THÉORIES DES LANGAGES

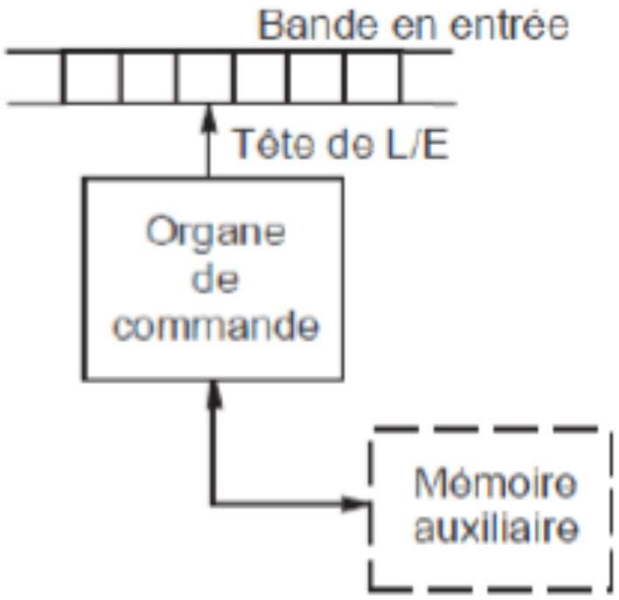
Mr,HEMIOUD
hemourad@yahoo,fr
Université de Jijel
Département d'informatique

Les automates finis

Les automates à états fini

- Un **automate** est une machine abstraite qui permet de lire un mot et de répondre à la question : "**un mot *w* appartient-il à un langage *L* ?**" par **oui** ou **non**.

- Un automate est composé de :



Formellement

- Un **alphabet** pour les mots en en entrée noté A ;
- Un ensemble non vide **d'états** noté Q;
- Un **état initial** noté $q_0 \in Q$;
- Un ensemble *non vide* d'états fin $\mathbf{Q_f} \in Q$;
- Une **fonction de transition** (permettant de changer d'état) notée **δ** (est une fonction totale de Q x *A* dans Q,

$$\delta(p,a)=q \ /p,q \in Q)$$

Configuration d'un automate

- Le fonctionnement d'un automate sur un mot se fait à travers un ***ensemble de configurations***.

- On appelle ***configuration d'un automate*** en fonctionnement les valeurs de ses différents composants, à savoir la ***position*** de la tête L/E, ***l'état*** de l'automate et éventuellement le ***contenu*** de la mémoire auxiliaire (lorsqu'elle existe).

Il existe deux configurations spéciales :

- La configuration ***initiale*** est celle qui correspond à l'état initial $\boldsymbol{q_0}$ et où la **tête** de L/E est *positionnée* sur le *premier* symbole du mot à lire.

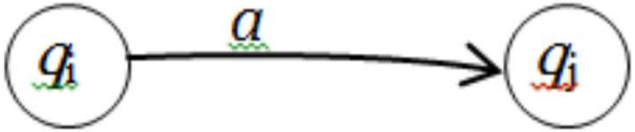
- Une configuration ***finale*** est celle qui correspond à un des états finaux $\mathbf{q_f}$ et où le mot a été *entièrement* lu.

Représentation graphique d'un automate

Un automate fini correspond à un graphe orienté

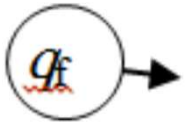
- L'état q_i

- la transition $\delta(q_i, a) = q_j$



- Etat initiale (q_0):

- Etats finaux (q_f)



Exemple : Représentation graphique de l'automate

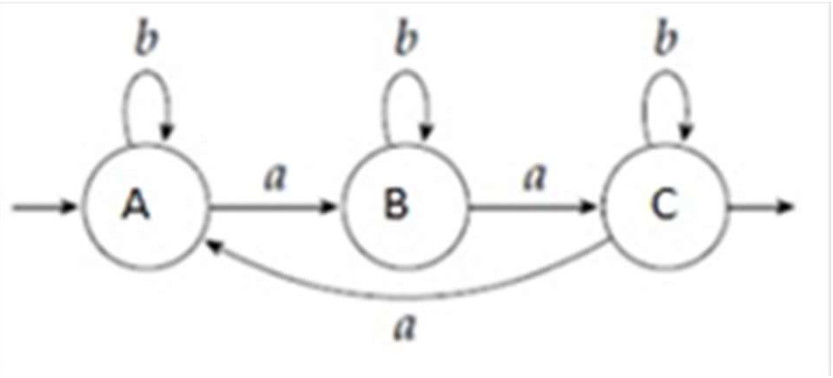
X=(**A**={a,b}, **Q**={A,B,C}, $\mathbf{q_0}$=A, $\mathbf{q_F}$={C} , **δ**) tel que :

δ (A, *a*)=B
δ (A, *b*)=A
δ (B, *a*)=C
δ (B, *b*)=B
δ (C, *b*)=C

Exemple : Représentation graphique de l'automate

X=(**A**={a,b}, **Q**={A,B,C}, $\mathbf{q_0}$=A, $\mathbf{q_F}$={C} , **δ**) tel que :

δ (A, *a*)=B
δ (A, *b*)=A
δ (B, *a*)=C
δ (B, *b*)=B
δ (C, *b*)=C



Mot reconnu par un automate

Un mot w est ***reconnu par l'automate*** A s'il existe une *configuration successive*

Configuration-initiale $(w) \vdash^*$ configuration-final(w)

$$(q_0, w) \vdash^* (q_f, \varepsilon)$$

- La relation $\vdash$ permet de formaliser la notion d'étape élémentaire de calcul d'un automate. Ainsi on écrira, pour $\boldsymbol{a}$ dans $\boldsymbol{A}$ et $\boldsymbol{v}$ dans $\boldsymbol{A}^*$:

$$(q, \underline{\boldsymbol{a}}v) \vdash (\delta(q, a); v)$$

Exemple : Représentation graphique de l'automate

X=(**A**={a,b}, **Q**={A,B,C}, $\mathbf{q_0}$=A, $\mathbf{q_F}$={C} , **δ**) tel que :

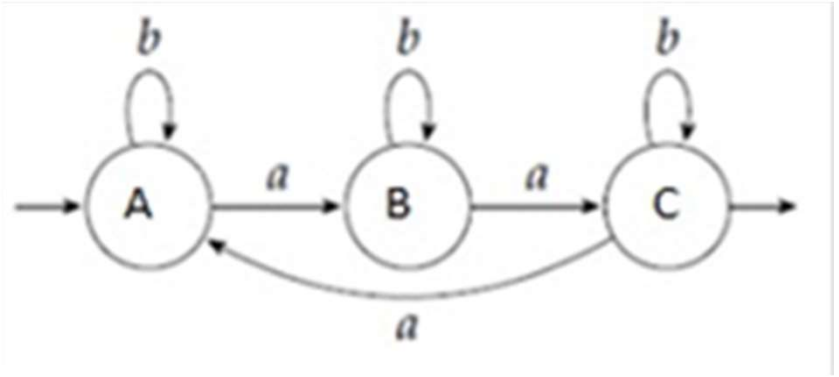
δ (A, *a*)=B

δ (A, *b*)=A

δ (B, *a*)=C

δ (B, *b*)=B

δ (C, *b*)=C



Donnez 2 mots qui sont reconnu par cet automate

Langage reconnu par un automate

On dit qu'un ***langage est reconnu par un automate*** X lorsque tous les mots de ce langage sont reconnus par l'automate on note $L(X)$

$L(X)=\{w \in A^* / \text{Configuration-initiale } (w) \vdash^* \text{configuration-final}(w)\}$

$L(X)=\{ w \in A^* / (q_0, w) \vdash^* (q, \varepsilon), \text{ avec } q \in Q_F \}$

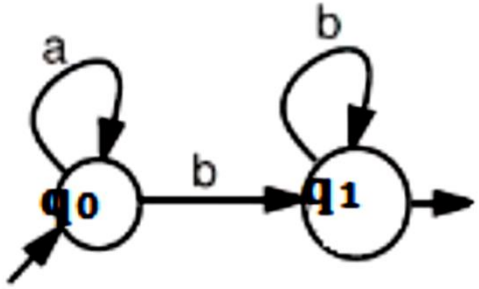
- **Exemple** : L'automate qui reconnaît les mots de la forme $\mathbf{a^n b^m (n \geq 0, m > 0)}$ est le suivant :

($\{a, b\}$, $\{ q_0, q_1\}$, q_0, $\{ q_1\}$, δ) tel que $\boldsymbol{\delta}$ est donnée par :

1) $\delta(q_0,a)=q_0$
2) $\delta(q_0,b)=q_1$
3) $\delta(q_1,b)=q_1$

ou par la table :

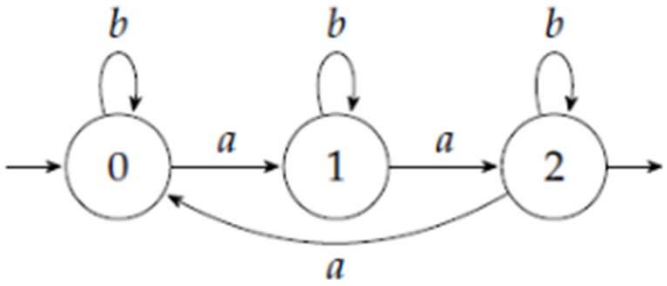
Etat	a	b
q_0	q_0	q_1
q_1	-	q_1



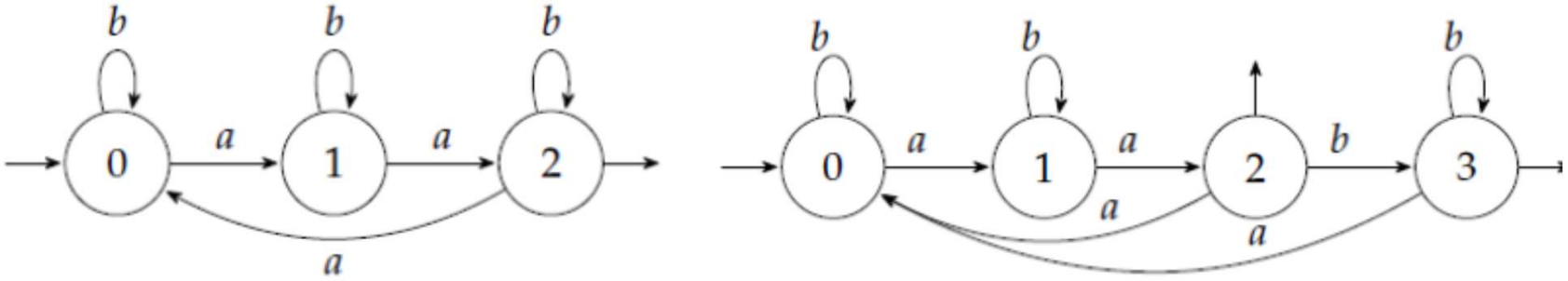
Exemple 2 :

La fonction de transition correspondant à ce graphe s'exprime matriciellement par

δ	a	b
0	1	0
1	2	1
2	0	2



- **Langage reconnaissable :** Un langage est reconnaissable s'il existe un automate fini qui le reconnaît.
- **Automates équivalents** : Deux *automates* finis X_1 et X_2 sont *équivalents* si et seulement s'ils reconnaissent le même langage ($L(X_1)=L(X_2)$)



Les deux automates reconnaissent le langage de tous les mots qui contiennent un nombre de ***a*** congru à 2 modulo 3.

- **Exemples**

Pour chacun des langages suivants, construire un automate d'états finis qui l'accepte

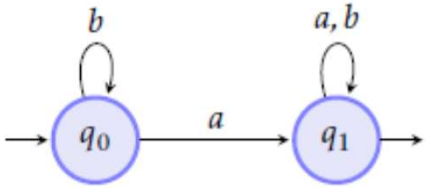
1. Le langage des mots contenant au moins une fois la lettre a :

2. Le langage des mots contenant au plus une fois la lettre a :

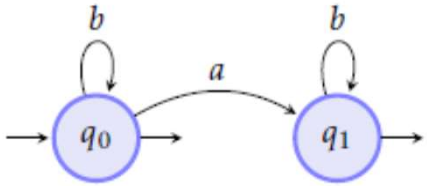
3. Le langage des mots contenant un nombre pair de fois la lettre a :

- **Exemples**

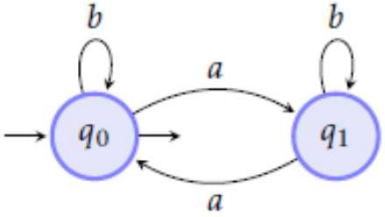
1. Le langage des mots contenant au moins une fois la lettre a :



1. Le langage des mots contenant au plus une fois la lettre a :



2. Le langage des mots contenant un nombre pair de fois la lettre a :

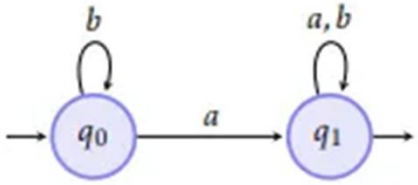


Exercice

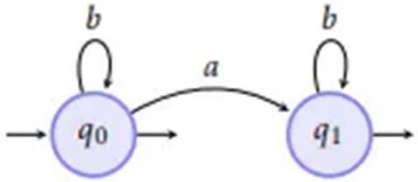
Pour chacun des langages suivants, construire un automate d'états finis qui l'accepte :

1. le langage dénoté par aba + bab.
2. Le langage des mots admettant aba pour facteur
3. le langage dénoté par $(aba)^*+ (bab)^*$.
4. $L = \{ w \in \{a, b\}^* / w = a^n b^m a$ ou $w = ba^n ; n, m \geq 1 \}$;
5. $L = \{w \in \{0, 1\}^* / w = 1(101)^n 00$ ou $w = 0(010)^n 11, n \geq 0 \}$

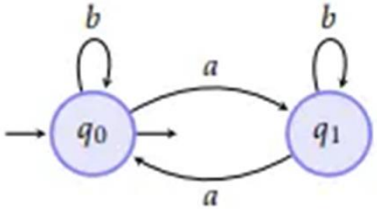
Le langage des mots contenant au moins une fois la lettre a :



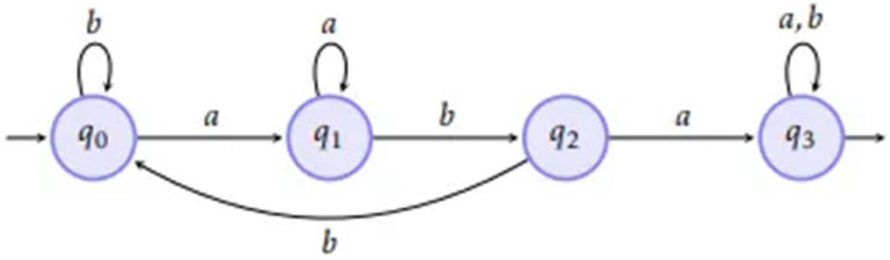
Le langage des mots contenant au plus une fois la lettre a :



Le langage des mots contenant un nombre pair de fois la lettre a :



Le langage des mots admettant aba pour facteur :



Passage de l'expression régulière vers l'automate

- Il existe de nombreuses stratégies pour construire de façon automatique un automate fini à partir d'une expression rationnelle

1. **Algorithme de Thompson**
2. Algorithme de Glushkov
3. La méthode des dérivées

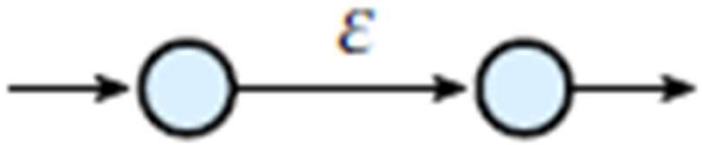
L'ALGORITHME DE CONSTRUCTION DE THOMPSON

- Plusieurs variantes sont possibles. Celle présentée ici est simple et surtout facile à implanter
- L'algorithme est dirigé par la syntaxe, c'est-à-dire qu'il utilise la structure syntaxique de l'expression rationnelle pour guider le processus de construction
- Il est récursif sur l'arbre syntaxique de l'expression rationnelle

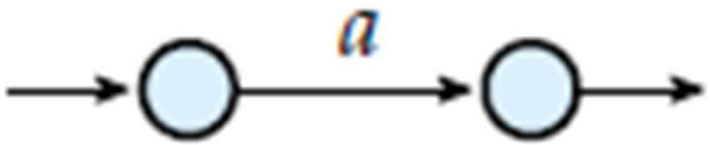
- la construction de Thompson « pure » qui présente quelques propriétés simples :
  - un unique état initial q_0
  - un unique état final q_F
  - exactement deux fois plus d'états que de symboles dans l'expression rationnelle (sans compter les parenthèses ni la concaténation, implicite).

Puisque les expressions rationnelles sont formellement définies de manière inductive (récursive) nous commençons par présenter les automates finis pour les « briques » de base

Automate de Thompson pour ; ϕ

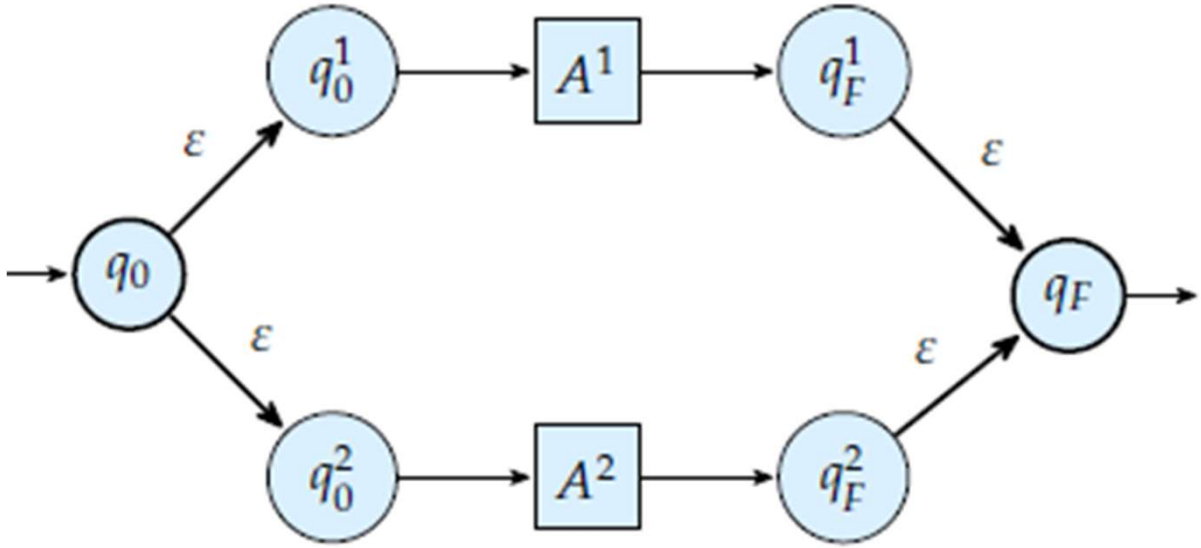


Automate de Thompson pour ; ε



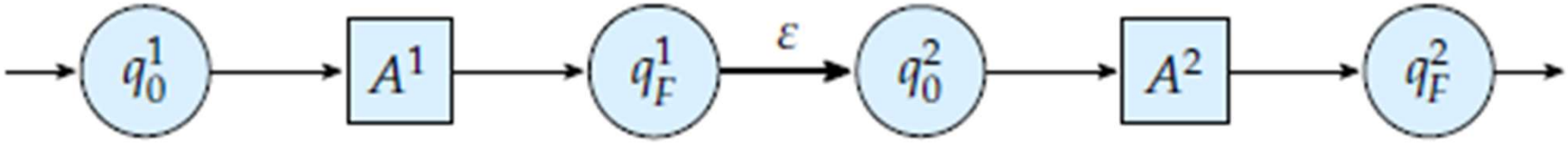
Automate de Thompson pour ; ***a***

Si A1 et A2 sont les automates de Thompson de e_1 et e_2



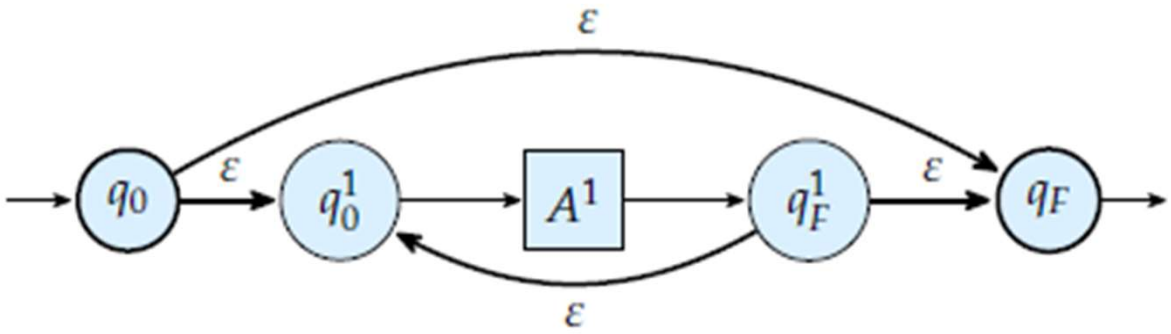
Automate de Thompson pour ; $\mathbf{e_1 + e_2}$

Si A1 et A2 sont les automates de Thompson de e_1 et e_2

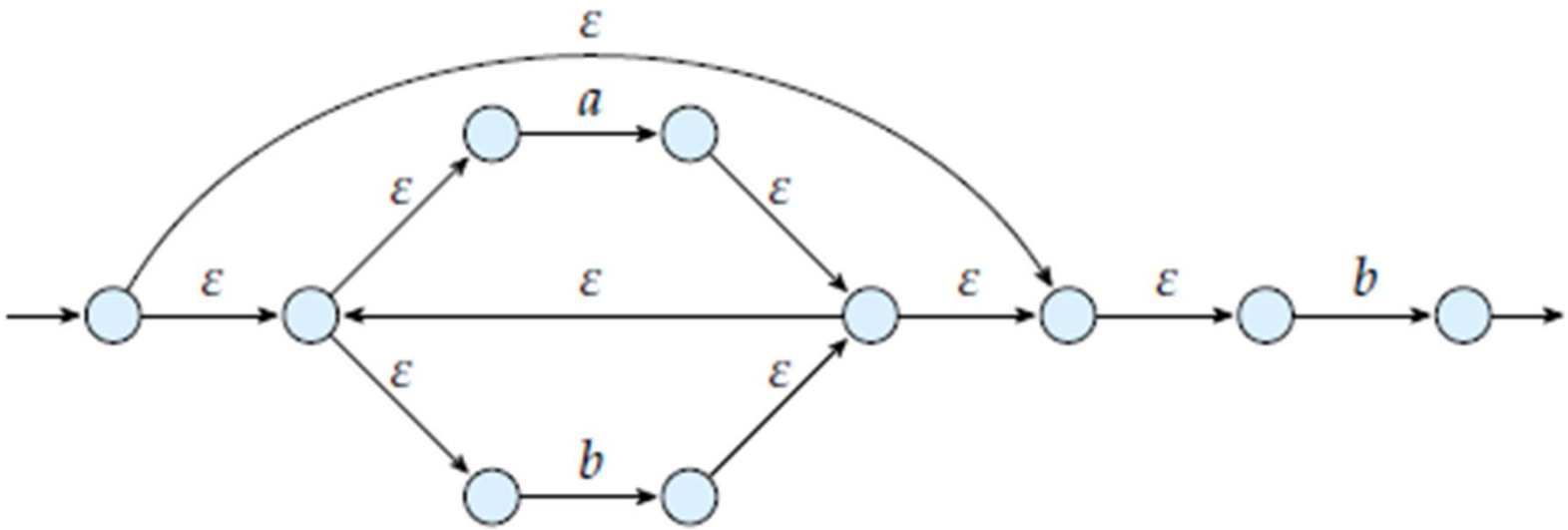


Automate de Thompson pour ; $\mathbf{e_1 e_2}$

Si A1 c'est l'automate de Thompson de e

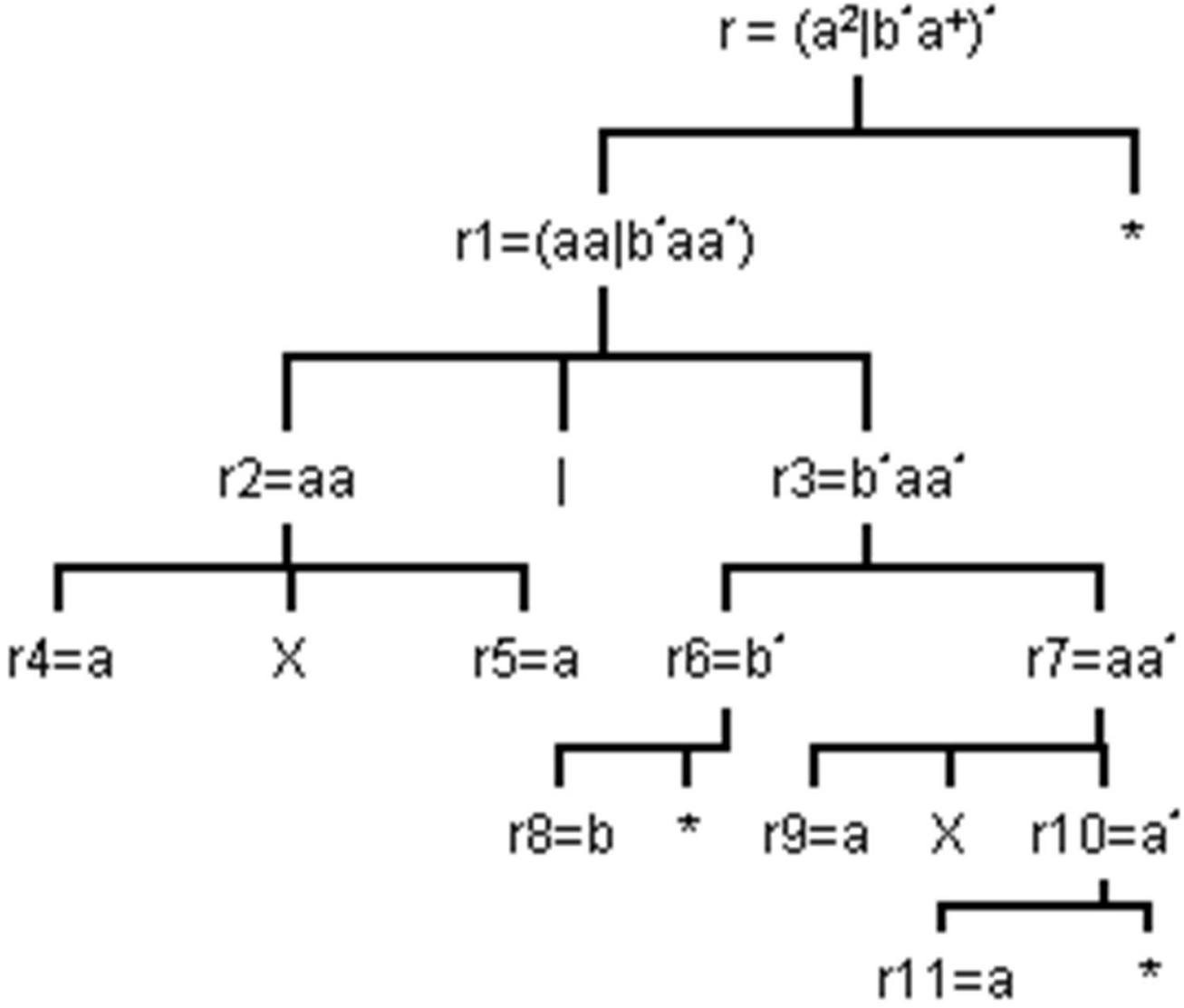


Automate de Thompson pour ; **e***



Automate de Thompson pour ; ***a+b***

Appliquons de cet algorithme sur l'expression régulière $(a^2 \mid b^*a^+)^*$.



Passage de l'expression régulière vers l'automate

- Il existe de nombreuses stratégies pour construire de façon automatique un automate fini à partir d'une expression rationnelle

1. Algorithme de Thompson
2. **Algorithme de Glushkov**
3. La méthode des dérivées

Passage de l'expression régulière vers l'automate (algorithme de Glushkov)

- Exemple: (ab + c)∗ab est linéarisée en
  (12 + 3)∗45.

	a	b	c
0	{1, 4}	∅	{3}
1	∅	{2}	∅
2	{1, 4}	∅	{3}
3	{1, 4}	∅	{3}
4	∅	{5}	∅
5	∅	∅	∅

Passage de l'expression régulière vers l'automate

- Il existe de nombreuses stratégies pour construire de façon automatique un automate fini à partir d'une expression rationnelle

1. Algorithme de Thompson
2. Algorithme de Glushkov
3. **La méthode des dérivées**

Passage de l'expression régulière vers l'automate

- **Dérivée d'un langage**

- **Définition** : Soit w un ***mot*** défini sur un alphabet A. On appelle **dérivée** de w par rapport à $u \in A^*$ le mot $v \in A^*$ tel que $w = uv$. On note cette opération par :

$$v = w \| _u .$$

- On peut étendre cette notion aux ***langages***. Ainsi la dérivée d'un ***langage*** par rapport à un mot $u \in A^*$ est le langage $\mathbf{L \| _u = \{v \in A^* \mid \exists w \in L : w = uv\}.}$
- **Exemple** :

– $L = \{a, ab, ba, aa\}$, $L \| _a = \{\varepsilon, b, a\}$, $L \| _b = \{a\}$, $L \| _{bb} = \emptyset$

Propriétés des dérivées

- $(\sum_{i=1}^{n} L_i)\|u = \sum_{i=1}^{n} (L_i\|u)$;
- $L.L'\|u = \begin{cases} (L\|u).L' + (L'\|u) & \text{si } \varepsilon \in L \\ (L\|u).L' & \text{sinon} \end{cases}$
- $(L^*)\|u = (L\|u)L^*$.

Méthode de construction de l'automate par la méthode des dérivées

- Soit ***R*** une expression régulière (utilisant l'alphabet A) pour laquelle on veut construire un AEF. L'algorithme suivant donne la construction de l'automate :

  1. Dériver ***R*** par rapport à chaque symbole de ***A*** ;

  2. Recommencer **1)** pour chaque nouveau langage obtenu jusqu'à ce qu'il n'y ait plus de nouveaux langages

  3. Chaque langage obtenu correspond à un état de l'automate. L'état ***initial*** correspond à *R*. Si ε appartient à un langage obtenu alors l'état correspondant est ***final*** ;

  4. si $\boldsymbol{Li \| a = Lj}$ alors on crée une ***transition*** entre l'état associé à *L*i et l'état associé à *L*j et on la décore par ***a***.

Exemple :

- $(a \mid b)^* a (a \mid b)^*$

Exemple :

- $(a \mid b)^* a (a \mid b)^*$

Passage de l'automate vers l'expression régulière

Passage de l'automate vers l'expression régulière

Soit X = (A,Q, q_0, Q_F, δ) un automate à états fini quelconque.

On note par $\boldsymbol{L_i}$ le langage reconnu par l'automate si son état initial était $\boldsymbol{q_i}$.

Par conséquent, trouver le langage reconnu par l'automate revient à trouver L_0 étant donné que la reconnaissance commence à partir de l'état initial $\boldsymbol{q_0}$. L'automate permet d'établir un système d'équations aux langages de la manière suivante :

– si $\delta(q_i, a) = q_j$ alors on écrit : $L_i = aL_j$;
– si $q_i \in Q_F$, alors on écrit : $L_i = \varepsilon$
– si $L_i = \alpha$ et $L_i = \beta$ alors on écrit : $L_i = \alpha \mid \beta$;

- Il suffit ensuite de résoudre le système précédant à des substitutions et en utilisant la règle suivante :

la solution de l'équation $\boldsymbol{L = \alpha L \mid \beta}$ **(ε ∉ α)** est le langage $\boldsymbol{L = \alpha^* \beta}$ (*Le lemme d'Arden*)

Le lemme d'Arden

Ce résultat dit que l'équation récursive suivante

$$X = LX + M$$

où X est le langage cherché et L et M sont deux langages connus, a une unique solution qui est :

$$X = L^*M:$$

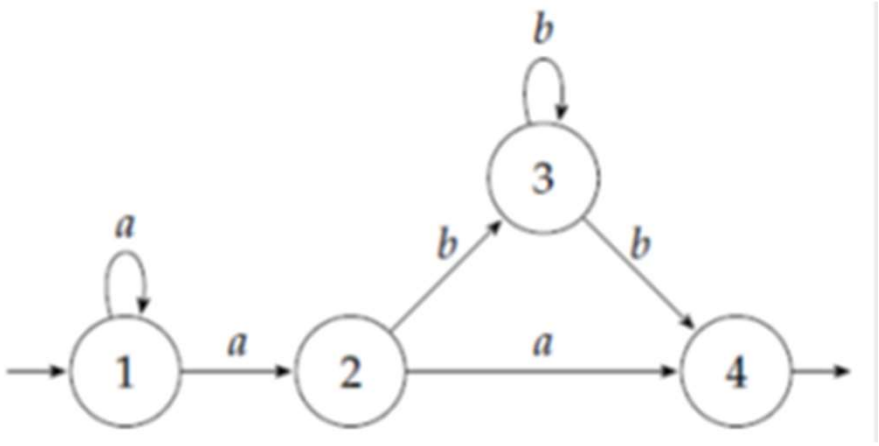
Attention : ici, on suppose que L ne contient pas le mot vide ($\varepsilon \notin L$).

Automate fini déterministe AFD

Automate fini déterministe AFD

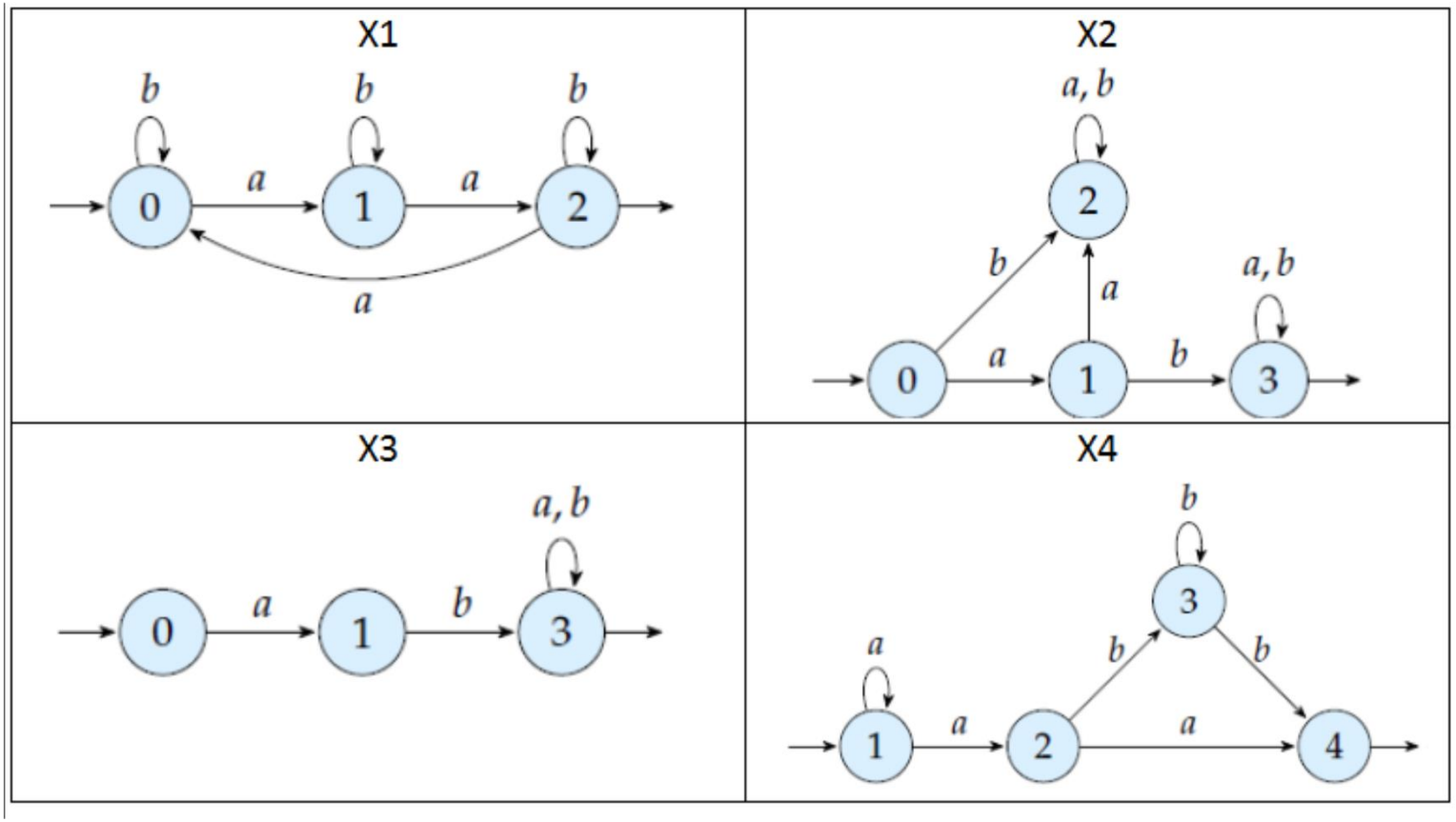
Un AEF (A,Q, q_0, Q_F, δ) est dit **déterministe** si les deux conditions sont vérifiées :

- $\forall q_i \in Q$, $\forall a \in X$, il existe au plus un état q_j tel que
  $\delta(q_i, a) = q_j$;
- L'automate ne comporte pas de ε-transitions.



Deux transitions sortantes de 1 sont étiquetées par a :

$$\delta(1, a) = \{1, 2\}$$


X1
b
b
b
a
a
0
1
2
a
X2
a, b
2
b
a
a, b
a
b
0
1
3
X3
a, b
a
b
0
1
3
X4
b
3
a
b
b
a
a
1
2
4

Théorème (Déterminisation)**.**

Pour tout AFN ***X***, on peut construire un AFD ***X'*** équivalent à ***X***.

De plus, si ***X*** a *n* états, alors ***X'*** a au plus 2^n états.

Algorithme : Déterminiser un AEF sans les ε-transitions

Principe : considérer des ensembles d'états plutôt que des états (dans l'algorithme suivant, chaque ensemble d'états représente un état du futur automate).

1- Partir de l'état initial $E^{(0)} = \{q_0\}$ (c'est l'état initial du nouvel automate) ;

2- Construire $E^{(1)}$ l'ensemble des états obtenus à partir de $E^{(0)}$ par la transition a :

$$E^{(1)} = \bigcup_{q' \in E(0)} \delta(q', a)$$

3- Recommencer l'étape 2 pour toutes les transitions possibles et pour chaque nouvel ensemble $E^{(i)}$;

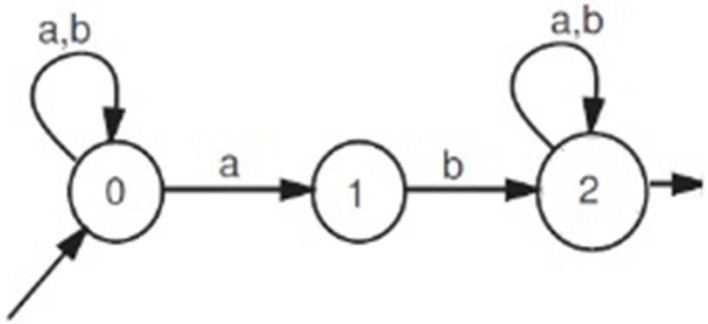
$$E^{(i)} = \bigcup_{q' \in E(i-1)} \delta(q', a)$$

4- Tous les ensembles contenant au moins un état final du premier automate deviennent finaux ;

5- Renuméroter les états en tant qu'états simples.

Déterminisation d'un AEF sans ε-transition

Exemple 1:

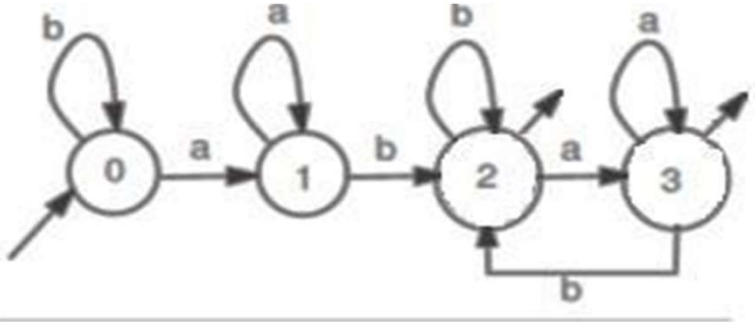


État	a	b
0	0,1	0
0,1	0,1	0,2
0,2	0,1,2	0,2
0,1,2	0,1,2	0.2

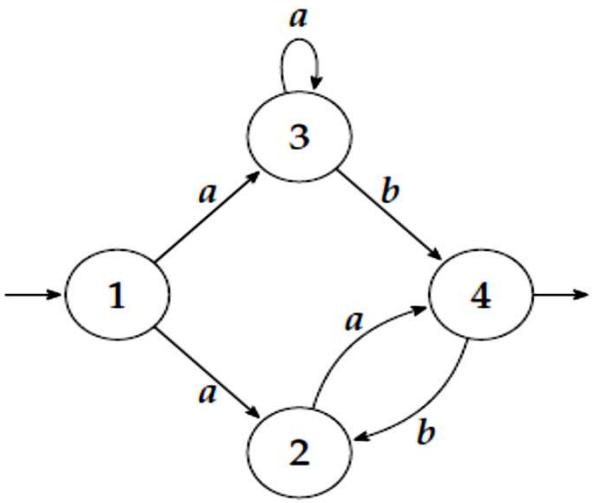
$\Longrightarrow$

État	a	b
0	1	0
1	1	2
2	3	2
3	3	2

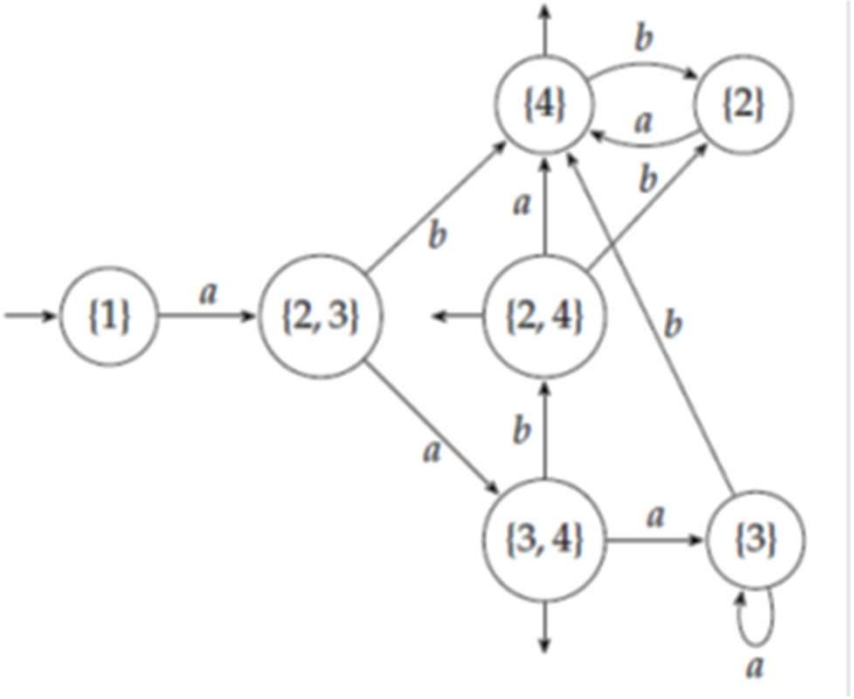
L'automate déterministe est



- Exemple 2

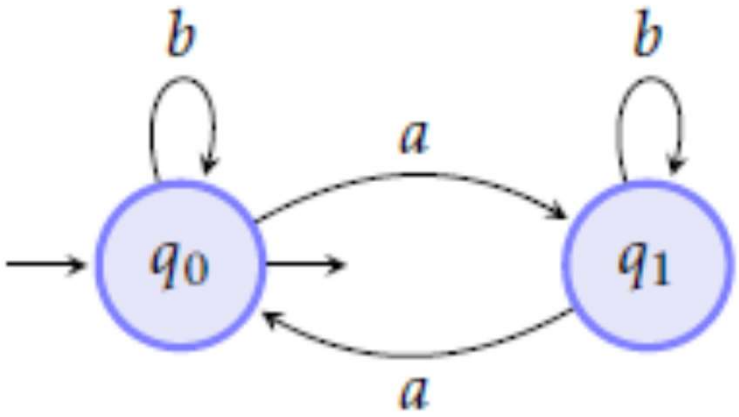


Le résultat de la déterminisation



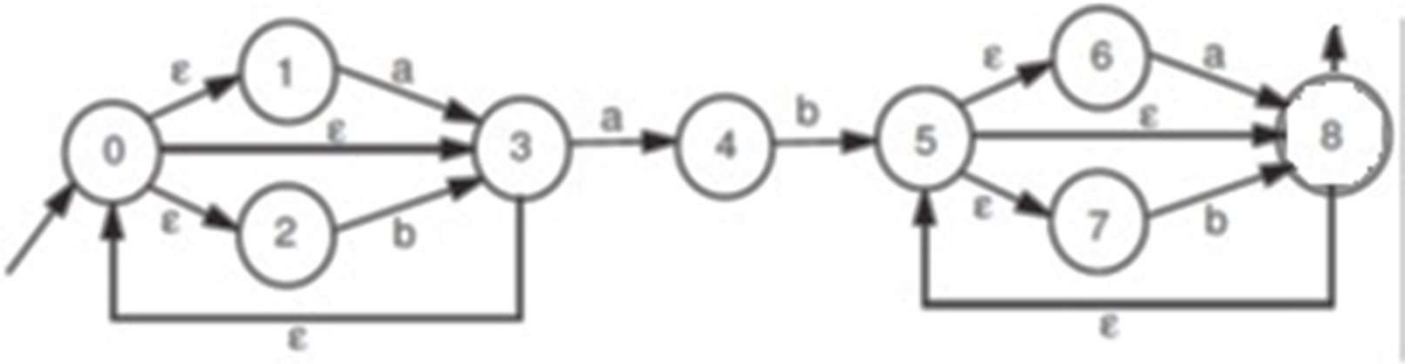
Exemple

Determiner le langage reconnu par l'automate suivant.



Déterminisation avec les ε-transitions

- On appelle **ε-fermeture** de E l'ensemble des états incluant, en plus de ceux de E, tous les états accessibles depuis les états de E par un chemin étiqueté par le mot ε.



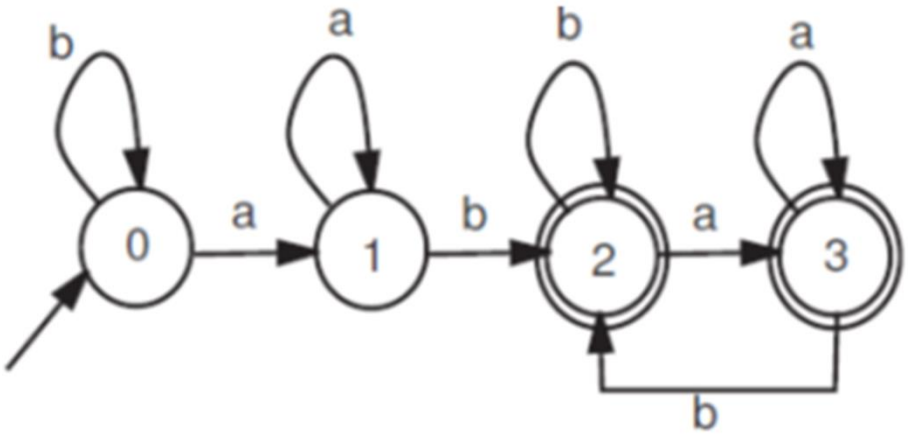
- ε-fermeture ({0}) = {0, 1, 2, 3}
- ε-fermeture ({1, 2}) = {1, 2}
- ε-fermeture ({3}) = {0, 1, 2, 3}

Exemple : Appliquons l'algorithme de détermination à l'automate précèdent

État	a	b
0,1,2,3	0,1,2,3,4	0,1,2,3
0,1,2,3,4	0,1,2,3,4	0,1,2,3,5,6,7,8
0,1,2,3,5,6,7,8	0,1,2,3,4,5,6,7,8	0,1,2,3,5,6,7,8
0,1,2,3,4,5,6,7,8	0,1,2,3,4,5,6,7,8	0,1,2,3,5,6,7,8

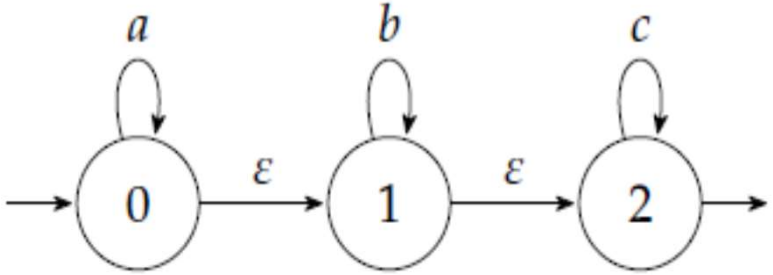
=>

État	a	b
0	1	0
1	1	2
2	3	2
3	3	2

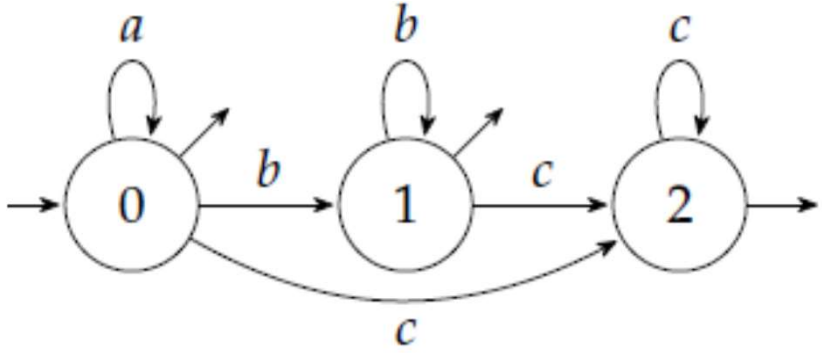


Exemple 2

- Un automate avec ε-transitions correspondant à a*b*c* ($a^n b^m c^l$ avec n,m,l≥0)



- L'automate déterministe équivalent



Minimisation d'un AEF déterministe

La minimisation s'effectue en éliminant les états dits ***inaccessibles*** et en *confondant* (ou fusionnant) les états reconnaissant le même langage.

- **Définition 1** : Un état est dit **inaccessible** s'il n'existe aucun chemin permettant de l'atteindre à partir de l'état initial.
  - →Donc les états inaccessibles sont improductifs, c'est-à-dire qu'ils ne participeront jamais à la reconnaissance d'un mot.
- **Définition 2** : Deux états q_i et q_j sont dits **β-équivalents** s'ils permettent d'atteindre les états finaux en utilisant les mêmes mots. On écrit alors : qi ß qj.
  - La relation ß-équivalence est donc dite une relation de **congruence**

Minimiser un AEF

La méthode de réduction d'un AEF est la suivante :

- *Nettoyer* l'automate en éliminant les états ***inaccessibles*** ;
- Regrouper les états ***congruents*** (appartenant à la même classe d'équivalence).

« Un état est dit ***inaccessible*** s'il n'existe aucun chemin permettant de l'atteindre à partir de l'état initial. »

- **Exemple 13** : Soit à minimiser l'automate suivant (les états finaux sont les états 1 et 2 tandis que l'état 1 est initial) :

État	a	b
1	2	5
2	2	4
3	3	2
4	5	3
5	4	6
6	6	1
7	5	7

1. La première étape consiste à éliminer les états **inaccessibles**, il s'agit juste de l'état 7.

Les étapes de détermination des classes de congruences sont les suivantes : A = {1, 2}, B = {3, 4, 5, 6} ;

2. $\delta(3, b) = 2 \in A, \delta(4, b) = 3 \in B$ ainsi il faut séparer 4 du reste de la classe B. Alors, on crée une classe C contenant l'état 4 ;
3. $\delta(3, b) = 2 \in A$, $\delta(5, b) = 6 \in A$ ainsi il faut séparer 5 du reste de la classe B. Mais inutile de créer une autre classe puisque

   $\delta(4, a) = 5 \in B$,

   $\delta(5, a) = 4 \in B$

   et $\delta(4, b) = 3 \in B$

   et $\delta(5, b) = 6 \in B$,

   il faut donc mettre 5 dans la classe C. C = {4, 5}

   et B = {3, 6} ;
4. Aucun autre changement n'est possible, alors on arrête l'algorithme.

- Le nouvel automate est donc le suivant (l'état initial est A) :

Etat	a	b
A	A	C
B	B	A
C	C	A

- L'automate obtenu est **minimal** et est **unique**,

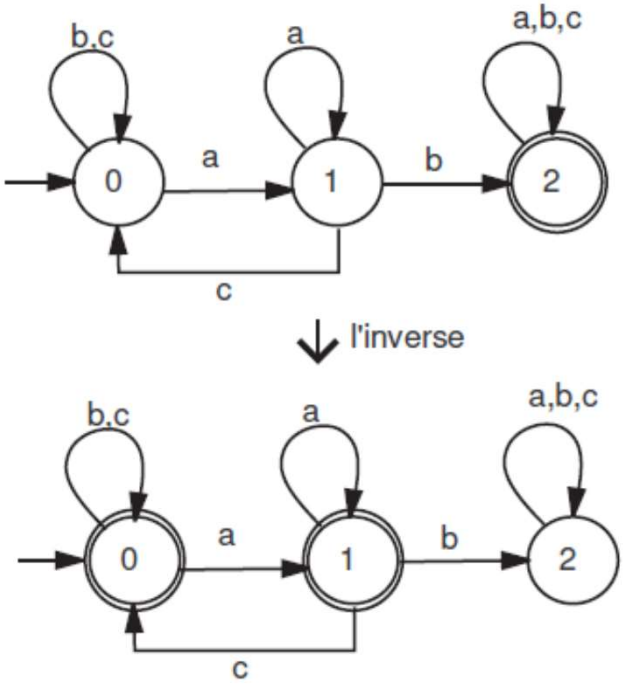
- **Opérations sur les automates**
- **Le complément**
- Soit A= (A,Q, q_0, Q_F, δ) un automate déterministe reconnaissant le langage ***L***.
- L'automate reconnaissant le **langage inverse** (c'est-à-dire L ou A* −*L*) est défini par le quintuplet (A,Q, q_0,Q − Q_F, δ) (*en d'autres termes, les états finaux deviennent non finaux et vice-versa*).

- cette propriété ne fonctionne que si l'automate est ***complet*** .

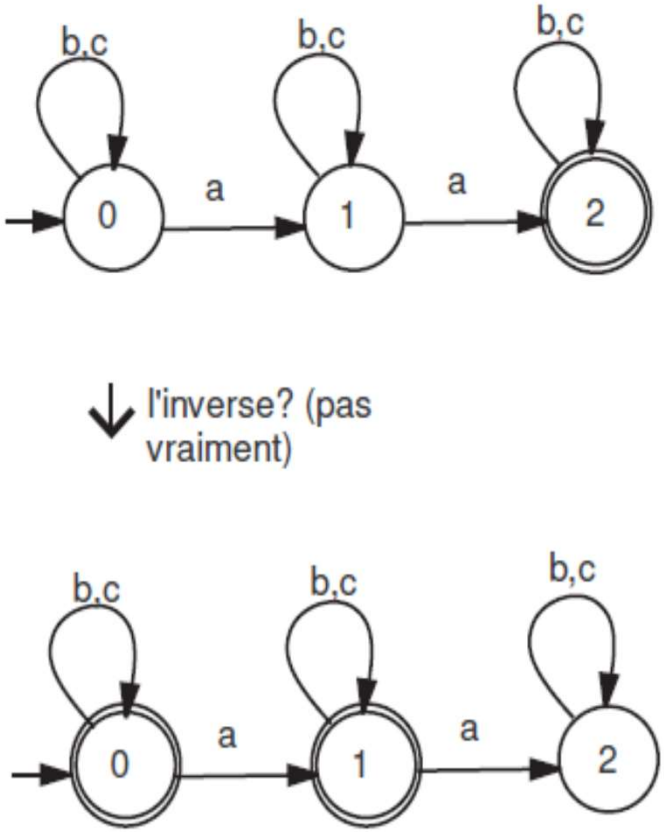
Exemple

:

- Soit le langage des mots définis sur l'alphabet {a, b, c} **<u>contenant le facteur ab</u>**.
- appliquer la propriété précédente pour trouver l'automate des mots qui **<u>ne contiennent</u>** pas le facteur **ab**

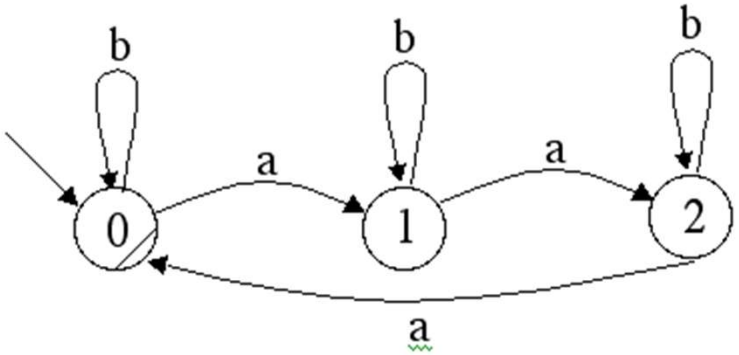


- Exemple

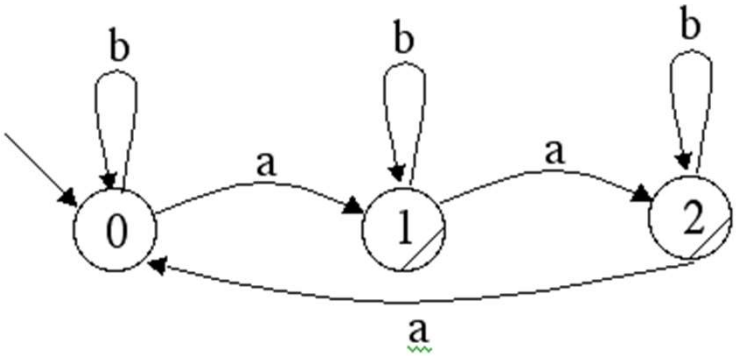


- L'automate obtenu reconnaît les mots contenant au plus 1 a

- On commence par L1 ayant un nombre de a multiple de 3. L1' est le complementaire de L1 (le nombre de a n'est pas multiple de 3).
- L1

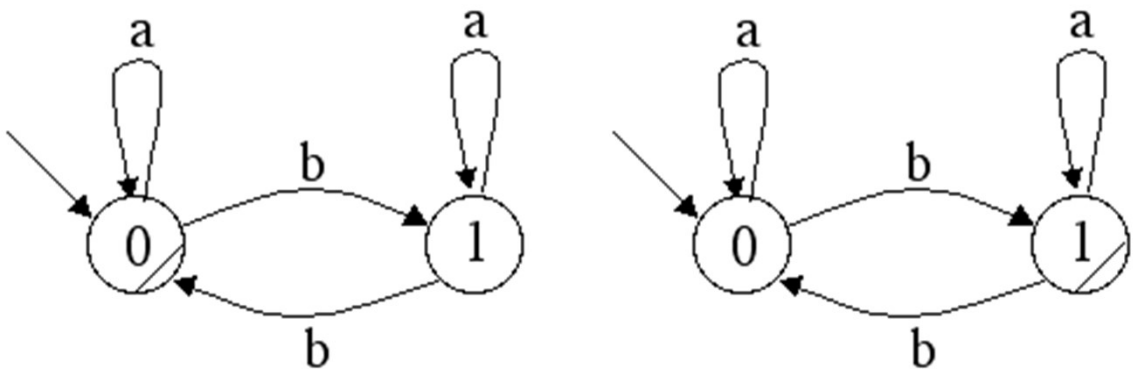


- L1'



Exemple

- Soit L2, le langage ou le nombre de b est pair et L2' son complémentaire. Nous avons les deux automates pour L2 et L2'
-



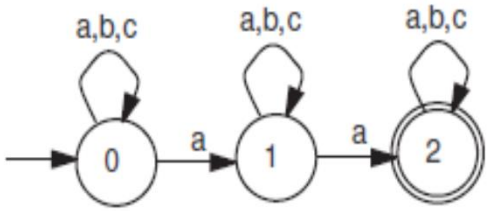
Produit d'automates

- Soit $X = (A, Q, q_0, Q_F, \delta)$ et $X' = (A', Q', q_0', Q'_F, \delta')$ deux automates à états finis.
- On appelle produit des deux automates X et X' l'automate $X''(A'', Q'', q_0'', Q_F'', \delta'')$ défini comme suit :
  - $A'' = A \cup A'$;
  - $Q'' = Q \times Q'$;
  - $q_0'' = (q_0, q_0')$;
  - $Q_F'' = Q_F \times Q_F'$;
  - $\delta((q, q') , a) = (\delta(q, a), \delta(q', a))$

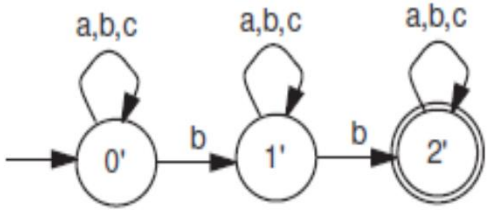
- $L(X*X') = L(X) \cap L(X')$

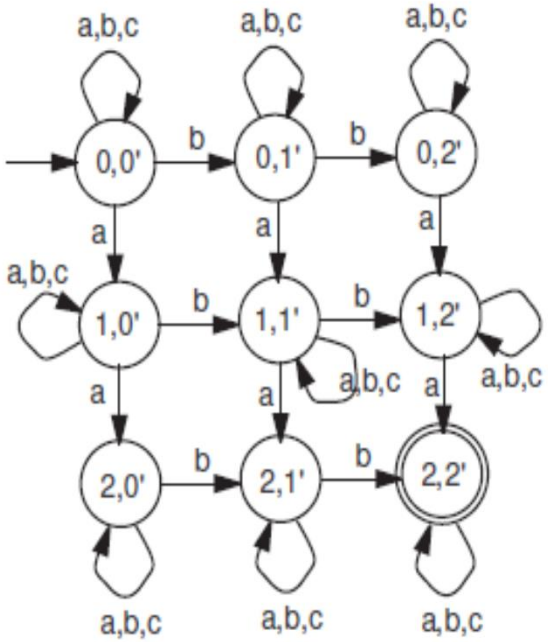
- Exemple

(1)


a,b,c
a,b,c
a,b,c
0
a
1
a
2

(2)


a,b,c
a,b,c
a,b,c
0'
b
1'
b
2'


a,b,c
a,b,c
a,b,c
0,0'
b
0,1'
b
0,2'
a
a
a
a,b,c
1,0'
b
1,1'
b
1,2'
a,b,c
a,b,c
a
a
a
2,0'
b
2,1'
b
2,2'
a,b,c
a,b,c
a,b,c

(3)

Le langage miroir

- Soit $X = (A, Q, q_0, Q_F, \delta)$ un automate reconnaissant le langage L(X).
- L'automate qui reconnaît le langage $(L(X))^R$ est reconnu par l'automate $X^R = (A, Q, Q_F, \{q_0\}, \delta^R)$ tel que : $\delta^R(q', a) = q$ si $\delta(q, a) = q'$.

- En d'autres termes, il suffit juste *d'inverser* les sens des arcs de l'automate et le statut initial/final des états initiaux et finaux