Cursus: M1, computer-science
Code UE: JEIN8602
Solutions to the subject: Formal languages theory
Date: March 2018
Duration: 3H
Documents: authorized
Lectures by: Mr Géraud Sénizergues

Exercice 1 [/4] We consider the finite automaton $\mathcal{A}$ described on figure 1.

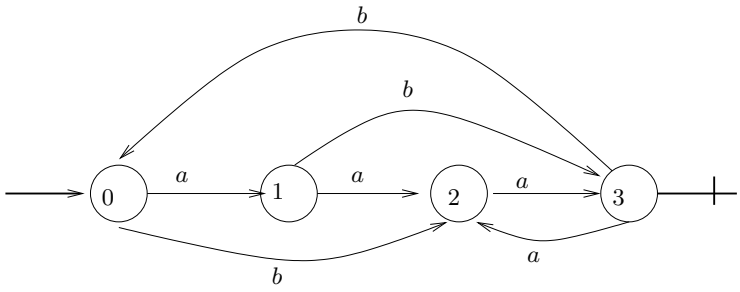

Figure 1: finite automaton $\mathcal{A}$

0- The following sequences are successful computations of $\mathcal{A}$:

$$0, a, 1, a, 2, a, 3$$
$$0, b, 2, a, 3$$
$$0, a, 1, b, 3, b, 0, b, 2, a, 3$$
$$0, b, 2, a, 3, b, 0, b, 2, a, 3$$
$$0, e, 3$$

1- 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: $0, 1, 2, 3$. We obtain the sequence of extended automata shown on figure 2.
It follows that:

$$[ab \cup (b \cup aa)a] \cdot [bab \cup (baa \cup bb \cup a)a]^*$$

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

Exercice 2 [/4] Let us consider the regular expression:

$$e := (ac)^* b (a \cup (bc)^*)^*$$

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

$$L' := (a_1 c_1)^* b_1 (a_2 \cup (b_2 c_2)^*)^*$$

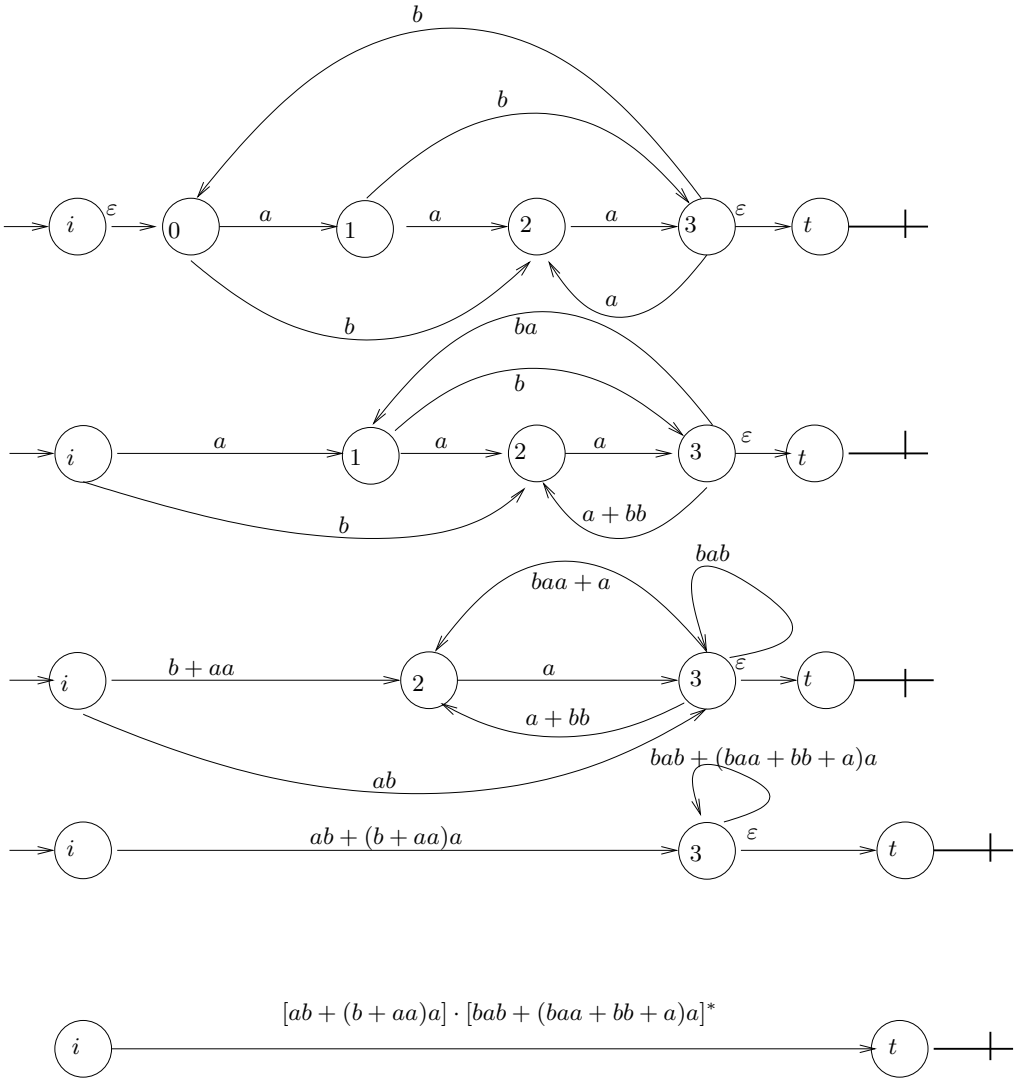

Figure 2: sequence of extended automata, ex. 2

We compute

$$Ini(L') = \{a_1, b_1\}, \quad Fin(L') = \{a_2, c_2, b_1\},$$

$$Dig(L') = \{a_1c_1, c_1a_1, c_1b_1, b_1a_2, b_1b_2, b_2c_2, c_2b_2, a_2a_2, c_2a_2, a_2b_2\}$$

This gives the finite automaton represented on figure 3, where the set of states is

$$\{0, a_1, a_2, b_1, b_2, c_1, c_2\}$$

0 is the unique initial state and the set final states is $\{a_2, c_2, b_1\}$.

Exercice 3 [/4]

1- The f.a. $\mathcal{A}$, with set of states $\{S_0, S_1, S_2\}$, initial state S_0 , final state S_2 and the following

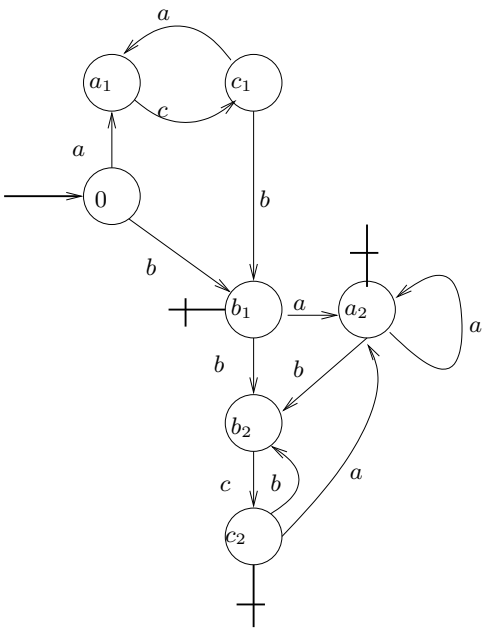

Figure 3: finite automaton for L_e

transitition table, recognizes the language $\mathrm{L}(G, S_0)$:

$x\backslash q:$	S_0	S_1	S_2
a	S_1	S_1	S_2
b	S_2	S_2	$-$

2- We take for $\mathcal{B}$ the above f.a. $\mathcal{A}$, where the arrows (i.e. transitions) have been half-turned, S_2 is initial and S_0 is final.

$x\backslash q:$	S_0	S_1	S_2
a	$-$	S_0, S_1	S_2
b	$-$	$-$	S_0, S_1

3- The following f.a. $\mathcal{C}$ is obtained by determinization of $\mathcal{B}$: its set of states is $\{S_0, S_1, S_2, \{S_0, S_1\}, \emptyset\}$, its initial state is S_2 , its final states are S_0 and $\{S_0, S_1\}$, and the transitition table is

$x\backslash q:$	S_0	S_1	S_2	$\{S_0, S_1\}$	$\emptyset$
a	$\emptyset$	$\{S_0, S_1\}$	S_2	$\{S_0, S_1\}$	$\emptyset$
b	$\emptyset$	$\emptyset$	$\{S_0, S_1\}$	$\emptyset$	$\emptyset$

Since the states S_0, S_1 are not accessible, we can just rename the (accessible) states by:

$$S_2 \mapsto p_1, \{S_0, S_1\} \mapsto p_2, \emptyset \mapsto p_3,$$

and obtain a new deterministic and complete f.a. $\mathcal{C}'$, recognizing ${}^t\mathrm{L}(G, S_0)$:

$x\backslash q:$	p_1	p_2	p_3
a	p_1	p_2	p_3
b	p_2	p_3	p_3

4- Let us compute the Nerode equivalence of $\mathcal{C}'$:
we apply the refinement algorithm exposed in the lectures:

$$\equiv_0 = \{\{p_1, p_3\}, \{p_2\}\};\ \equiv_1 = \{\{p_1\}, \{p_3\}, \{p_2\}\};\ \equiv_2 = \equiv_1;$$

Hence the Nerode equivalence over $Q_{\mathcal{C}'}$ is just the equality. This implies that $\mathcal{C}'$ is minimal, among all the *complete* deterministic finite automata recognizing ${}^t\mathrm{L}(G, S_0)$.

Exercice 4 [/5] 1- Let $A = \{a, b, c, d\}$. For every $i \in \{1, 2, 3, 4, 5\}$, we define a non-terminal alphabet N_i and a set of rules R_i over A and N_i, such that the c.f. grammar $G_i := (A, N_i, R_i)$ generates the language L_i.
$N_1 = \{S\}$, R_1 is the set of rules:

$$S \to aS,\ \ S \to Sb,\ \ S \to \varepsilon$$

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

$$S \to aSb,\ \ S \to \varepsilon,$$

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

$$S \to aS,\ \ S \to a,\ \ S \to T,\ \ T \to aTb,\ \ T \to \varepsilon.$$

$N_4 = \{S, T\}$,R_4 is the set of rules:

$$S \to acS,\ \ S \to ac,\ \ S \to T,\ \ T \to acTbd,\ \ T \to \varepsilon.$$

$N_5 = \{\sigma, S_0, S_1, T\}$,$R_5$ is the set of rules:

$$\sigma \to S_0,\ \ \sigma \to S_1,\ \ S_0 \to acS_0,\ \ S_0 \to acT,\ \ S_1 \to S_1bd,\ \ S_1 \to Tbd,$$

$$T \to acTbd,\ \ T \to \varepsilon.$$

2- G_3 is non-ambiguous: for every $p > q \geq 0$, the only derivation from S to a^pb^q is

$$S \to^{p-q} a^{p-q}S \to a^{p-q}T \to^{q} a^pTb^q \to a^pb^q.$$

3- G_4 is non-ambiguous: for every $0 \leq p < q$, the only derivation from S to a^pb^q is

$$S \to^{p-q} a^{p-q}S \to a^{p-q}T \to^{q} a^pTb^q \to a^pb^q.$$

4- Let us define the auxiliary language:

$$L'_4 := \{(ac)^p(bd)^q \mid p \geq 0, q \geq 0, p < q\}.$$

L_5 is the disjoint union of L_4 and L'_4. The set of rules $\sigma \to S_0,\ \ S_0 \to acS_0,\ \ S_0 \to acT,\ \ T \to acTbd,\ \ T \to \varepsilon$ generates L_4, in a non-ambiguous fashion (by question 3);
Similarly, the set of rules $\sigma \to S_1,\ \ S_1 \to S_1bd,\ \ S_1 \to Tbd,\ \ T \to acTbd,\ \ T \to \varepsilon$ generates L'_4, in a non-ambiguous fashion.
Thus $\mathrm{L}(G_5, S_0) = L_4$ and $\mathrm{L}(G_5, S_1) = L'_4$. It follows that $\mathrm{L}(G_5, \sigma) = L_4 \cup L'_4$ and, since the union is disjoint, G_5 is non-ambiguous.

Exercice 5 [/5] 1- We compute the subset of *productive* non-terminals of G by the fixpoint technique explained in the lecture:

$$V_1 = \{T, U\}, V_2 = \{S, T, U, V\}, V_3 = \{S, T, U, V, W\}, V_4 = \{S, T, U, V, W, Y\}, V_5 = V_4.$$

Hence the set of productive non-terminals is $\{S, T, U, V, W, Y\}$.
2- The c.f. grammar obtained by removing all the non-productive non-terminals is thus: $\hat{G} := (A, \hat{N}, \hat{R})$ where $A = \{a, b\}$, $\hat{N} = \{S, T, U, V, W, Y\}$ and $\hat{R}$ consists of the following rules:

$$\begin{array}{rrr} S \to ST & S \to T & S \to U \\ T \to bT & T \to \varepsilon & \\ U \to bU & U \to abW & U \to \varepsilon \\ V \to bT & V \to UV & \\ W \to TUV & & \\ Y \to aYb & Y \to aW & \end{array}$$

We compute the subset of *useful* non-terminals of $\hat{G}$ by the fixpoint technique explained in the lecture:

$$N_1 = \{S\}, N_2 = \{S, T, U\}, N_3 = \{S, T, U, W\}, N_4 = \{S, T, U, V, W\}, N_5 = N_4.$$

Hence the set of useful non-terminals is $\{S, T, U, V, W\}$.
3- We can thus transform the grammar $\hat{G}$ into an equivalent grammar G' where every non-terminal is productive and useful, just by restricting both the non-terminal alphabet and the rules to the subset N_5:
$G' := (A, N', R')$ where $A = \{a, b, c\}$, $N' = \{S, T, U, V, W\}$ and R' consists of the rules:

$$\begin{array}{rrr} S \to ST & S \to T & S \to U \\ T \to bT & T \to \varepsilon & \\ U \to bU & U \to abW & U \to \varepsilon \\ V \to bT & V \to UV & \\ W \to TUV & & \end{array}$$

4- Since S is productive (question 1), $\mathrm{L}(G, S) \neq \emptyset$.
5-

$$S \to^* ST^n \to^* Sb^n \to^* b^n$$

hence $\{b^n \mid n \geq 0\} \subseteq \mathrm{L}(G, S)$, which proves that $\mathrm{L}(G, S)$ is infinite.
Exercice 6 [/4] We consider the context-free grammar $G := (A, N, R)$ where $A = \{a, b\}$, $N = \{S, T, U, V, W\}$ and R consists of the following rules:

$$\begin{array}{rrr} S \to STU & S \to Sa & S \to SbW \\ S \to a & T \to aU & \\ U \to bT & U \to bU & U \to b \\ W \to Wab & W \to Wbb & W \to \varepsilon \end{array}$$

1- All the rules $W \to Wab, W \to Wbb, W \to \varepsilon$ are left-linear and they are generating $\mathrm{L}(G, W)$. Hence $\mathrm{L}(G, W)$ is regular. In fact $\mathrm{L}(G, W) = (ab + bb)^*$.

2- The set of rules

$$T \to aU, \;\; U \to bT, \; ; U \to bU, \;\; U \to b$$

is enough to generate $\mathrm{L}(G,T), \mathrm{L}(G,U)$. Since these rules are right-linear, the languages $\mathrm{L}(G,T), \mathrm{L}(G,U)$ are regular. From these rules we obtain the regular expressions:

$$\mathrm{L}(G,T) = (ab^*b)^*ab^+, \;\; \mathrm{L}(G,U) = (ba+b)^*b.$$

3- We consider the context-free grammar $H := (A', N', R')$ where $A' = \{a, b, t, u, w\}$, $N' = \{S\}$ and R' consists of the following rules:

$$\begin{array}{ll} S \to Stu & S \to Sa \\ S \to Sbw & S \to a \end{array}$$

$$\mathrm{L}(H,S) = (tu + a + bw)^*a.$$

4- The language $\mathrm{L}(G,S)$ is obtained from the language $\mathrm{L}(H,S)$ by applying the substitution

$$a \mapsto a, \;\; b \mapsto b, t \mapsto \mathrm{L}(G,T), \;\; u \mapsto \mathrm{L}(G,U), \;\; w \mapsto \mathrm{L}(G,W).$$

Using the above expressions, we obtain the following regular expression for $\mathrm{L}(G,S)$:

$$[(ab^+)^*ab^+(ba+b)^*b + a + b(ab+bb)^*]^*a$$

5- The words a and aa both belong to $\mathrm{L}(G,S)$. But every simple language is *prefix-free*. Hence $\mathrm{L}(G,S)$ is not a simple language.
6- Let c be a letter not in $\{a,b\}$. A simple grammar K generating the language $\mathrm{L}(G,S)\cdot c$ can be built along the following principles:
- build a f.a. $\mathcal{A}$ for the regular expression $(ab^+)^*ab^+(ba+b)^*b + a + b(ab+bb)^*]^*ac$ (that represents $\mathrm{L}(G,S)\cdot c$)
- transform $\mathcal{A}$ into the minimal deterministic,complete f.a. $\mathcal{B} = \langle Q, \{a,b,c\}, \delta, q_0, F\rangle$ recognizing the same language; note that, since $\mathcal{B}$ is minimal, it has at most one non-coaccessible state (that we call the sink); since $\mathrm{L}(\mathcal{B})$ is prefix-free, if $p \in F$ and $(p,x,q) \in \delta$, then q is the sink-state;
- build from $\mathcal{B}$ a right-linear grammar H generating $\mathrm{L}(G,S)\cdot c$: the set of non-terminals is $N := Q \setminus F$, the rules are all the

$$p \to xq$$

for $(p,x,q) \in \delta, x \neq c$, union all the

$$p \to c$$

for $(p,c,q) \in \delta$.
Since every transition (p,c,q) must lead to the sink q, the above grammar H indeed generates $\mathrm{L}(\mathcal{B})$. Suppose that $p \to xq, p \to xr$ are two rules of H:
determinism of $\mathcal{B}$ implies that $q = r$. Hence H is *simple* grammar that generates $\mathrm{L}(G,S)\cdot c$.