• *Le complément d'un langage reconnaissable*

Le complément d'un langage reconnaissable (l'ensemble de mots sur le même alphabet n'appartenant pas au langage en question) est encore reconnaissable.

Soit D un automate déterministe et complet reconnaissant le langage X.

Pour construire un automate reconnaissant le complément de X, il suffit de prendre comme ensemble d'états terminaux le complément de l'ensemble d'états terminaux de D.

Exemple

Construisons un automate sur $A = \{a, b\}$ qui reconnaît l'ensemble des mots n'ayant pas aba en facteur.

On commence par la construction d'un automate (non déterministe) reconnaissant tous les mots ayant aba en facteur.

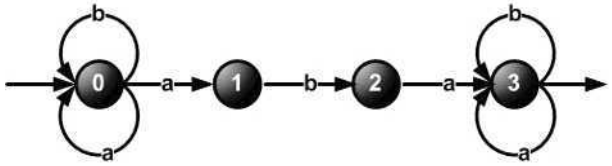

Figure 5.8

Appliquons l'algorithme de déterminisation :

Tableau 5.7

	État	États résultant	
		en lisant a	en lisant b
(initial)	(0)	(0,1)	(0)
	(0,1)	(0,1)	(0,2)
	(0,2)	(0,1,3)	(0)
(terminal)	(0,1,3)	(0,1,3)	(0,2,3)
(terminal)	(0,2,3)	(0,1,3)	(0,3)
(terminal)	(0,3)	(0,1,3)	(0,3)

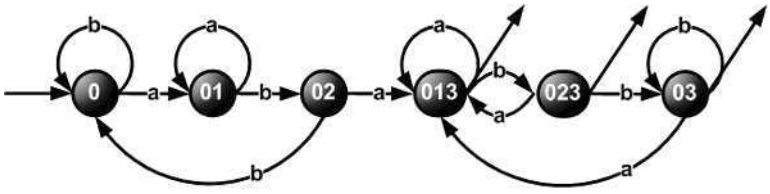

Figure 5.9

Cet automate déterministe et complet reconnaît tous les mots ayant ***aba*** en facteur. Il a trois états terminaux : (0,1,3), (0,2,3) et (0,3).

Maintenant pour obtenir un automate (lui aussi déterministe et complet) reconnaissant tous les mots **qui n'ont pas *aba*** en facteur, il suffit de faire de tous les états terminaux, des états non terminaux, et vice versa :

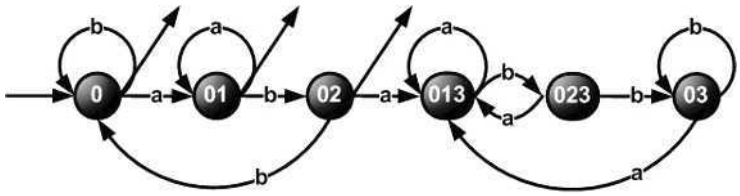

Figure 5.10

Ici, on a obtenu un automate complet sans qu'il y ait besoin de le compléter en introduisant l'état poubelle. Évidemment, ceci n'est pas toujours le cas. **Si on a affaire à un automate non complet, on n'a pas le droit de remplacer directement les états terminaux par des états non terminaux et vice versa : il faut d'abord le compléter.** Alors l'état poubelle (qui ne peut pas être terminal) devient un état terminal de l'automate reconnaissant le complément du langage.

Minimisation

Pour tout langage reconnaissable il existe ***le plus petit*** (c.-à-d. contenant le plus petit nombre d'états) automate déterministe qui le reconnaît, et il est ***unique*** : c'est l'automate minimal du langage.

• *Algorithme de minimisation par la méthode des équivalences*

Il est basé sur la notion de partitionnement de l'ensemble d'états de l'automate.

Principe Tout d'abord, si l'automate à minimiser n'est pas complet, *il faut lui ajouter l'état poubelle pour qu'il le devienne.* Sinon on risque de faire une erreur grave, qui se manifeste facilement au cas d'un automate déterministe non complet dont tous les états sont des états terminaux (essayez le voir vous-mêmes après avoir compris l'algorithme de minimisation).

On sépare tous les états de l'automate déterministe initial en deux groupes : terminaux et non-terminaux. Puis, on analyse la table des transitions en marquant vers quel groupe va chaque transition. On repartitionne les groupes selon les patterns des transitions en terme de groupes. On répète ce processus en utilisant cette nouvelle partition, et on réitère jusqu'à ce qu'on arrive à ne plus pouvoir partitionner. Les groupes restants forment l'automate minimal. On appelle souvent les itérations les étapes.

Description du processus de partitionnement itératif Soit un automate fini déterministe complet $D = (Q, i, T, E)$.

Résultat à obtenir : Un automate fini déterministe complet D' qui reconnaît le même langage que D et qui a aussi peu d'états que possible.

Méthode Construire une partition initiale Θ_0 de l'ensemble des états avec deux groupes : les états terminaux T et les états non-terminaux $Q - T$.

Procédure applicable à la partition courante Θ_i, à commencer par Θ_0 :

```
POUR chaque groupe G de Θi  FAIRE
début
partitionner G en sous-groupes de manière que deux états e et t de G
soient dans le même sous-groupe ssi pour tout symbole a ∈ A, les états e
et t ont des transitions sur a vers des états du même groupe de Θi ;
/* au pire, un état formera un sous-groupe par lui-même */
remplacer G dans par tous les sous-groupes ainsi formés ;
on obtient la partition de l'étape suivant, Θi+1 ;
fin
Si Θi+1 = Θi   alors Θfinal = Θi et continuer à l'étape 4.
Sinon, répéter l'étape (2) avec Θi+1 comme partition courante.
```

Choisir un état dans chaque groupe de la partition Θ_{final} en tant que représentant de ce groupe.

Les représentants seront les états de l'automate fini déterministe réduit D'.

Soit e un état représentatif, et supposons que dans D il y a une transition $e \xrightarrow{a} t$.

Soit r le représentant du groupe de t (r peut être égal à t).

Alors D' a une transition de e vers r sur a ($e \xrightarrow{a} r$).

L'état initial de D' est le représentant du groupe qui contient l'état initial i de D ; les états terminaux de D' sont des représentants des états de T.

Pour tout groupe G de Θ_{final}, soit G est entièrement composé d'états de T, soit G n'en contient aucun.

Remarque

En réalité, il est parfois plus facile, au lieu de choisir un représentant, marquer les groupes finaux comme tels, ou bien les renommer par, par exemple, A, B, C ... , et remplacer dans les transitions tout état par le nom de son groupe :
Si $e \xrightarrow{a} r$, $e \in A$, $r \in B$ où les groupes $A, B \in \Theta_{final}$, alors dans l'automate minimisé on a $A \xrightarrow{a} B$ où maintenant on considère A et B comme états de l'automate minimisé. Cette remarque deviendra tout à fait claire à la fin de l'exemple suivant.

Exemple 1

Minimisons l'automate déterministe complet :

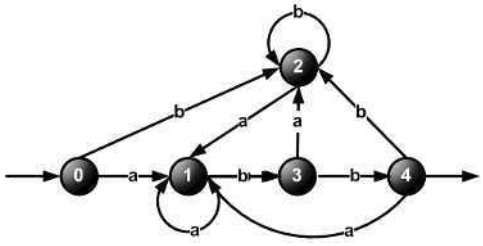

Figure 5.11

décrit par le tableau de transitions suivant :

Tableau 5.8

	État	États résultant	
		en lisant a	en lisant b
(initial)	0	1	2
	1	1	3
	2	1	2
	3	2	4
(terminal)	4	1	2

L'unique état terminal est l'état 4.
Donc, la partition initiale : $\Theta_0 = \{(0,1,2,3),(4)\}$.
Notons les groupes non terminal et terminal : $I = \{(0,1,2,3)\}$ et $II = \{(4)\}$.
On ne peut pas, évidemment, essayer de séparer le groupe II qui consiste déjà en un seul état.
On regarde donc dans quel groupe tombent les transitions à partir des états du groupe I :

Tableau 5.9

État	Groupes résultant	
	en lisant a	en lisant b
0	I	I
1	I	I
2	I	I
3	I	II

Donc, le groupe I se sépare en deux, et on a $\Theta_1 = \{(0,1,2),(3),(4)\}$.
Notons les groupes (en recyclant la notation) : $I = \{(0,1,2)\},\ II = \{(3)\},\ III = \{(4)\}$.
Maintenant essayons de séparer le groupe I en regardant où tombent les transitions à partir de ses états dans les termes de la séparation $\Theta_1 = \{(0,1,2),(3),(4)\}$.

Tableau 5.10

État	Groupes résultant	
	en lisant a	en lisant b
0	I	I
1	I	II
2	I	I

Donc, le groupe I se sépare en deux, et on a $\Theta_2 = \{(0,2),(1),(3),(4)\}$.
Notons les groupes (en recyclant la notation) :

$$I = \{(0,2)\},\ II = \{(1)\},\ III = \{(3)\},\ IV = \{(4)\}.$$

Essayons de séparer le groupe I en regardant où tombent les transitions à partir de ses états dans les termes de la séparation $\Theta_2 = \{(0,2),(1),(3),(4)\}$.

Tableau 5.11

État	Groupes résultant	
	en lisant a	en lisant b
0	II	I
2	II	I

Le groupe I ne se sépare pas, $\Theta_3$reste identique à la séparation de l'étape précédent :

$$\Theta_3 = \Theta_2 = \{(0,2), (1), (3), (4)\}.$$

Fin de la procédure itérative de séparation.
Voici les itérations qu'on a effectuées :
Étape 1 : $\Theta_{courant} = \{(0,1,2,3), (4)\}$

$$\Theta_{new} = \{(0,1,2), (3), (4)\}$$

Étape 2 : $\Theta_{courant} = \{(0,1,2), (3), (4)\}$

$$\Theta_{new} = \{(0,2), (1), (3), (4)\}$$

Étape 3 : $\Theta_{courant} = \{(0,2), (1), (3), (4)\}$

$$\Theta_{new} = \{(0,2), (1), (3), (4)\} = \Theta_{final}$$

Passons aux représentants :

Tableau 5.12

Groupe	Représentant
I	0 ou 2
II	1
III	3
IV	4

Il devient clair maintenant qu'on peut marquer les états de l'automate minimisé soit par un représentant, soit par les chiffres I, II, III, IV suivant le groupe du dernier étape, soit par le contenu du groupe (dans ce cas, l'état initial sera marqué (0,2) ; cette dernière solution est pratique tant que le groupe consiste en peu d'états).
L'automate minimisé :

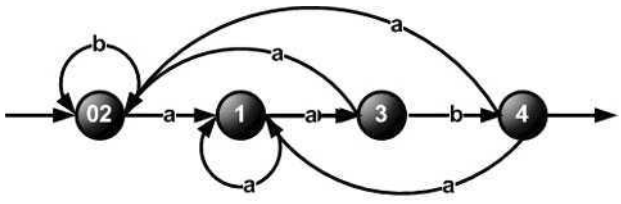

Figure 5.12

On peut maintenant poser la question suivante : pourquoi les états 0 et 2 sont resté ensemble après la minimisation ? Qu'y a-t-il de spécial concernant ces deux états par rapport aux autres ? Regardons à nouveau l'automate initial :

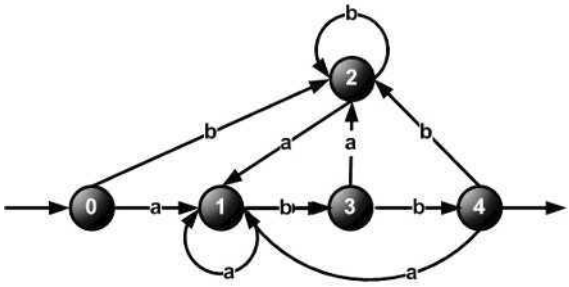

Figure 5.13

et introduisons la terminologie suivante :
on dit que la chaîne w distingue l'état s de l'état t si quand on commence dans l'état s et lit w on arrive dans un état terminal, et si on commence dans t et lit w on arrive dans un état non terminal ou vice-versa.

Ici, tous les états peuvent être distingués par une chaîne ou une autre, sauf les états 0 et 2. C'est facile à voir : et à partir de 0, et à partir de 2 on arrive en 1 en lisant un nombre quelconque (y compris zéro) de b et un a; puis, comme on est dans le même état (1), il ne nous reste que les mêmes chaînes pour arriver à l'état final (ici, il y en a un seul). Donc, les états non distinguables se fondent dans un même état (en l'occurrence (0,2)) de l'automate minimisé.

En fait, on peut montrer que l'algorithme de minimisation qu'on a expliqué, est basé sur la recherche des tous les groupes qui peuvent être distingués par une chaîne ou $I = \{1\}$, $T = \{1,2\}$ une autre.

Exemple 2

Voici un automate déterministe complet avec deux états terminaux, avec l'alphabet A $= \{0, 1\}$:

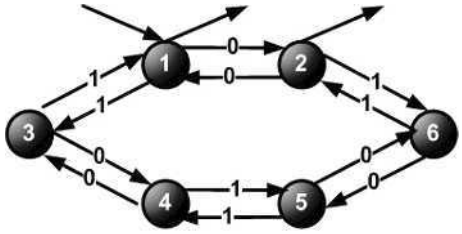

Figure 5.14

La procédure de minimisation donne :

$I = \{1\}, \quad T = \{1,2\}$

$\Theta_0 = \{(1,2),(3,4,5,6)\} = (I, II\}$
$\Theta_1 = (\{(1,2).(3,6),(4,5)\} = \{I, II, III\}$ (en recyclant les chiffres romains)
$\Theta_2 = \Theta_1$
Voici l'automate minimisé :

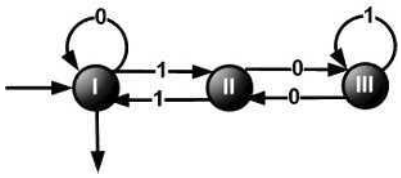

Figure 5.15

5.3.2 Automate asynchrone

Souvenons-nous que jusqu'à ce point nous avons évité **le mot vide**.

Un automate fini dans lequel certaines flèches sont étiquetées par le mot vide (« ***ε-transitions*** ») s'appelle **automate asynchrone**. L'ensemble des flèches vérifie donc $E \subset Q \times \left(A \cup \{\varepsilon\}\right) \times Q$.

Proposition : pour tout automate asynchrone, il existe un automate fini ordinaire (c.à.d. sans ε-transitions) qui reconnaît le même langage.

Comme tout automate non déterministe est équivalent à un automate déterministe, il en suit que **tout automate asynchrone est équivalent à un automate déterministe**.

Il existe un algorithme de suppressions de ε-transitions.

Pour un automate asynchrone $C = (Q, I, T, E)$ avec $E \subset Q \times \left(A \cup \{\varepsilon\}\right) \times Q$ nous allons construire un automate $B = (Q, I, T, F)$ qui ne diffère de C que par l'ensemble de ses flèches et ce, suivant une règle bien déterminée :

- par définition on a $(p.a.q) \in F$ s'il existe un chemin

$$c : p_0 \xrightarrow{\varepsilon} p_1 \xrightarrow{\varepsilon} \ldots \xrightarrow{\varepsilon} p_n \xrightarrow{a} q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} \ldots \xrightarrow{\varepsilon} q_m$$

$$\text{et } p_0 = p \wedge q_m = q$$

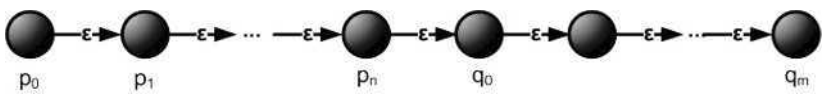

Figure 5.16

- Le nombre de ε-transitions à droite ou à gauche de la transition $p_n \xrightarrow{a} q_0$ peut être nul (dans ce cas $p_n = p_0$ou $p_n = p_0$).

Exemple

a) Prenons un automate asynchrone :

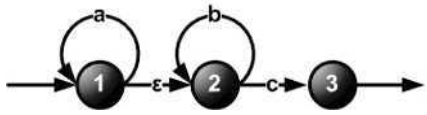

Figure 5.17 **(A1)**

Ici, suivant le raisonnement précédent, il existent les chemins : $1a1,\ 1a2,\ 1b2,\ 1c3,\ 2b2,\ 2c3$.

b) Donc cet automate est équivalent à l'automate non déterministe suivant :

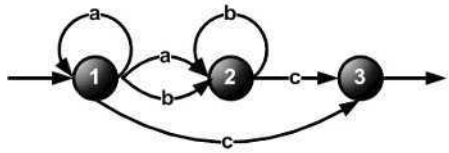

Figure 5.18 **(A2)**

c) Maintenant on peut le déterminiser :

Tableau 5.13

État	États résultant		
	en lisant a	en lisant b	en lisant c
1	1,2	2	3
1,2	1,2	2	3
2	–	2	3
3	–	–	–

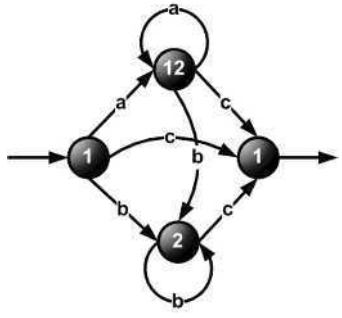

Figure 5.19 **(A3)**

ou bien, en ajoutant l'état poubelle :

Tableau 5.14

État	États résultant		
	en lisant a	en lisant b	en lisant c
1	1,2	2	3
1,2	1,2	2	3
2	P	2	3
3	P	P	P
P	P	P	P

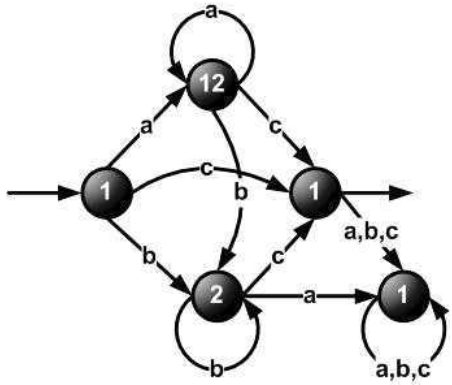

Figure 5.20 (A4)

d) Mais en réalité, il est plus simple de se passer de l'étape (b) et construire un automate déterministe directement à partir d'un automate asynchrone.
Redessinons l'automate (A1) :

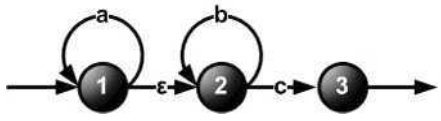

Figure 5.21 (A1)

L'état initial de l'automate déterministe correspondant à (A1) est (1,2), car en lisant le mot vide on arrive et en 1, et en 2. Continuons à déterminiser en marquant dans les colonnes correspondant à la lecture de chaque caractère, tous les états où on arrive après la lecture du caractère en question, c.à.d. non seulement l'état où mène la flèche étiquetée par ce caractère (disons, a), mais aussi tous les états où on peut arriver en lisant $a\varepsilon$, $a\varepsilon\varepsilon$ etc. Dans notre cas particulier, il n'y a pas de telles transitions, mais il faut systématiquement en tenir compte.

Tableau 5.15

État	États résultant		
	en lisant a	en lisant b	en lisant c
1,2	1,2	2	3
2	–	2	3
3	–	–	–

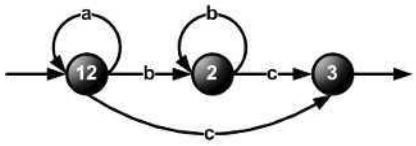

Figure 5.22 **(A5)**

On voit que l'automate (A5) n'est pas le même que l'automate (A3) ou (A4) ! Pourquoi ?

Parce que l'automate déterministe reconnaissant un certain langage n'est pas unique, ce n'est qu'en minimisant qu'on arrive à un automate unique. En minimisant (A4) et (A5), on doit obtenir la même chose.

Minimisons (A4) donc on rappelle la table de transitions :

Tableau 5.16

État	États résultant		
	en lisant a	en lisant b	en lisant c
1	1,2	2	3
1,2	1,2	2	3
2	P	2	3
3	P	P	P
P	P	P	P

Ici , on voit immédiatement que les états 1 et 1,2 ne se sépareront jamais !

$\Theta_0 = \{(1,(1,2),2,P),\ (3)\} = \{I,(3)\}$ (Utilisons une notation un peu plus intelligente pour ne pas modifier le sens des chiffres romains sans cesse)

Tableau 5.17

État	Groupes résultant		
	en lisant a	en lisant b	en lisant c
1	I	I	3
1,2	I	I	3
2	I	I	3
P	I	I	I

(5) se sépare en formant un groupe à part (c'est toujours le cas)

$\Theta_1 = \{(1,(1,2),2),\ (P),\ (3)\} = \{I,(P),(3)\}$

Tableau 5.18

État	Groupes résultant		
	en lisant a	en lisant b	en lisant c
1	I	I	3
1,2	I	I	3
2	P	I	3

(2) se sépare.
$\Theta_2 = \{(1,(1,2)),\ (2),\ (P),\ (3)\} = \{I,(2),(P),(3)\}$

Tableau 5.19

État	Groupes résultant		
	en lisant a	en lisant b	en lisant c
1	I	2	3
1,2	I	2	3

Le groupe (1,(1,2)) n'a aucun chance à se scinder en deux, car dès le début (1) et (1,2) ont les mêmes transitions. Donc on a terminé, et on obtient :

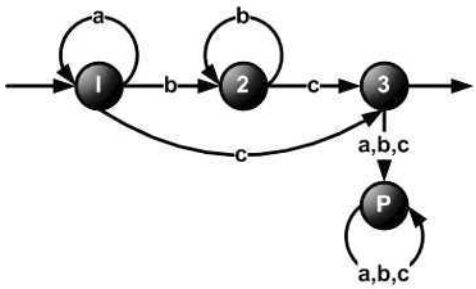

Figure 5.23

ce qui est le même automate que le résultat de compléter l'automate (A5) ! (en minimisant l'automate (A5), la seule chose qu'il reste à faire c'est de le compléter, autrement il est déjà minimal).

ÉNONCÉS DES EXERCICES

Exercice 5.1 Construire un automate fini dont les états correspondent aux situations de famille possibles d'une personne (célibataire, marié, divorcé, veuf) et dont les flèches correspondent aux changements de situation possible. Étiqueter ces flèches par M (mariage), D (divorce) et V (veuvage).

Exercice 5.2 Construire des automates finis qui reconnaissent les langages suivants :
- (a) L'ensemble des mots sur l'alphabet $\{0, 1\}$ dont l'avant dernier symbole est 0.

- (b) L'ensemble des mots sur l'alphabet $\{0, 1\}$ qui commencent et qui finissent par 0. (On considère que le mot consistant en un seul 0 vérifie cette condition).
- (c) L'ensemble des mots sur l'alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, .\}$ qui représentent en langage C des constantes numériques.
- (d) L'ensemble des mots sur l'alphabet $\{0, 1\}$ qui comportent au moins une fois le motif 10 et au moins une fois le motif 01 (ces deux motifs ne sont pas obligés d'être séparés l'un de l'autre – par exemple, la séquence 010 comporte les deux motifs).
- (e) $\{a^n b^m | n + m \text{ pair}\}$

Exercice 5.3 Construire un automate fini reconnaissant les entiers écrits en base deux divisibles par cinq.

Exercice 5.4 Construire un automate fini reconnaissant les lignes contenant des commentaires d'un programme écrit en langage C. La sortie de l'automate correspond à la fin du commentaire.
On prendra comme alphabet $\{/, *, ", a\}$ où a représentera tous les caractères autres que ", * et /.
Tout commentaire commence par /* et finit par */.
Le commentaire peut contenir des / et des *, mais il ne peut pas contenir le motif */, sauf à l'intérieur d'une chaîne qui commence par " et finit par " sans contenir d'autres ". En fait, toute chaîne de caractères placée entre les guillemets doubles, est déspécialisée ; donc le nombre de guillemets doubles doit être pair et avant le commentaire, et à l'intérieur du commentaire.
La séquence /*/ n'est pas considérée comme comportant et le motif /*, et le motif */.
Exemple d'un commentaire reconnu par l'automate : /*commentaire "/*" toto"*/" **/

Exercice 5.5 Quel est le langage reconnu par l'automate suivant ? Quels états de cet automate sont inutiles ?

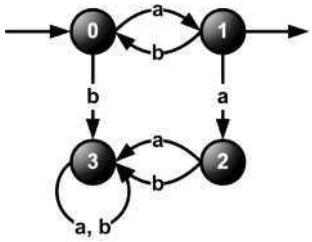

Figure 5.24

Exercice 5.6 Voici cinq automates notés A_n (ils n'ont rien à voir les uns avec les autres, ce n'est pas une séquence !). L'alphabet A = {a, b} sauf pour A_4 où il est {0,1}. Pour chacun de ces automates :
- Décrire $L(A_n)$ en langage ordinaire (sauf pour A_5).
- Caractériser A_n (dire s'il est accessible, coaccessible, émondé...)
- Calculer l'automate déterministe équivalent à A_n.

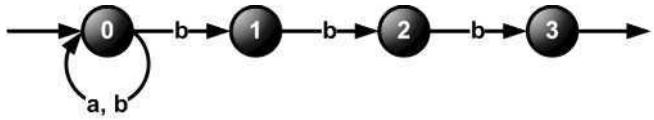

Figure 5.25 A1

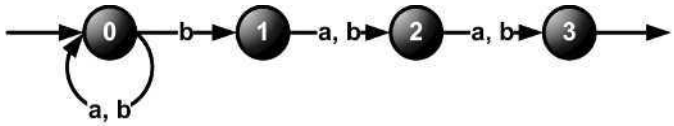

Figure 5.26 A2

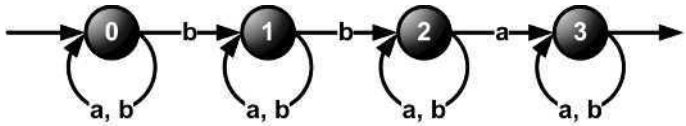

Figure 5.27 A3

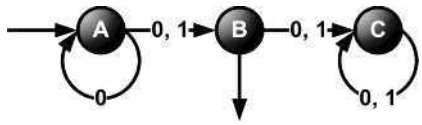

Figure 5.28 A4

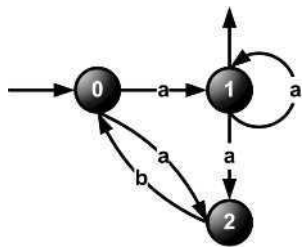

Figure 5.29 A5

(Ici, ne pas décrire $L(A_5)$ avant que l'automate ne soit déterminisé)