Cursus: M1, computer-science
Code UE: JEIN8602
Solutions to the subject: Formal languages theory
Date: March 4th 2017
Duration: 3H
Documents: authorized
Lectures by: Mr Géraud Sénizergues

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

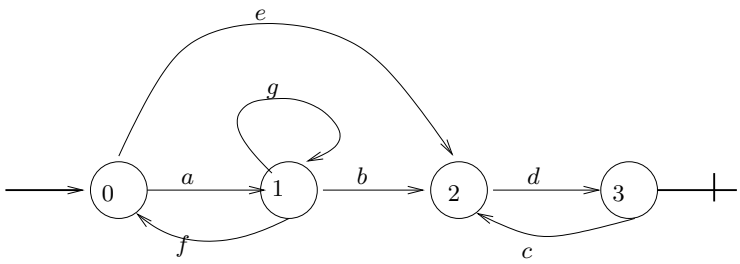

Figure 1: finite automaton $\mathcal{A}$

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

$$0, a, 1, g, 1, b, 2, d, 3$$
$$0, e, 2, d, 3$$
$$0, a, 1, b, 2, d, 3, c, 2, d, 3$$
$$0, a, 1, f, 0, e, 2, d, 3$$

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:

$$(a(f \cup ga)^* bd \cup e)(cd)^*$$

is a regular expression for $L_{\mathcal{A}}$.
2- A regular expression for the language $h(L_{\mathcal{A}})$ is obtained by replacing each letter $x \in \{a, b, c, d, e, f, g\}$ by its image $h(x)$ in the above regular expression. We thus obtain the expression:

$$(a(ba \cup aa)^* bc \cup aa)^*(bac)^*$$

3- A finite automaton recognizing the language $h(L_{\mathcal{A}})$ is depicted on figure 3

Exercice 2 [/4]

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

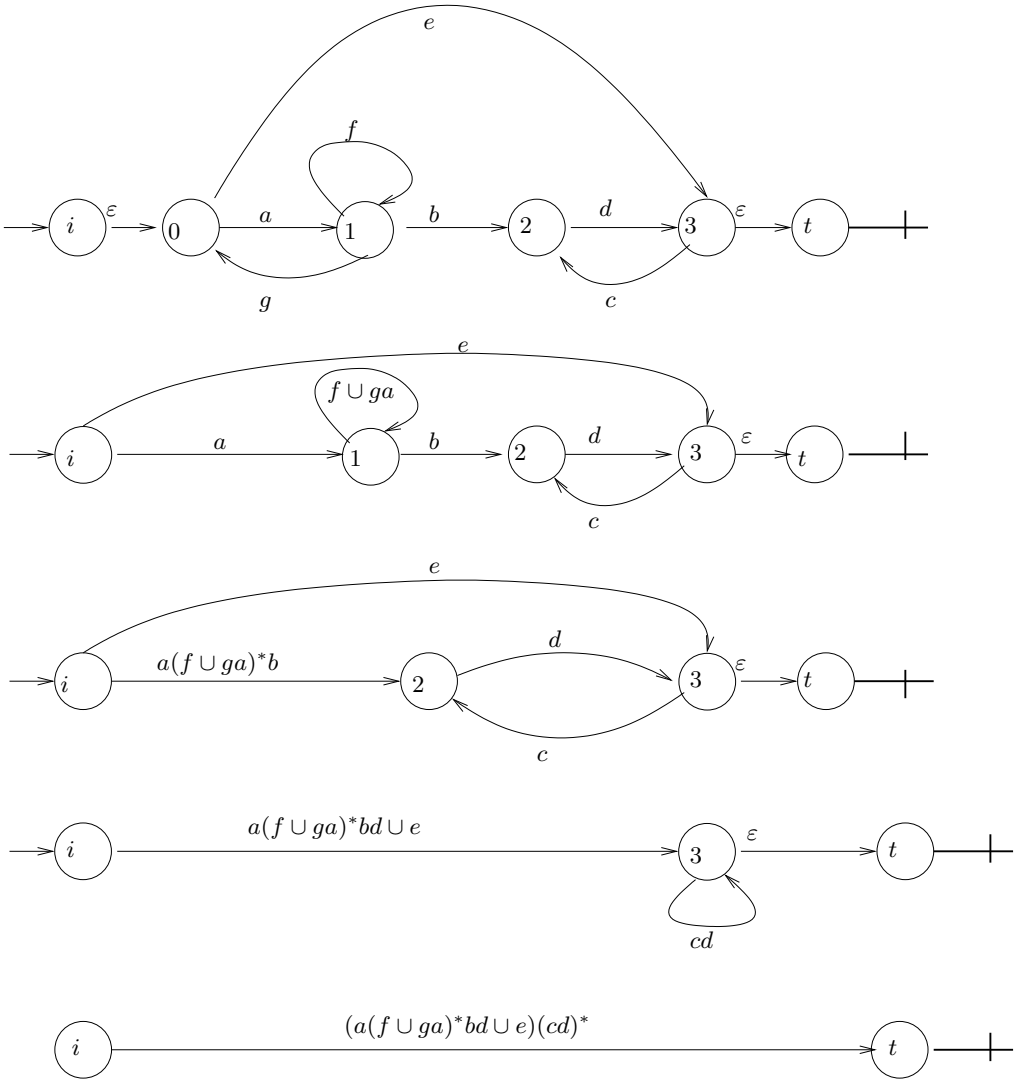

Figure 2: sequence of extended automata, ex.1

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

$$L' := (a_1 b_1^* c_1)^* a_2 b_2 (a_3 \cup b_3)^*$$

We compute

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

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

This gives the finite automaton represented on figure 4, 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.

Exercice 3 [/4]

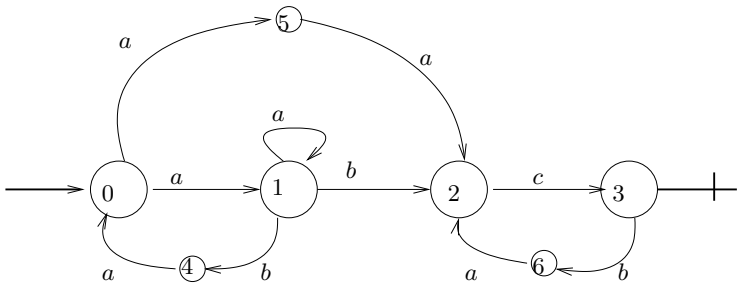

Figure 3: a finite automaton for $h(L_{\mathcal{A}})$

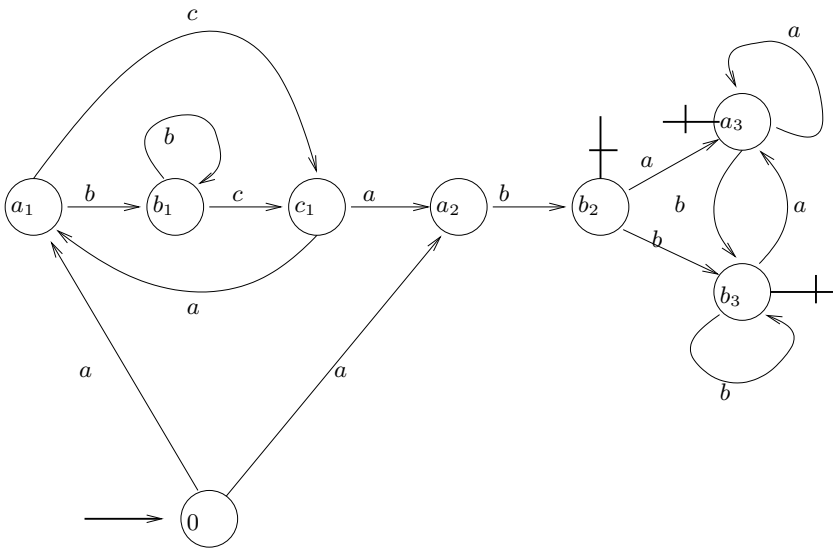

Figure 4: finite automaton for L_e

1- The d.f. automaton $\mathcal{A}_2$ recognizes L_2 (see figure 5).

2- Let us compute the Nerode equivalence of $\mathcal{A}_2$:
we apply the refinement algorithm (called " Moore's algorithm") exposed in the lectures:

$$\equiv_0 = \{\{0,1,2\},\{3\}\};\ \equiv_1 = \{\{0,1\},\{2\},\{3\}\};\ \equiv_2 = \{\{0\},\{1\},\{2\},\{3\}\}$$

Hence $\equiv_2 = \equiv_3 = \equiv$. Hence the Nerode equivalence over Q is just the equality. This implies that $\mathcal{A}_2$ is minimal (among all the *complete* deterministic finite automata recognizing L_2).
3- Since $\mathcal{A}_2$ is deterministic and complete, $X^* - L_2$ is recognized by the d.f.a. $\mathcal{A}'_2$ obtained from $\mathcal{A}_2$ by exchanging the final states with the non-final states. Thus the set of final states of $\mathcal{A}'_2$ is $\{0,1,2\}$.
Since the state 3 is not co-accessible, we cancel this state and obtain a f.a. $\mathcal{A}''_2$ which still recognizes the language $X^* - L_2$. Let us apply the Brozowsky-Mc-Cluskey algorithm on $\mathcal{A}''_2$ (see figure 6). We obtain the rational expression:

$$b^*(\varepsilon \cup a(a \cup ba)^*(\varepsilon \cup b))$$

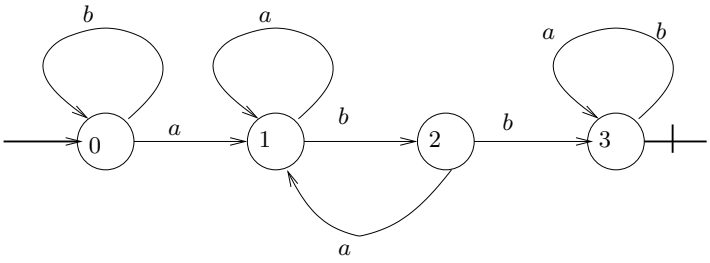

Figure 5: d.f.a. $\mathcal{A}_2$ for L_2

4*- The d.f. automaton $\mathcal{A}_n$ recognizes L_n (see figure 7).
Let $i \in [1, 1+n]$:

$$\hat{\delta}(0, b^{1+n-i}) = 0 \notin F, \;\; \hat{\delta}(i, b^{1+n-i}) = 1+n \in F$$

Hence $0 \not\equiv i$.
Let $i, j \in [1, 1+n]$ such that $i < j$:

$$\hat{\delta}(j, b^{1+n-j}) = 1+n \in F, \;\; \hat{\delta}(i, b^{1+n-j}) = 1+n-(j-i) \notin F$$

Hence $i \not\equiv j$.
It follows that all the states $p \in [0, 1+n]$ are two by two non-equivalent. Hence $\mathcal{A}_n$ is the minimal automaton of L_n. It has exactly $2+n$ states.
5- $L = \{a,b\}^* abb^* \{a,b\}^*$, hence L is regular.
The language M_n is the product

$$M_n = L_n \cdot L_{n+1}.$$

It is thus the product of two regular languages, hence it is regular.

$$M \cap ab^+ab^+ = \{ab^n ab^{n+1} \mid n \geq 1\}.$$

The language $\{ab^n ab^{n+1} \mid n \geq 1\}$ does not fulfill the star-lemma, hence is not regular. Since the intersection of two regular sets is regular, M cannot be regular.

Exercice 4 [/5]

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_4 consists of the rules:

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

$N_5 := \{S, T_1, T_2\}$, R_5 consists of the rules:

$$S \to abSc, \;\; S \to T_1, \;\; S \to T_2, \;\; T_1 \to abT_1, \;\; T_1 \to ab, \;\; T_2 \to T_2c, \;\; T_2 \to c$$

2- The grammar G_5 is non-ambiguous:
If $w = (ab)^n c^m$ for some $n \geq 0, m \geq 0, n > m$, then it is generated by a derivation of the form

$$S \to^m (ab)^m S c^m \to (ab)^m T_1 c^m \to^{n-m} (ab)^n c^m.$$

If $w = (ab)^n c^m$ for some $n \geq 0, m \geq 0, n < m$, then it is generated by a derivation of the form

$$S \to^n (ab)^n S c^n \to (ab)^n T_2 c^n \to^{m-n} (ab)^n c^m.$$

Hence every word of $L(G_5, S)$ has *exactly one* leftmost derivation (and, in fact, since the grammar is linear, exactly one derivation).

Exercice 5 [/5] We consider the context-free grammar $G := (A, N, R)$ where $A = \{a, b, c\}$, $N = \{S_1, S_2, S_3, S_4, S_5\}$ and R consists of the following 12 rules:

$$\begin{array}{rrr} S_1 \to aS_1S_1 & S_1 \to bS_4S_1 & S_1 \to S_3c \\ S_2 \to aS_2 & S_2 \to aS_5a & S_3 \to S_3S_1 \\ S_3 \to aS_4 & S_3 \to S_1S_3S_1 & S_4 \to a \\ S_4 \to S_1S_4 & S_5 \to cS_5 & S_5 \to aS_5S_2 \end{array}$$

The start symbol of G is S_1.
1- We compute the subset of *productive* non-terminals of G by the fixpoint technique explained in the lecture:

$$V_1 = \{S_4\}, V_2 = \{S_4, S_3\}. V_3 = \{S_4, S_3, S_1\}, V_4 = V_3.$$

Hence the set of productive non-terminals is $\{S_1, S_3, S_4\}$.
2- We compute the subset of *useful* non-terminals of G by the fixpoint technique explained in the lecture:

$$N_1 = \{S_1\}, N_2 = \{S_1, S_3, S_4\}, N_3 = N_2$$

Hence the set of useful non-terminals is $\{S_1, S_3, S_4\}$.
3- We can thus transform the grammar 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_2:
$G' := (A, N', R')$ where $A = \{a, b, c\}$, $N' = \{S_1, S_3, S_4\}$ and R' consists of the following 8 rules:

$$\begin{array}{rrr} S_1 \to aS_1S_1 & S_1 \to bS_4S_1 & S_1 \to S_3c \\ S_3 \to S_3S_1 & S_3 \to aS_4 & S_3 \to S_1S_3S_1 \\ S_4 \to a & S_4 \to S_1S_4 & \end{array}$$

4- The language $\mathrm{L}(G, S_1)$ is not empty, since the non-terminal S_1 is productive.
5- We observe that:

$$S_1 \to aS_1S_1 \to^* aS_1aac \text{ and } S_1 \to^* aac$$

hence $\{a^n(aac)^{n+1}\} \subseteq \mathrm{L}(G, S_1)$, which shows that the language $\mathrm{L}(G, S_1)$ is *infinite*.

Exercice 6 [/4] We consider the context-free grammar $G := (A, N, R)$ where $A = \{a, b, c, d\}$, $N = \{S_0, S_1, S_2, S_3\}$ and R consists of the following rules:

r1: $S_0 \to aS_1$
r2: $S_0 \to aS_2$
r3: $S_1 \to bS_2$
r4: $S_1 \to cS_3$
r5: $S_2 \to bS_1$
r6: $S_2 \to dS_3$
r7: $S_3 \to \varepsilon$

The start symbol of G is S_0.

1-

$$S_1 \to bS_2 \to bbS_1 \to^2 bbbbS_1 \to bbbbcS_3 \to bbbbc\varepsilon = bbbbc.$$

$$S_1 \to^4 bbbbS_1 \to bbbbbS_2 \to bbbbbdS_3 \to bbbbbd.$$

2-

$$S_2 \to^4 bbbbS_2 \to bbbbdS_3 \to bbbbd,$$

$$S_2 \to^4 bbbbS_2 \to bbbbbS_1 \to bbbbbcS_3 \to bbbbbc.$$

3-

$$S_0 \to aS_2 \to^* a(b^4d) \text{ by Q2 .}$$

The corresponding derivation-tree is depicted on figure 8.

4-4.1

$$L(G, S_1) = (b^2)^*c \cup (b^2)^*bd, \quad L(G, S_2) = (b^2)^*d \cup (b^2)^*bc.$$

Let $w \in L(G, S_1) \cap L(G, S_2)$:
either $w = b^nc$ (case 1) or $w = b^nd$ (case 2).
case 1: $S_1 \to^* w$, hence $n \equiv 0 \pmod 2$ and $S_2 \to^* w$, hence $n \equiv 1 \pmod 2$, which is contradictory. Hence this case is impossible.
case 2: $S_1 \to^* w$, hence $n \equiv 1 \pmod 2$ and $S_2 \to^* w$, hence $n \equiv 0 \pmod 2$, which is also contradictory. Hence this case is impossible.
It follows that $L(G, S_1) \cap L(G, S_2) = \emptyset$.

4.2 Let us first remark that, for every $w \in \{a, b\}^*$, there exists *at most one* left-derivation $S_1 \to^* w$ and there exists *at most one* left-derivation $S_2 \to^* w$
because: for $v \in \{S_1, S_2\}$ and $x \in \{a, b\}$ there exists exactly one rule in $\{v\} \times x \cdot (N \cup A)^*$, and for $v = S_3$ there exists exactly one rule in $\{v\} \times (N \cup A)^*$.
Let us now consider a word $w \in a \cdot (L(G, S_1) \cup L(G, S_2))$:

$$w = aw', \text{ where } w' \in L(G, S_1) \cup L(G, S_2).$$

By 4.1, either $w' \in L(G, S_1) \setminus L(G, S_2)$ (case 4.2.1) or $w' \in L(G, S_2) \setminus L(G, S_1)$ (case 4.2.2).
In case 4.2.1 (resp. 4.2.2) there is exactly one derivation from S_1 (resp. S_2) to w'. It follows that there exists exactly one derivation $S_0 \to^* w$.
We have proved that G is non-ambiguous.
Since $S_3 \to \varepsilon$ has a rhs that does not begin with a terminal letter, G is not simple.

5- Let us note that $L(G, S_0) = ab^*\{c, d\}$. This language is generated by the grammar: $G' := (A, N', R')$ where $A = \{a, b, c, d\}$, $N' = \{S', T\}$ and R consists of the following rules:

r1: $S' \to aT$
r2: $T \to bT$
r3: $T \to c$
r4: $T \to d$
which is simple.
6- We note that

$$L(H, T_1) = \{b^{2n}c\#cb^{2n} \mid n \geq 0\} \cup \{b^{2n+1}d\#db^{2n+1} \mid n \geq 0\},$$

$$L(H, T_2) = \{b^{2n+1}c\#cb^{2n+1} \mid n \geq 0\} \cup \{b^{2n}d\#db^{2n} \mid n \geq 0\}.$$

hence

$$L(H, S_1) \cap L(H, S_2) = \emptyset. \tag{1}$$

and

$$L(H, T_0) = a(L(H, T_1) \cup L(H, T_2))a. \tag{2}$$

Given a word $w \in L(H, T_0)$, by (2), it belongs either to $aL(H, T_1)a$ or to $aL(H, T_2)a$, and by (1) it cannot belong to both. We conclude that H is non-ambiguous.
Since $T_0 \to U_1 a$ has a rhs that does not begin with a terminal letter, H is not simple.
7- This language is generated by the grammar: $H' := (A, N', R'')$ where $A = \{a, b, c, d\}$, $N' = \{T', T\}$ and R'' consists of the following rules:
r1: $T' \to aTa$
r2: $T \to bTb$
r3: $T \to c\#c$
r4: $T \to d\#d$
which is simple.

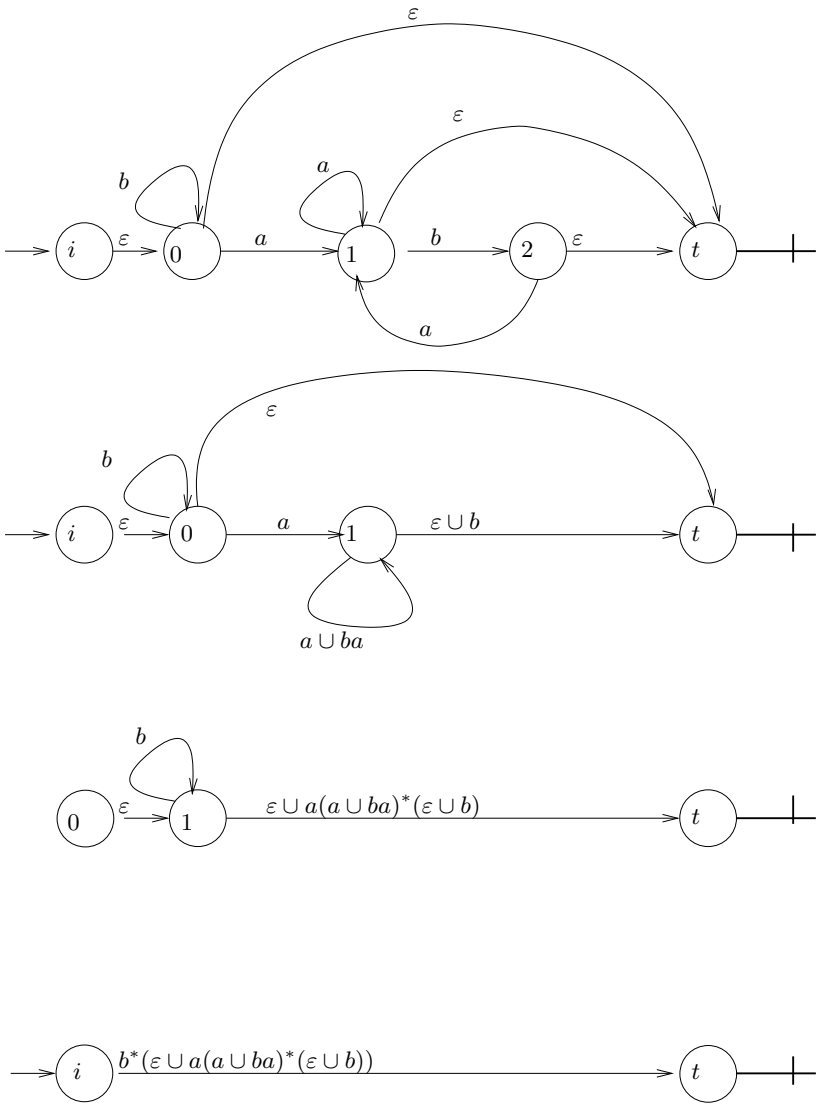

Figure 6: sequence of extended automata, ex.3

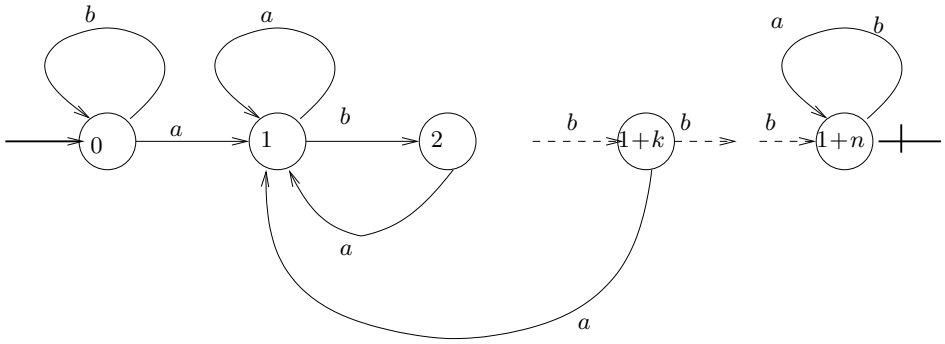

Figure 7: d.f.a. $\mathcal{A}_n$ for L_n (ex. 3)

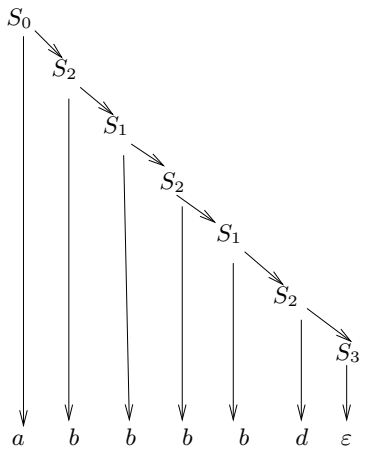

Figure 8: a derivation tree for *abbbbd*