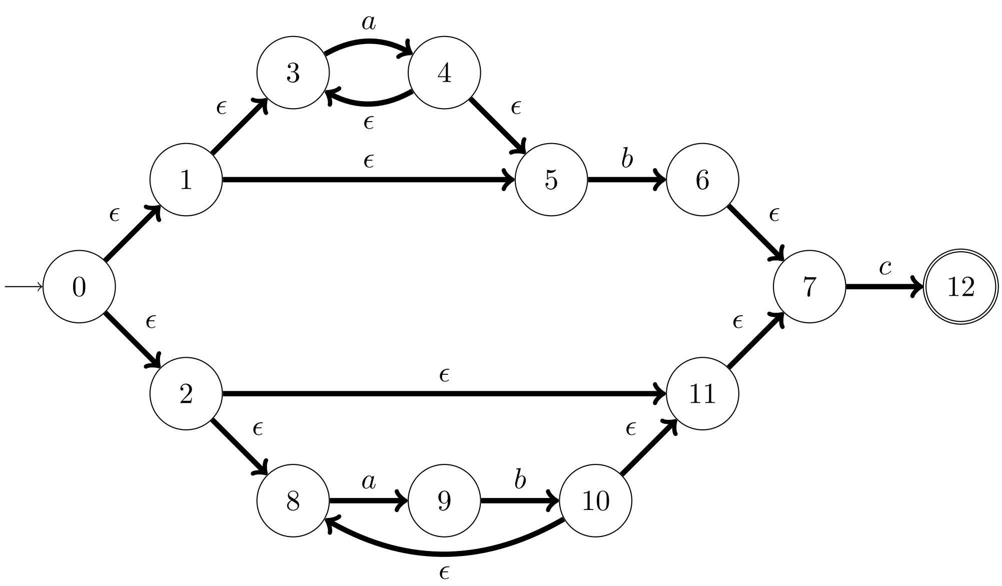

FIGURE 1 – NFA of $((a^*b)|(ab)^*)c$

1.2 NFA to DFA step-by-step

First we regroup all the states reachable from the initial only by using ϵ-closure (i.e. reachable only with ϵ-transitions).

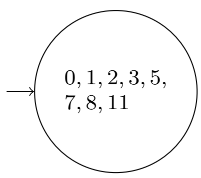

FIGURE 2 – NFA to DFA : Initial state

Then, we add which states are reachable from this initial state with an a-transition (including ϵ-transition) :

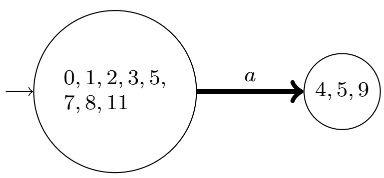

FIGURE 3 – NFA to DFA : Initial state and an a-transition

Then, we add which states are reachable from those states with a b-transition :

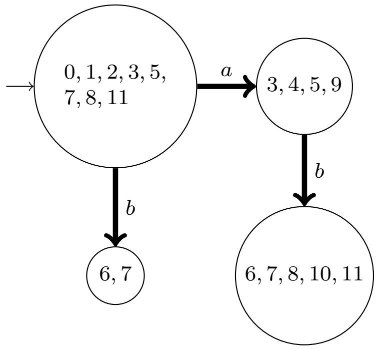

Figure 4 – NFA to DFA : Initial state and an a-transition and a b-transition

Then, we add which states are reachable from those states with an a-transition :

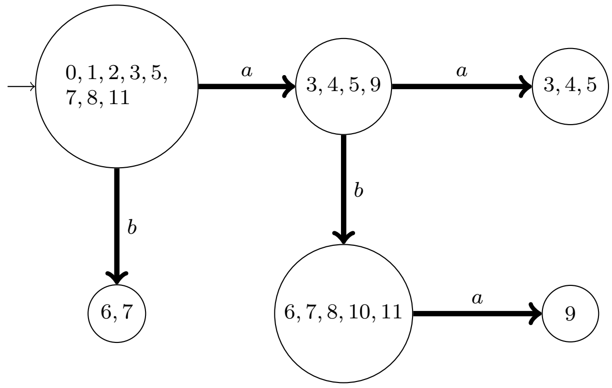

Figure 5 – NFA to DFA : Initial state and two a-transitions and a b-transition

We skip some steps by adding the remaining a- and b-transitions :

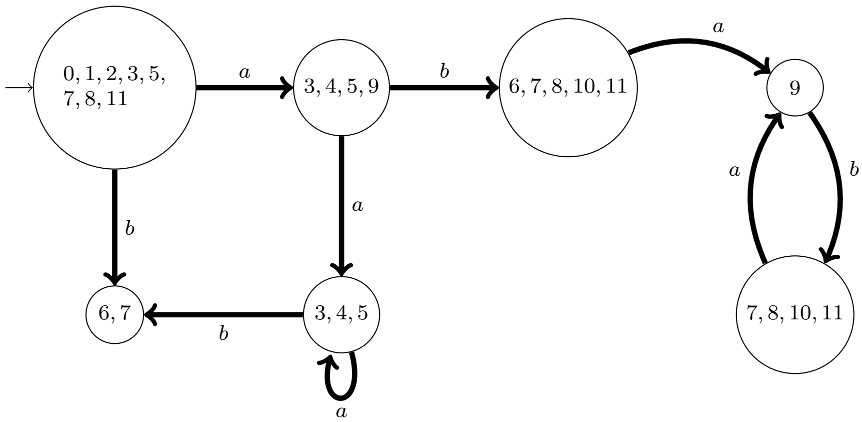

Figure 6 – NFA to DFA : Initial state and two a-transitions and a b-transition

We add the c-transitions :

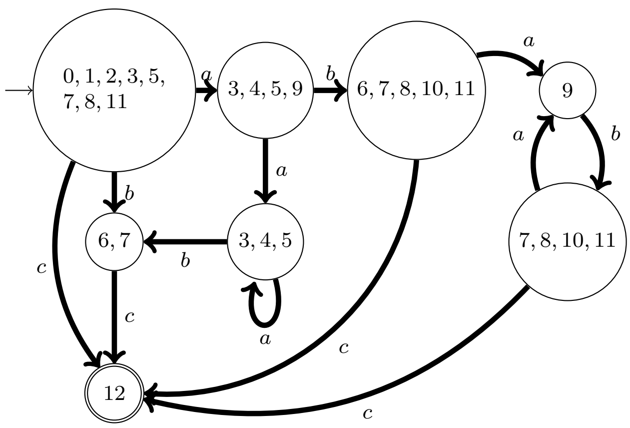

Figure 7 – NFA to DFA : Initial state and two a-transitions and a b-transition

We rename the sets of states to number to be more readable :

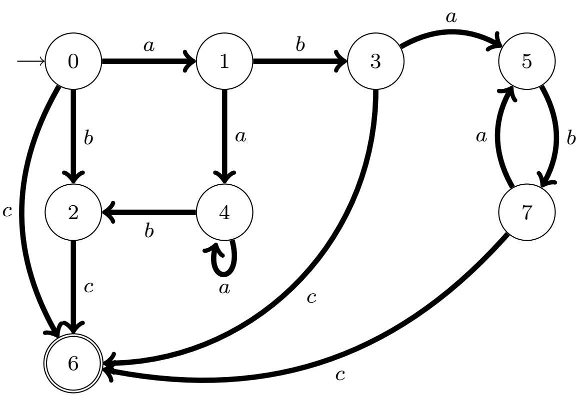

FIGURE 8 – NFA to DFA : Initial state and two a-transitions and a b-transition

Here is the minimized DFA :

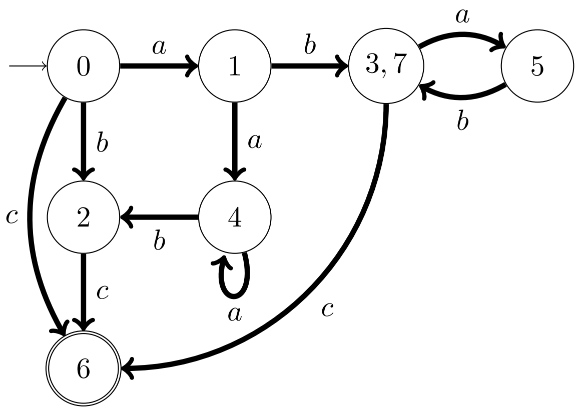

FIGURE 9 – Minimized DFA

1.3 Parsing

This grammar is not LL(1) as it is left recursive and as it has a first set conflict. An equivalent LL(1) grammar would be :

1. Y : := move(X Y1
2. Y1 : :=)
3. Y1 : := ,P)
4. X : := id
5. P : := D P1
6. P1 : := $\rightarrow$ D P1
7. P1 : := ϵ
8. D : := left
9. D : := right
10. D : := forward
11. D : := backward

To build the table, we must then identify the first and follow set of our axioms. Thus obtaining the two following set :

First set :

- first(Y) = { move(}
- first(Y1) = {) ; virgule }
- first(X) = { id }
- first(P) = { left ; right ; forward ; backward }
- first(P1) = { $\rightarrow$ }
- first(D) = { left ; right ; forward ; backward }

Follow set :

- first(Y) = { $\emptyset$ }
- first(Y1) = { $\emptyset$ }
- first(X) = {) ; virgule }
- first(P) = {) }
- first(P1) = {) }
- first(D) = { $\rightarrow$;) }

Which allow us to build the following LL(1) parsing table :

		move(	,	)	id	$\rightarrow$	left	right	forward	backward
Y		1								
Y1			3	2						
X					4					
P							5	5	5	5
P1				7		6				
D							8	9	10	11

Parsing move(id, left $\rightarrow$ right) would result in the following steps :

1. Y
2. move(X Y1
3. move(id Y1
4. move(id , P)
5. move(id , D P1)
6. move(id , left P1)
7. move(id , left $\rightarrow$ D P1)
8. move(id , left $\rightarrow$ right P1)
9. move(id , left $\rightarrow$ right ϵ)
10. move(id , left $\rightarrow$ right)

1.4 DFA

Yes, it is possible to accept an infinite language, for example ((1)* 0) accept an infinite sequence of bit but can be represented by the following DFA :

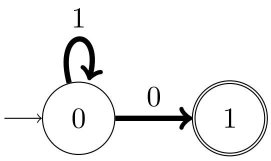

Figure 10 – DFA accepting an infinite language

1.5 Language

The language of balanced parenthesis is not regular. To demonstrate it, let us consider a simplified version of the language where we consider only a sequence of k left parenthesis followed by a sequence of k right parenthesis, this simplified version forming a subset of the language we wish to prove irregular :

$$\{(^k)^k\}$$

Using pumping lemmas, we can easily prove this language irregular by defining $w = \{(^k)^k\}$. By the pumping lemma, there should be some decomposition w = xyz with $|xy| \leq p$ and $|y| \geq 1$ such that $x(y^i)z$ in L for every $i \geq 0$.

Since $|xy| \leq p$, we know that y can only contain a non-null sequence of left parenthesis. This means that by pumping y and obtaining xy^2z, we will a sequence of parenthesis with more open parenthesis than closed parenthesis.

Thus, this sequence will never be part of our simplified language L, and since this subset of the original language cannot be represent by a regular expression, we can infer that the whole language similarly cannot be represented.