

## TD N° 5

**1. Automate reconnaissant une expression régulière**

Soit  $E = (a + b)^*(abb + \varepsilon)$ . Appliquer

a. L'algorithme de Glushkov

b. L'algorithme de Thompson

pour construire un automate reconnaissant le langage décrit par  $E$ .

Quel est le langage décrit par cette expression régulière ?

**2. Minimisation d'automates**

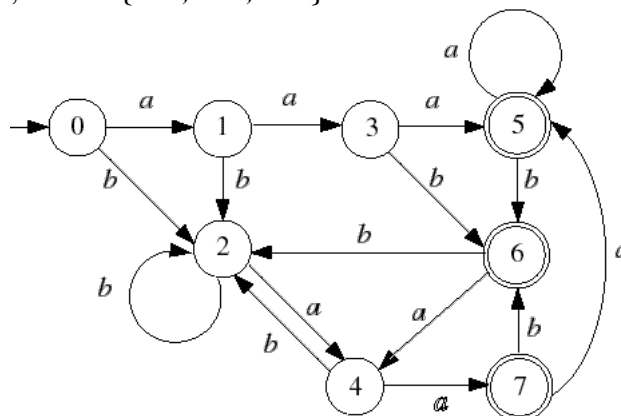
a. Soit le langage fini :  $\{ba, baa, aba, aaa\}$ .

- Donner un AFD  $A_1$  le reconnaissant en essayant intuitivement de le rendre « minimal ».

- Donnez un AFD  $A_2$  en appliquant la méthode de l'« arbre des lettres » (ou « trie » en anglais, voir TD n° 2). Intuitivement : comment le minimiser ?

- Appliquez l'algorithme de minimisation du cours et retrouvez  $A_1$ .

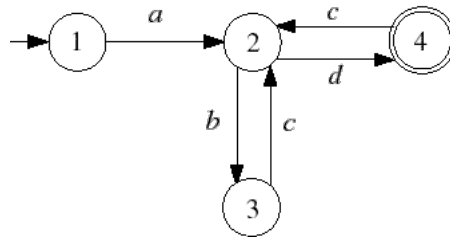
b. Appliquer l'algorithme de minimisation du cours à l'automate obtenu dans l'exercice **1-b** de la feuille TD n°4 « Reconnaissance d'un ensemble fini de motifs », rappelé ci-dessous, reconnaissant  $\{a, b\}^*L$ , où  $L = \{aab, baa, aaa\}$ .

**3. Un langage non régulier**

a. Donner tous les mots de longueur 4 reconnus par l'automate suivant, en indiquant le chemin suivi dans l'automate.

Pour chacun de ces mots :

- montrer que le chemin passe deux fois par un même état,
- en répétant indéfiniment la boucle correspondante, donner une expression rationnelle qui définit une infinité de mots reconnus par l'automate.



**b. Généralisation : Lemme de la pompe.**

Soit  $L$  un langage régulier. Il existe un entier  $p$  tel que pour tout mot  $w$  de  $L$  de longueur  $|w| \geq p$  se décompose en :

$$w = (u x v) \quad \text{avec } |x| > 0 \text{ et } |u x| \leq p \quad \text{et } \forall z (u x^z v) \text{ est dans } L$$

Montrer que le langage  $L = \{a^n b^n / n = 0, 1, 2, \dots\}$  n'est pas régulier.

Indication : Supposer qu'un automate  $A$  reconnaît  $L$ . Soit  $p$  le nombre d'état de  $A$  et  $n > p$ . Si  $a^n b^n$  est reconnu par  $A$ , quels autres mots le sont aussi ? (utiliser une boucle de l'automate pour répondre, en tenant compte du fait que cette boucle est incluse dans la partie  $a^n$ ).

**4. Automates à pile et langages algébriques**

L'exercice précédent montre qu'il existe des langages « naturels » non réguliers. Se pose la question de la reconnaissance de ces langages. On invente alors un dispositif proche des AFD (ou AFN) mais « plus puissant » : les automates à pile. L'idée est simplement de gérer, en plus de l'automate « classique » une pile en associant à chaque transition une opération sur cette pile.

Définition : un *automate à pile* (AP) pour un alphabet  $\Sigma$  est la donnée :

- Un ensemble  $Q$  d'états, avec un état initial  $q_0$ .
- Une pile  $P$  d'éléments pris dans un alphabet  $\Gamma$  (qui peut être disjoint ou non de  $\Sigma$ )
- Une fonction de transition  $\delta : \delta(q, x, top) = (q', action)$ , où  $x$  est la lettre lue,  $top$  désigne la lettre du sommet de pile et  $action$  une action qui concerne le sommet de la pile : *empiler*( $x$ ), *dépiler*, ou *rien*, où  $x$  est un élément de  $\Gamma$  (on ne peut donc dépiler que si la pile n'est pas vide).

On note les transitions :  $(x, top) // action$ .

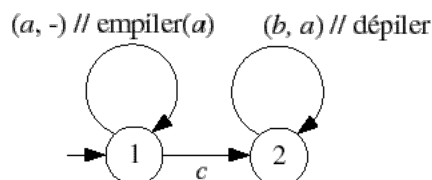
On met  $-$  à la place de  $top$  quand la transition est possible quelle que soit l'état de la pile.

On utilise aussi des transitions habituelles notées  $x$ , correspondant à  $(x, -) // rien$ , c'est-à-dire indépendantes de la pile et sans action associée.

Un mot  $w$  sur  $\Sigma^*$  est reconnu s'il existe un « parcours » de l'AP tel que, la pile étant initialement vide : 1)  $w$  est lu en entier et 2) la pile est vide.

**NB.** Il existe des variantes (équivalentes), par exemple avec un ensemble d'état finaux à atteindre pour « réussir ».

Exemple 1 : AP reconnaissant  $\{a^n c b^n / n \geq 0\}$



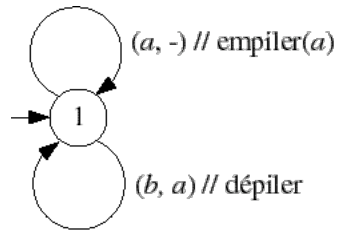
Les AP caractérisent donc une classe de langages plus vaste que les langages réguliers : *les langages algébriques* (ou *hors contexte*), dont nous verrons une caractérisation par des *grammaires* en seconde partie du cours.

Questions :

**a.** Définir un AP reconnaissant le langage de l'exercice 3 :  $L = \{a^n b^n / n = 0, 1, 2, \dots\}$

Donner la valeur de la pile pour chaque nouvelle lettre lue dans le mot *aaabbb* et vérifier qu'il est accepté. Même question pour le mot *abab*, est-il accepté ?

**b.** Donner la valeur de la pile pour chaque nouvelle lettre lue dans le mot *aaabbabb* par l'AP suivant :



préfixe lu = *a*, pile = *a*

préfixe lu = *aa*, pile = *aa*, etc.

En déduire que le mot *aaabbabb* est accepté.

Même question pour le mot *abba*. Est-il accepté ?

En déduire le langage reconnu par l'automate.

**c.** Définir un AP reconnaissant les expressions *bien parenthésées*. On pourra prendre  $\Sigma = \{ (, ), a \}$  ou *a* symbolise une lettre qui n'est pas une parenthèse. Exemple : *a(a(aa)(a))a* sont OK mais ni *a()a*, ni *a(a(aa)(a)a*, ni *a(a(aa)(a))a*.

**d.** Définir un AP reconnaissant les *expressions arithmétiques préfixées* sur les opérateurs + et \*. Exemple : *+ 2 \* 3 4 5* est OK, mais pas *+ 2 \* 3 4 + 5*.

## 5. Exercice supplémentaire : un cas défavorable pour la déterminisation

Soit le langage  $L = \{a, b\}^* a \{a, b\} \{a, b\}$

Donner un AFN reconnaissant ce langage, puis le déterminer. Vérifier que si *n* est le nombre d'états de l'AFN, alors l'AFD équivalent a  $2^{n-1}$  états.

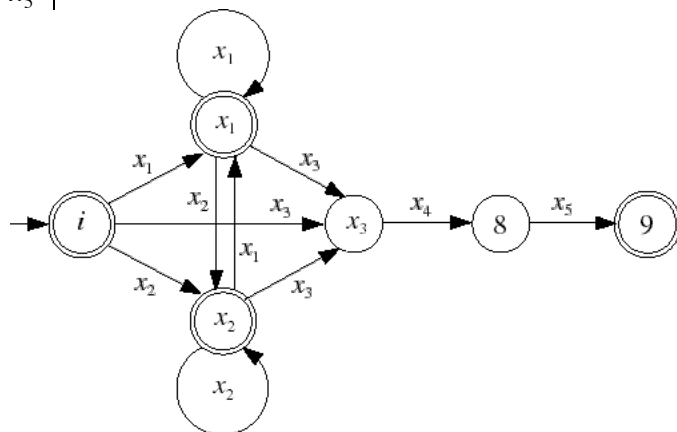
TD N° 5 - CORRECTION

1-a. Algorithme de Glushkov

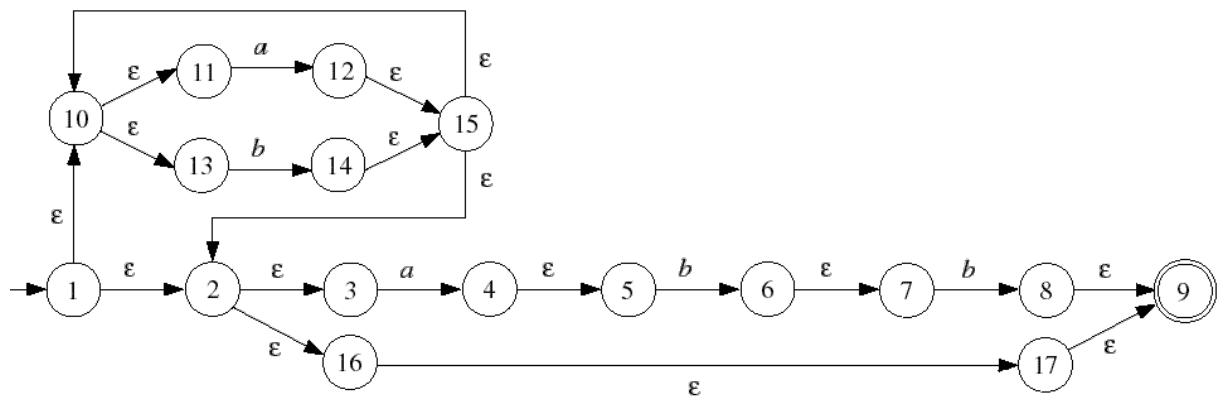
$$(x_1 + x_2)^*(x_3x_4x_5 + \epsilon)$$

- Premier =  $\{x_1, x_2, x_3\}$ ,
- Dernier =  $\{x_1, x_2, x_5\}$ ,

	Suivant
$x_1$	$x_1, x_2, x_3$
$x_2$	$x_1, x_2, x_3$
$x_3$	$x_4$
$x_4$	$x_5$
$x_5$	

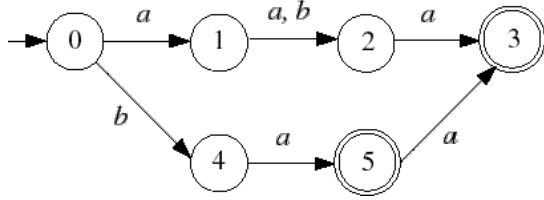


1-b. Algorithme de Thompson

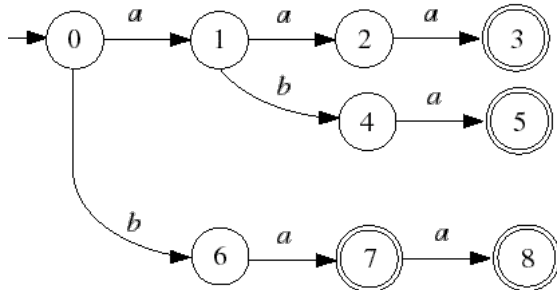


2. Minimisation d'automate

2-a. Automate intuitif (il est minimal) :



Arbre des lettres :



On voit intuitivement qu'on peut regrouper 3, 5 et 8 , ainsi que 2 et 4.

**2-b. Rappel : idée de l'algorithme**

Le but est d'obtenir une partition de l'ensemble des états cohérente avec les transitions, c'est-à-dire :

Si  $q$  et  $q'$  sont dans la même classe  $C$ , il ne peut y avoir de transitions pour une même lettre telles que :  $q \xrightarrow{x} q_1$  dans  $C_1$  et  $q' \xrightarrow{x} q_2$  dans  $C_2$  ( $x$  « distingue, sépare »  $q$  et  $q'$ ).

Et on va donc « éclater » le premier état en 2.

On aura alors une situation du type de la figure ci-dessous. Un parcours, par exemple  $aba$  conduira à la même classe quelque soit le point de départ dans  $C$ . On peut alors identifier tous les états d'une même classe.

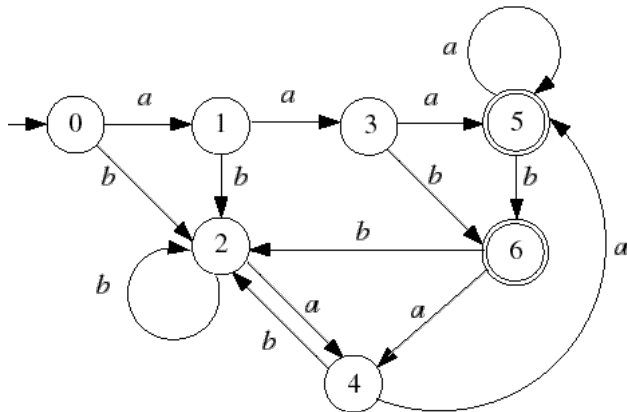
Quand on ne peut lire une lettre à partir d'une classe, on peut mettre /.

	0	1	2	3	4	5	6	7
$\sim_0$	$[0]_0$	$[0]_0$	$[0]_0$	$[0]_0$	$[0]_0$	$[5]_0$	$[5]_0$	$[5]_0$

	0	1	2	3	4	5	6	7
$\sim_0$	$[0]_0$	$[0]_0$	$[0]_0$	$[0]_0$	$[0]_0$	$[5]_0$	$[5]_0$	$[5]_0$
$a$				$[5]_0$	$[5]_0$		$[0]_0$	
$b$				$[5]_0$			$[0]_0$	
$\sim_1$	$[0]_1$	$[0]_1$	$[0]_1$	$[3]_1$	$[4]_1$	$[5]_1$	$[6]_1$	$[5]_1$

	0	1	2	3	4	5	6	7
$\sim_1$	$[0]_1$	$[0]_1$	$[0]_1$	$[3]_1$	$[4]_1$	$[5]_1$	$[6]_1$	$[5]_1$
$a$		$[3]_1$	$[4]_1$	$[5]_1$				
$b$				$[6]_1$	$[0]_1$			
$\sim_2$	$[0]_2$	$[1]_2$	$[2]_2$	$[3]_2$	$[4]_2$	$[5]_2$	$[6]_2$	$[5]_2$

	0	1	2	3	4	5	6	7
$\sim_2$	$[0]_2$	$[1]_2$	$[2]_2$	$[3]_2$	$[4]_2$	$[5]_2$	$[6]_2$	$[5]_2$
$a$								
$b$								
$\sim_3$	$[0]_3$	$[0]_3$	$[0]_3$	$[3]_3$	$[4]_3$	$[5]_3$	$[6]_3$	$[5]_3$

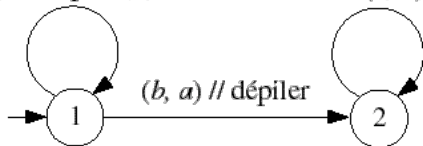


#### 4. Automates à pile et langages algébriques

##### 4-a.

$(a, -)$  // empiler( $a$ )

$(b, a)$  // dépiler



##### 4-b.

préfixe lu =  $a$ , pile =  $a$

préfixe lu =  $aa$ , pile =  $aa$

préfixe lu =  $aaa$ , pile =  $aaa$

préfixe lu =  $aaab$ , pile =  $aa$

préfixe lu =  $aaabb$ , pile =  $a$

préfixe lu =  $aaabba$ , pile =  $aa$

préfixe lu =  $aaabbab$ , pile =  $a$

préfixe lu =  $aaabbabb$ , pile = vide, donc  $aaabbabb$  accepté

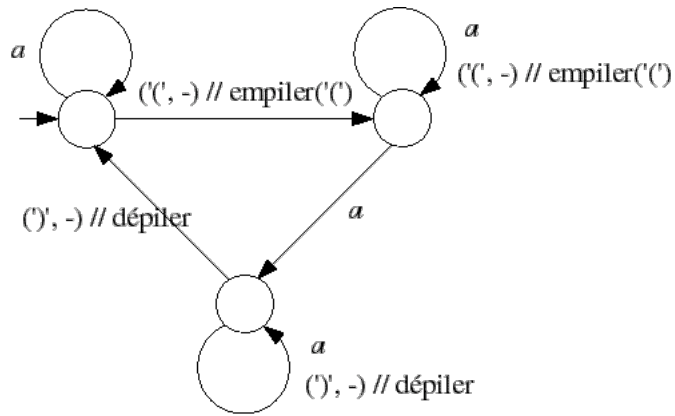
préfixe lu =  $a$ , pile =  $a$

préfixe lu =  $ab$ , pile = vide

impossible de lire le  $b$  suivant, donc  $abba$  non accepté

$L$  = mots avec autant de  $a$  que de  $b$ , et tout préfixe contient plus de  $a$  que de  $b$ .

##### 4-c.



**5. Un cas défavorable pour la détermination**

