Cursus: M1, computer-science
Code UE: JEIN8602
Solutions to the subject: Formal languages theory
Date: 05 July 2013
Duration: 3H
Documents: authorized
Lectures by: Mr Géraud Sénizergues

Exercice 1 [/4] 1-

$$u_1 := c, \;\; u_2 := ba, \;\; u_3 := c, \;\; u_4 := c$$

are fulfilling the required membership assertions. Note also that the first line of the table of question 2 was based on this choice for u_1, u_3, u_4
2-

u	L_{e_1}	L_{e_2}	L_{e_3}	L_{e_4}
c	yes	no	yes	yes
abc	yes	no	yes	yes
ba	no	yes	yes	yes
$abba$	no	no	no	yes
ε	no	yes	yes	yes

(The first row is given as an example; no precise justification is required for this question).
3- $L_{e_3} = L_{e_4}$?:no, the word $abba$ belongs to L_{e_4} but does not belong to L_{e_3}.
$L_{e_2} \subseteq L_{e_1}$?:no, because $ba \in L_{e_2}$ but $ba \notin L_{e_1}$.
$L_{e_2} \subseteq L_{e_3}$?:yes
$L_{e_3} = L_{e_1} \cup L_{e_2}$?:yes

Exercice 2 [/4]

$$e := (ab^*c)^*ab(a \cup b)^*$$

Let us apply Glushkov's method.
The locally testable language associated to e is:

$$L' := (a_1b_1^*c_1)^*a_2b_2(a_3 \cup b_3)^*$$

We compute

$$Ini(L') = \{a_1, a_2\}, \;\; Fin(L') = \{b_2, a_3, b_3\},$$

$$Dig(L') = \{a_1b_1, b_1b_1, b_1c_1, a_1c_1, c_1a_1, c_1a_2, a_2b_2, b_2a_3, b_2b_3, a_3a_3, a_3b_3, b_3b_3, b_3a_3\}.$$

This gives the finite automaton represented on figure 1, where the set of states is $\{0, a_1, a_2, a_3, b_1, b_2, b_3, c_1\}$, 0 is the unique initial state and b_2, a_3, b_3 are the final states.

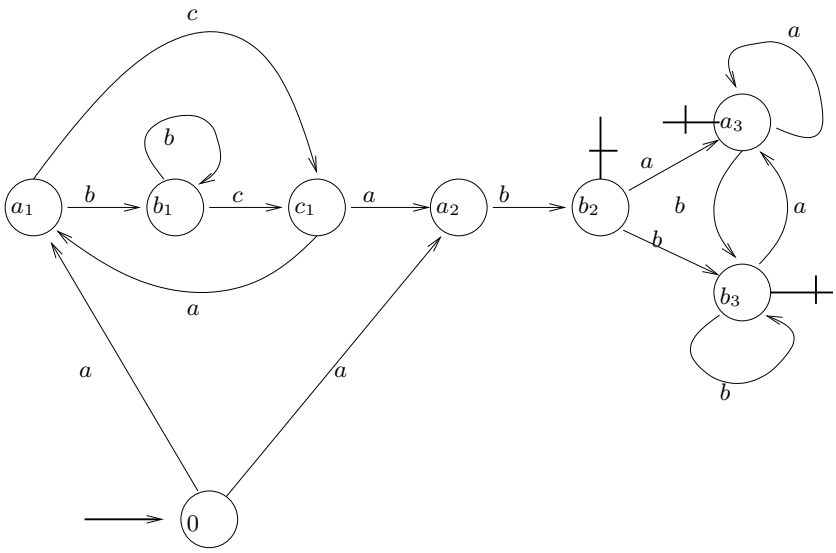

Figure 1: finite automaton for L_e

Exercice 3 [/5] We transform $\mathcal{A}$ into a normalized extended f.a. $\mathcal{A}_1$ where i (resp. t) is the initial (resp. terminal) state. We then eliminate successively states in the ordering: $3, 1, 0, 2$. We obtain the sequence of extended automata shown on figure 2. It follows that:

$$(ab^*c)(b \cup ab^*a)[c \cup ac^*bb^*a \cup ac^*bb^*c(ab^*c)^*(b \cup ab^*a)]^*ac^*$$

is a regular expression for $L_{\mathcal{A}}$.

Exercice 4 [/5] We consider the finite automaton $\mathcal{B}$.

1- It is deterministic.

2- Every state of $\mathcal{B}$ is accessible i.e. is reachable from the initial state.

3- It is also complete.

Computing the Nerode equivalence over the set of states by Moore's algorithm, we obtain the successive partitions:

$$\begin{aligned}
\equiv_0 &= \{\{1,2,3,4,6,7\},\{0,5\}\} \\
\equiv_1 &= \{\{1,3,6,7\},\{2,4\},\{0,5\}\} \\
\equiv_2 &= \{\{1,3\},\{6,7\},\{2,4\},\{0,5\}\} \\
\equiv_3 &= \{\{1,3\},\{6,7\},\{2,4\},\{0,5\}\}
\end{aligned}$$

Thus $\equiv_2$ is the Nerode equivalence over the set of states of $\mathcal{B}$. The minimal complete deterministic finite automaton $\mathcal{C}$ that recognizes the same language is obtained by identifying the states which are equivalent (w.r.t. $\equiv$). It is described on figure 3.

Exercice 5 [/6] 1- For every $i \in [1,5]$ we define a context-free grammar $G_i = (A, N_i, R_i)$ generating L_i from the start symbol S.
$N_1 := \{S\}$,R_1 consists of the rules:

$$S \to abS, \;\; S \to \varepsilon$$

$N_2 := \{S\}$,R_2 consists of the rules:

$$S \to abSc, \;\; S \to \varepsilon.$$

$N_3 := \{S, T\}$,R_3 consists of the rules:

$$S \to abSc, \;\; S \to T, \;\; T \to abT, \;\; T \to \varepsilon$$

$N_4 := \{S, T\}$,R_3 consists of the rules:

$$S \to abSd, \;\; S \to T, \;\; T \to abTc, \;\; T \to \varepsilon$$

$N_5 := \{S, T, U\}$,R_3 consists of the rules:

$$S \to abS, \;\; S \to T, \;\; T \to abTd, \;\; T \to U, \;\; U \to abUc, \;\; U \to \varepsilon$$

2- The grammar G_5 is non-ambiguous.
Exercice 6 [/5] We consider the context-free grammar $G := (A, N, R)$ where $A = \{a, b\}$, $N = \{S, T, U\}$ and R consists of the following five rules:
r1: $S \to TU$
r2: $T \to aTT$
r3: $T \to b$
r4: $U \to aUT$
r5: $U \to bTT$
The start symbol of G is S.
1- A leftmost derivation from T to $aabbaabbb$:

$$T \to aTT \to aaTTT \to aabTT \to aabbT \to aabbaTT \to aabbaaTTT \to aabbaabTT \to aabbaabbT \to aabbaab$$

Give a rightmost derivation from T to $aabbaabbb$:

$$S \to aTT \to aTaTT \to aTaTb \to aTaaTTb \to aTaaTbb \to aTaabbb \to aaTTaabbb \to aaTbaabbb \to aabbaab$$

2- We know from q1 that:

$$S \to^* aabbaabbb$$

We remark that:

$$U \to bTT \to bbT \to bbb$$

Hence, combining rule r1 with the two above derivations:

$$S \to TU \to^* aabbaabbbU \to^* aabbaabbbbbb.$$

The above combination is also leftmost:

$$S \to TU \to aTTU \to aaTTTU \to aabTTU \to aabbTU \to aabbaTTU \to aabbaaTTTU \to aabbaabTTU$$

$$\rightarrow aabbaabbTU \rightarrow aabbaabbbU \rightarrow aabbaabbbbTT \rightarrow aabbaabbbbbT \rightarrow aabbaabbbbbb.$$

A derivation-*tree* for $aabbaabbbbbb$ is depicted on figure 4. The corresponding absract-syntax tree (or construction-tree) is given on figure 5.

3- All the non-terminals are productive and useful.
4- The grammar G is not *simple* because the righthand side of rule r1 does not begin by a terminal letter. Let us construct G', by replacing the leading non-terminal symbol T in r1 by its righthand-sides. We obtain the context-free grammar $G' := (A, N, R')$ where $A = \{a, b\}$, $N = \{S, T, U\}$ and R' consists of the following five rules:
r1': $S \rightarrow aTTU$
r1": $S \rightarrow bU$
r2: $T \rightarrow aTT$
r3: $T \rightarrow b$
r4: $U \rightarrow aUT$
r5: $U \rightarrow bTT$
We know, from the lecture, that such a transformation preserves the language.

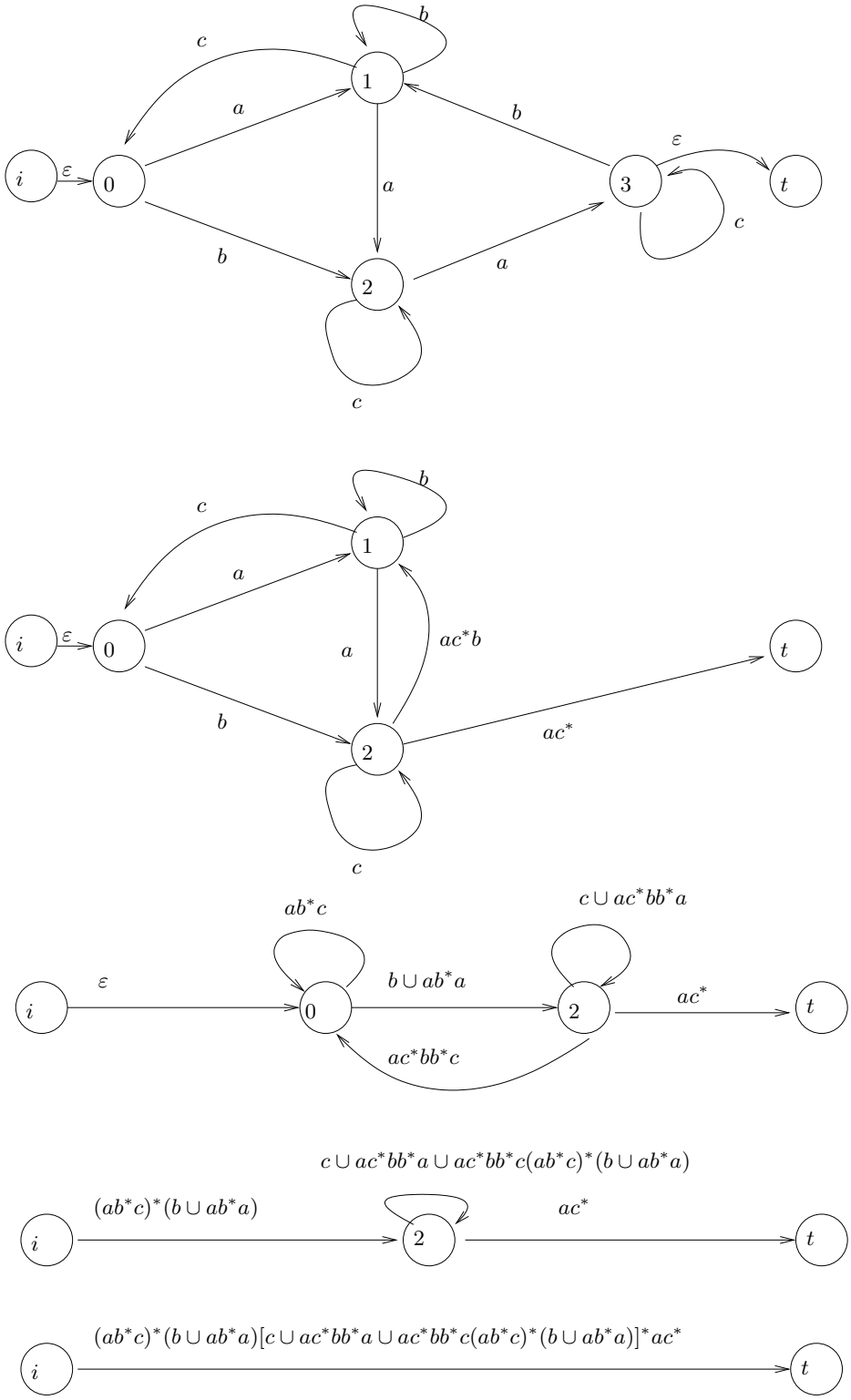

Figure 2: sequence of extended automata

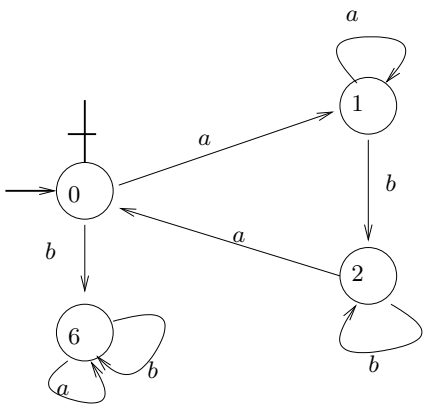

Figure 3: $\mathcal{C}$

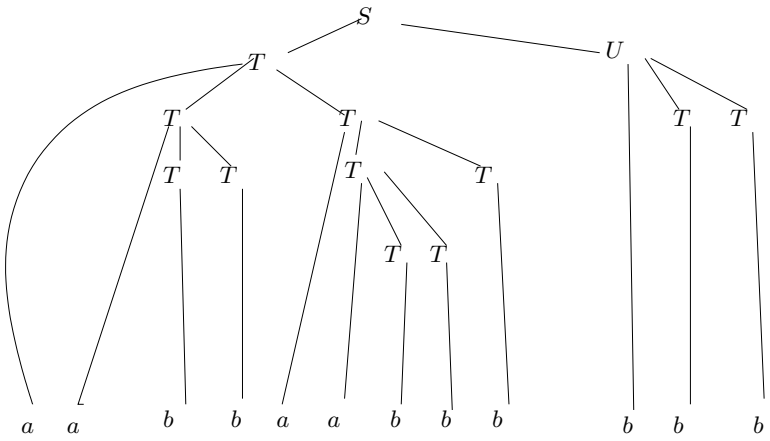

Figure 4: derivation tree

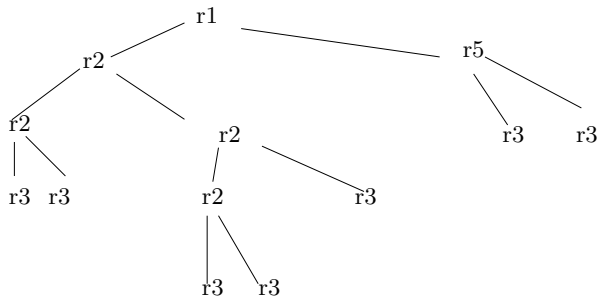

Figure 5: abstract syntax tree