II. Les automates finis et les langages réguliers

Plan

II.1. Les automates finis non déterministes
II.2. Langage accepté par un automate fini
II.3. Rendre déterministe un automate fini non déterministe
II.4. Les langages réguliers LR
II.5. Minimisation d'un automate fini déterministe
II.6. Limites des automates finis

Motivation

***Problème**: est ce qu'une chaîne w appartient à un langage L?*

Quels sont les ressources nécessaires pour répondre à cette question.

Reconnaisseur d'un langage

Programme qui prend en entrée une chaîne *x* et répond par **oui** ou **non**. (*x* appartient ou non au langage)

On peut résonner pour les problèmes non pas comme une réponse Oui /non mais comme transformation des entrées en des sorties.

Utilisation des automates à états finis

Définition

Un AEF est un algorithme composé de :

1- une bande de lecture comportant des cases où chaque case contient un caractère de l'alphabet

2- une tête de lecture désignant une case particulière

Un AEF peut être considéré comme un mécanisme de calcul sans mémoire; ceci est réalisé à travers une table de transition permettant de spécifier la cible selon le caractère lu courant

3- un dispositif de contrôle pouvant prendre un nombre fini d'états

4- une table de transition précisant les changement d'états

- *Un ou +ieurs états initiaux spécifie(nt) l'entréé*
- *Les états finaux spécifient la sortie*

δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5

Les automates finis déterministes

Définition formelle: Automate fini

Un automate fini consiste en un quintuple de la forme $(Q, \sum, \delta, q_0, F)$ avec

Q est un ensemble **fini** d'états

X est un alphabet

$\boldsymbol{\delta}$ est une fonction de transition

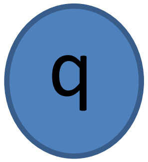
$\delta: Q \times X \rightarrow P(Q)$ (l'ensemble de tous les sous ensembles de Q)

q0 est l'état initial

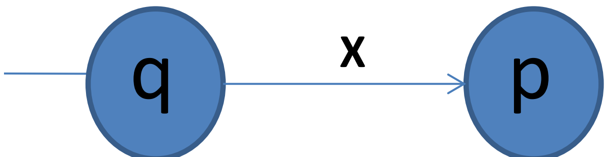
F est l'ensemble des états finaux $F \subseteq Q$

Représentation graphique

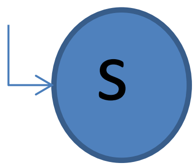
État



Transition sur x appartient àΣ



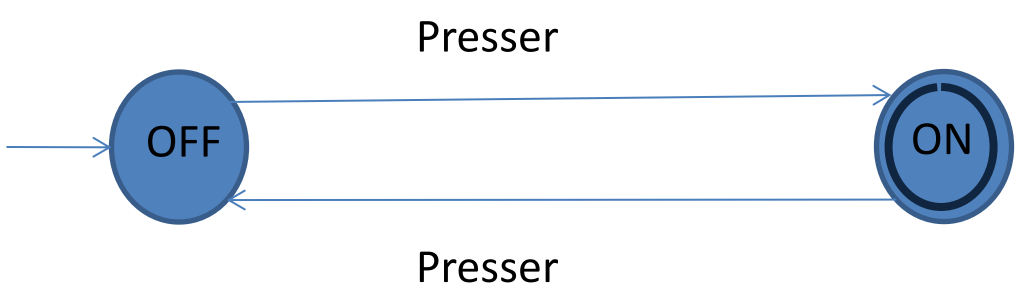
État initial



État final ou d'acceptation

II.1. Les automates finis déterministes

Exemple d'un automate fini



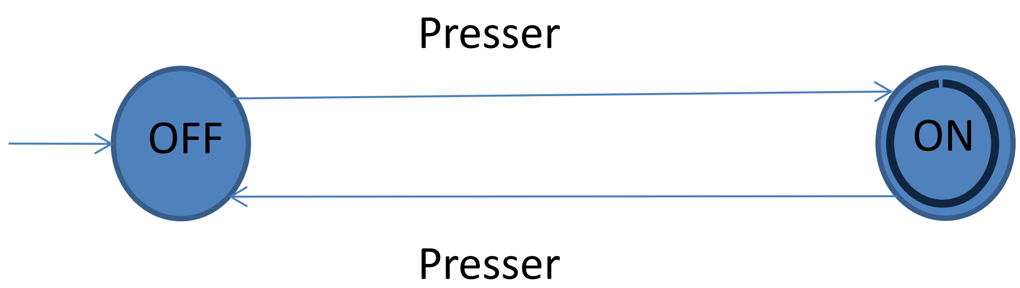
Cet automate reconnaît les séquences de presser qui nous mènent toujours à l'état final souhaité ON (lampe allumée)
En partant d'un état où la lampe est éteinte (OFF)

Q = {OFF, ON}
X = {presser}
$\mathbf{q_0}$ = OFF
F = {ON}

δ(OFF, presser) = {ON}
δ(ON, presser) = {OFF}

II.1. Les automates finis déterministes

Exemple d'un automate fini



La séquence:

presser presser presser est acceptée

Par ce que

•en partant de l'état off, avec un presser on passe à l'état ON

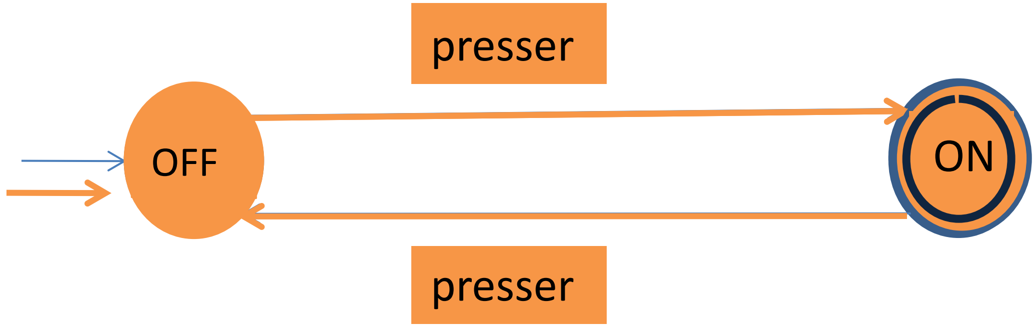
•En lisant le deuxième presser à partir de l'état ON, on revient à l'état OFF

•Le dernier presser lu nous mène de l'état OFF à l'état ON

Donc à la fin de la lecture des trois presser, trois transitions sont effectuées et la dernière nous mène à l'état ON donc la séquence est acceptée.

II.1. Les automates finis déterministes

Exemple d'un automate fini



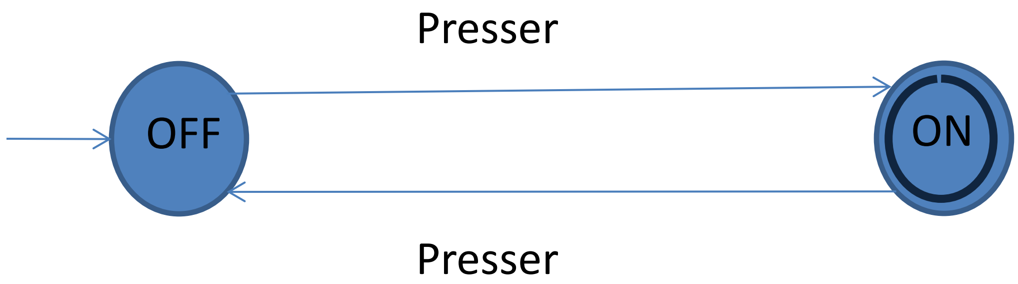
presser presser presser

Le mot est lu et l'état est final donc le mot est accepté

La séquence:
presser presser presser est acceptée
Par ce que
•en partant de l'état off, avec un presser on passe à l'état ON
•En lisant le deuxième presser à partir de l'état ON, on revient à l'état OFF
•Le dernier presser lu nous mène de l'état OFF à l'état ON
Donc à la fin de la lecture des trois presser, trois transitions sont effectuées et la dernière nous mène à l'état ON donc la séquence est acceptée.

II.1. Les automates finis déterministes

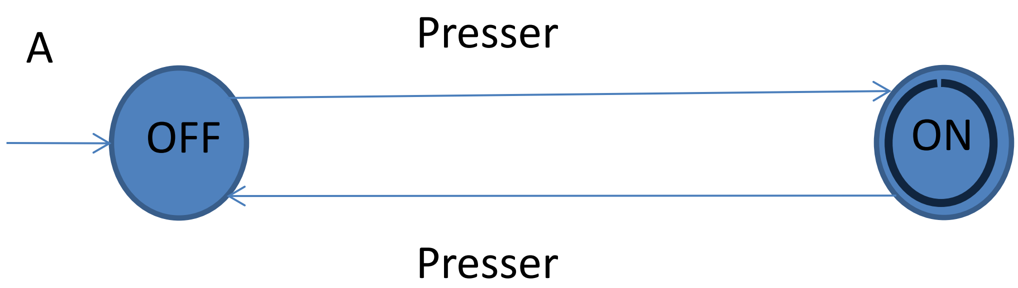
Exemple d'un automate fini



La séquence:
presser presser n'est pas acceptée par l'automate
Par ce que
•en partant de l'état off, avec un presser on passe à l'état ON
•En lisant le deuxième presser à partir de l'état ON, on revient à l'état OFF

II.1. Les automates finis déterministes

Exemple d'un automate fini



Le langage accepté par cet automate est l'ensemble des séquences de presser qui, partant de l'état initial OFF, après la lecture de tous les symboles de la séquence, on se trouve à l'état ON.

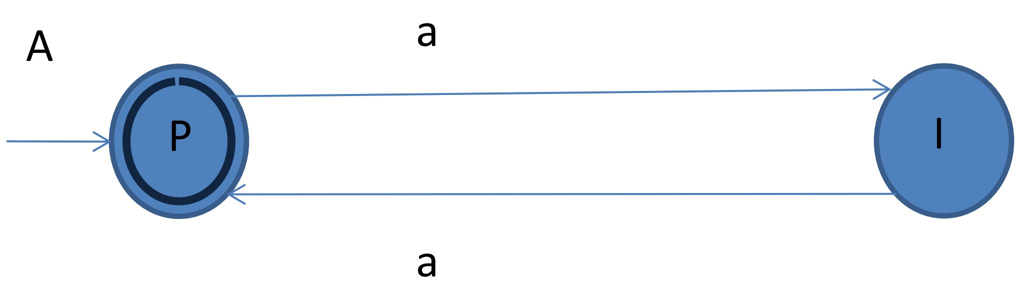
Ce langage (le langage accepté par l'automate A, L(A)) est l'ensemble des séquences de presser de longueur impaire.

$L(A) = \{\omega \in \{presser\}^* / \ |\omega| = 2k + 1, k \geq 0\}$

II.1. Les automates finis déterministes

Exemple d'un automate fini qui accepte
$\{\omega \in \{a\}^*/ \ |\omega| = 2k, k \geq 0\}$

(aa)*



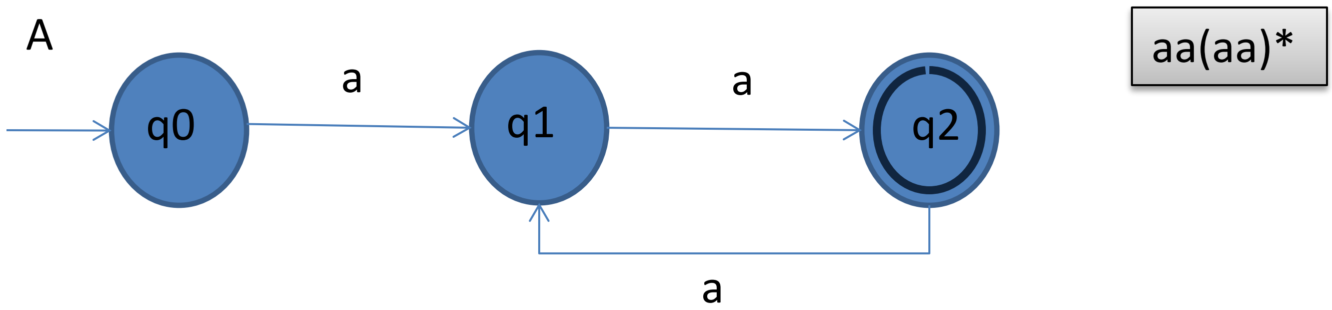
L'état initial devient un état final puisque le mot ε est accepté (pour k = 0).
En étant à l'état initial de l'automate, on peut ne rien lire (c-a-d lire 0 symboles donc lire ε) et on est déjà à un état final.

II.1. Les automates finis déterministes

Exercices :

1) Construire un automate fini qui accepte le langage suivant:

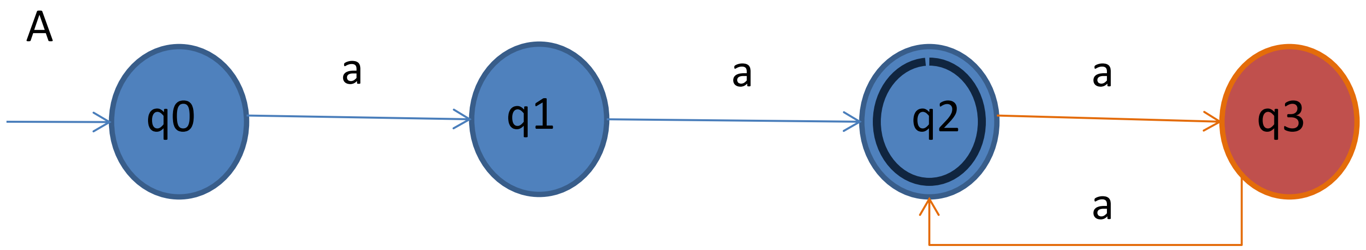
$\{\omega \in \{a\}^*/\ |\omega| = 2k,\ k > 0\}$



Le plus petit mot accepté est aa (On atteint l'état final après la lecture de aa). Après il faudra continuer à avoir un nombre paire de a. Donc quand on se trouve à l'état d'acceptation ou l'état final, il faudra continuer à lire des séquences de 2a pour revenir à l'état final.

II.1. Les automates finis déterministes

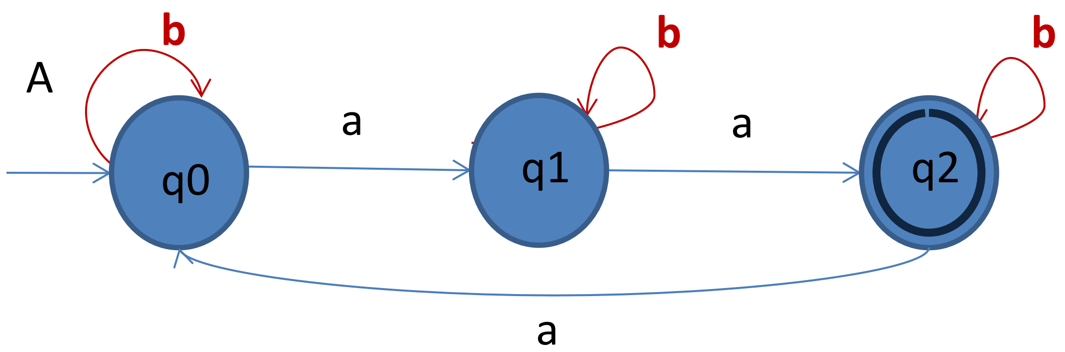
$\{\omega \in \{a\}^* / \ |\omega| = 2k, k > 0\}$ (autre solution)



Le plus petit mot accepté est aa (On atteint l'état final après la lecture de aa). Après il faudra continuer à avoir un nombre paire de a. Donc quand on se trouve à l'état d'acceptation ou l'état final, il faudra continuer à lire des séquences de 2a pour revenir à l'état final.

II.1. Les automates finis déterministes

2) Construire un automate fini qui accepte le langage suivant:
$\{a,b\}^*/\ |\omega|_a = 3k+2,\ k \geq 0\}$



b*ab*ab*(ab*ab*ab*)*

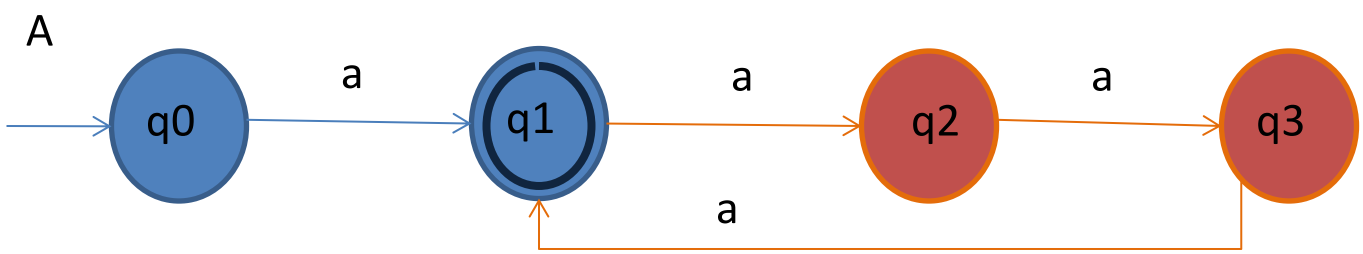
Le plus petit mot accepté est aa (On atteint l'état final après la lecture de aa). Après il faudra continuer à avoir un nombre paire de a. Donc quand on se trouve à l'état d'acceptation ou l'état final, il faudra continuer à lire des séquences de 2a pour revenir à l'état final. Les b on peut les lire à n'importe quel état et on reste au même état puisqu'ils n'engendrent pas une transitions contrairement à un a lu qui nous fait passer de q0 à q1 ou de q1 à q2 ou de q2 à q0 par ce qu'il faut les comptabiliser.

II.1. Les automates finis déterministes

3) Construire un automate fini qui accepte le langage suivant:

$\{\omega \in \{a\}^*/ \ |\omega| = 3k+1, k \geq 0\}$

a(aaa)*

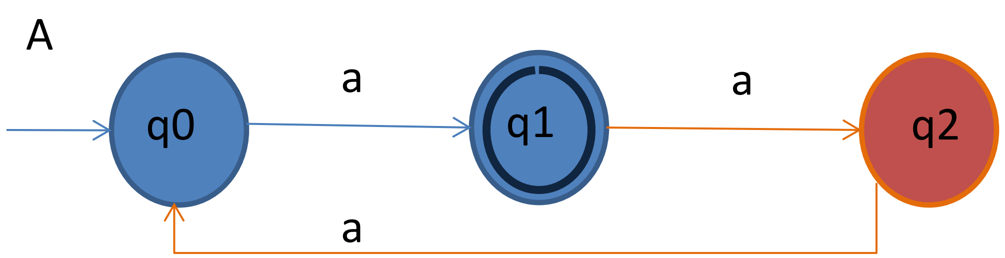


Le plus petit mot accepté est a (On atteint l'état final après la lecture de a). Après il faudra continuer à avoir un nombre multiple de 3a. Donc quand on se trouve à l'état d'acceptation ou l'état final, il faudra continuer à lire des séquences de 3a pour revenir à l'état final.

II.1. Les automates finis déterministes

$\{\omega \in \{a\}^*/\ |\omega| = 3k+1,\ k \geq 0\}$ **(autre solution)**

a(aaa)*

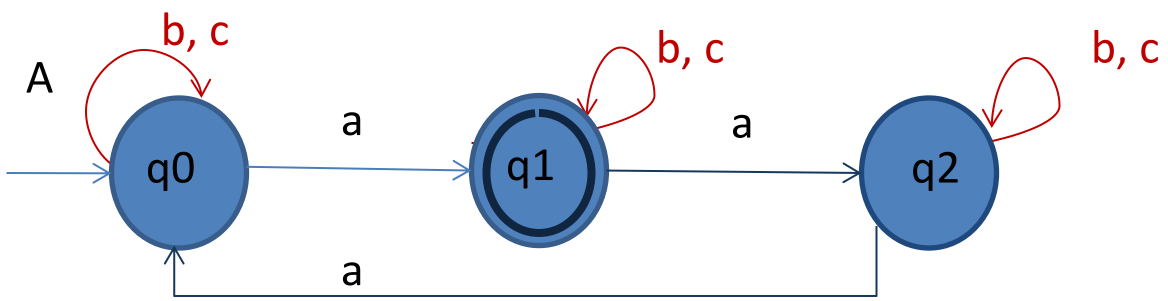


Le plus petit mot accepté est a (On atteint l'état final après la lecture de a). Après il faudra continuer à avoir un nombre multiple de 3a. Donc quand on se trouve à l'état d'acceptation ou l'état final, il faudra continuer à lire des séquences de 3a pour revenir à l'état final.

II.1. Les automates finis déterministes

4) Construire un automate fini qui accepte le langage suivant:

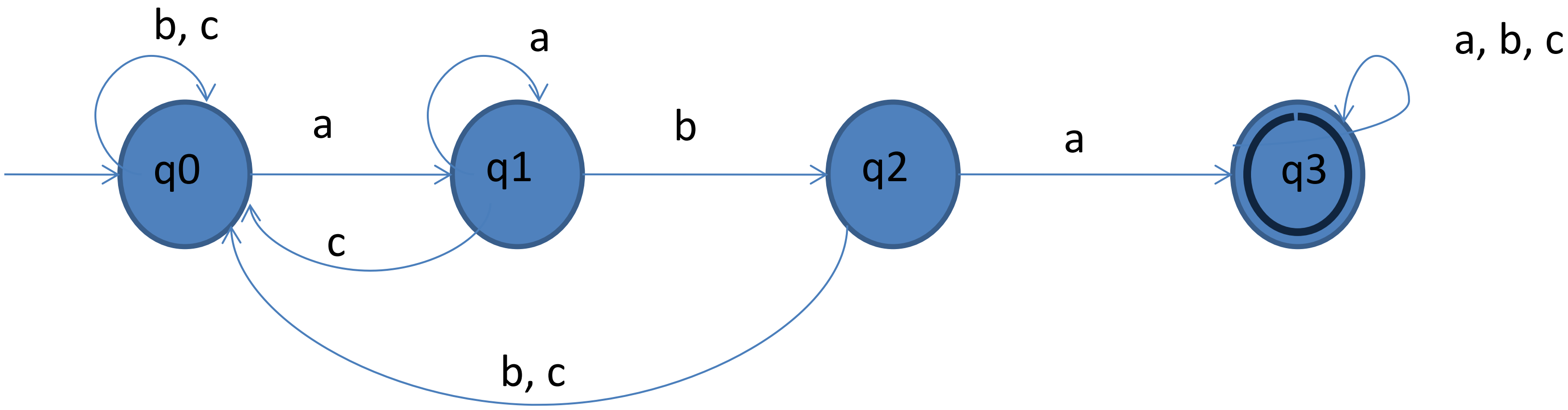
$\{\omega \in \{a, b, c\}^* / \ |\omega|_a = 3k+1, k \geq 0\}$



(b+c)*a(b+c)*(a(b+c)*a(b+c)*a(b+c)*)*

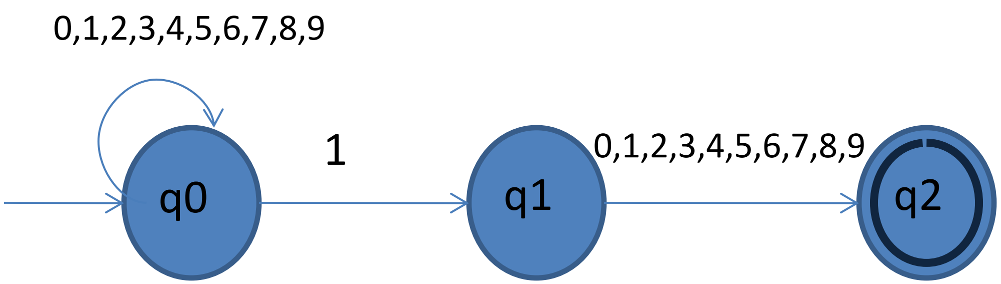
II.1. Les automates finis déterministes

5) **Construire un automate fini qui accepte l'ensemble des mots sur {a, b, c} ayant aba comme facteur (ou sous chaine)**



II.1. Les automates finis déterministes

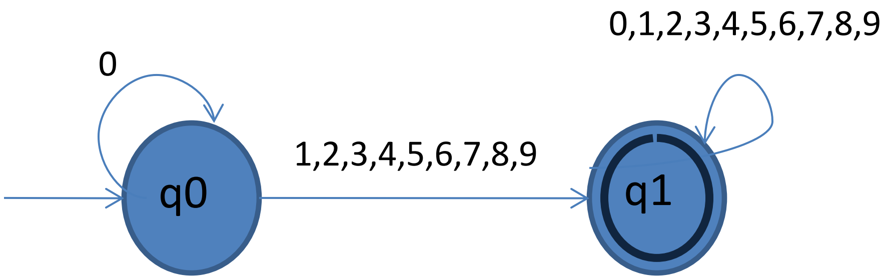
6) Construire un automate fini qui accepte l'ensemble des représentations décimales des nombres entiers ayant le chiffre 1 dans les dizaines (Exemple: 12316, 13, 210)



c*1c
(0+1+2+…+9)*1(0+1+2+…9)

II.1. Les automates finis déterministes

7) **Construire un automate fini qui accepte l'ensemble des représentations décimales des nombres entiers différents de Zéro**



0*(1+2+…9)(0+1+2+…+9)*

II.1. Les automates finis déterministes

Définition 8. Automate fini complet

Un automate fini est dit **complet** sur un vocabulaire X ssi pour chaque état q et chaque symbole s, il existe **au moins** une transition qui quitte q avec le symbole s.

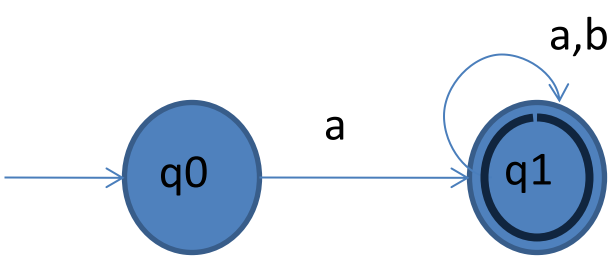
Définition 9. Automate fini non ambigu

Un automate fini est dit **non ambigu** sur un vocabulaire X ssi pour chaque état q et chaque symbole s, il existe **au plus** une transition qui quitte q avec le symbole s.

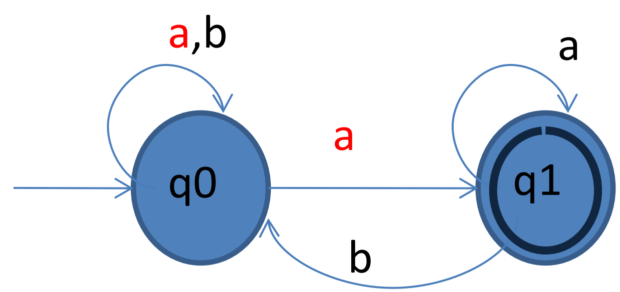
Définition 10. Automate fini déterministe

Un automate fini est dit **déterministe** sur un vocabulaire X ssi il est **complet et non ambigu** (pour chaque état q et chaque symbole s, il existe une et une seule transition qui quitte q avec le symbole s).

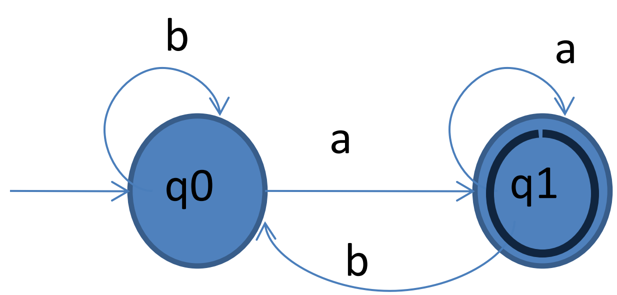
II.1. Les automates finis déterministes



Non complet sur {a, b} car
$\delta(q0, b) = \phi$
$L(A) = a(a+b)^*$



Ambigu sur {a, b} car
$\delta(q0, a) = \{q0, q1\}$
$L(A) = (a+b)^*a^+(b(a+b)^*a^+)^*$



$L(A) = b^*a^+(b^+a^+)^*$

Automate fini déterministe sur {a, b} car il est complet et non ambigu:
$\delta(q0, a) = \{q1\}$ $\delta(q0, b) = \{q0\}$ $\delta(q1, a) = \{q1\}$ $\delta(q1, b) = \{q0\}$

II.2. Langage accepté par un automate fini

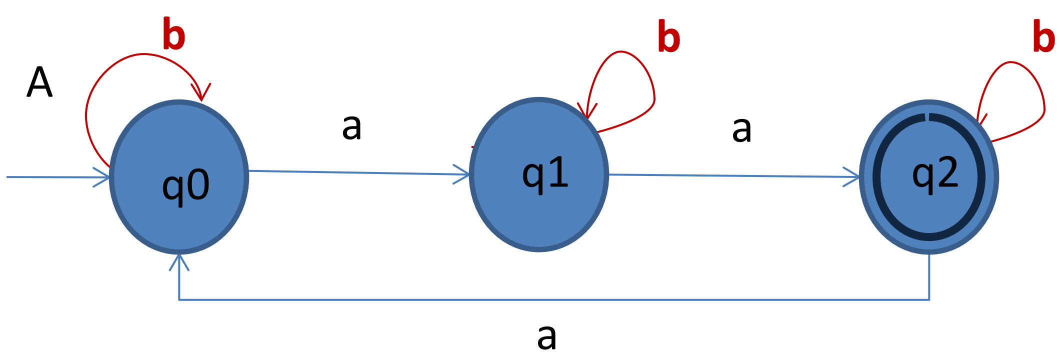
Définition 11. Configuration

Une configuration est un couple

(q, ω)

où q est l'état courant et ω est le reste du mot à lire.

Pour l'automate suivant, des configurations possibles sont:



$(q0, aa)$ $(q1, a)$ $(q2, \varepsilon)$

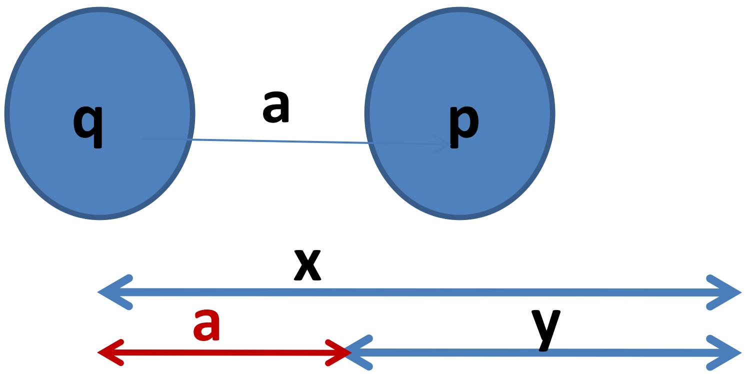
$(q0, bba)$ $(q0, ba)$ $(q0, a)$ $(q1, \varepsilon)$

II.2. Langage accepté par un automate fini

Définition 12. Configuration successeur

Une configuration (p, y) est successeur d'une configuration (q, x) qu'on note:

$(q, x) \mid\!\text{--}\ (p, y)$

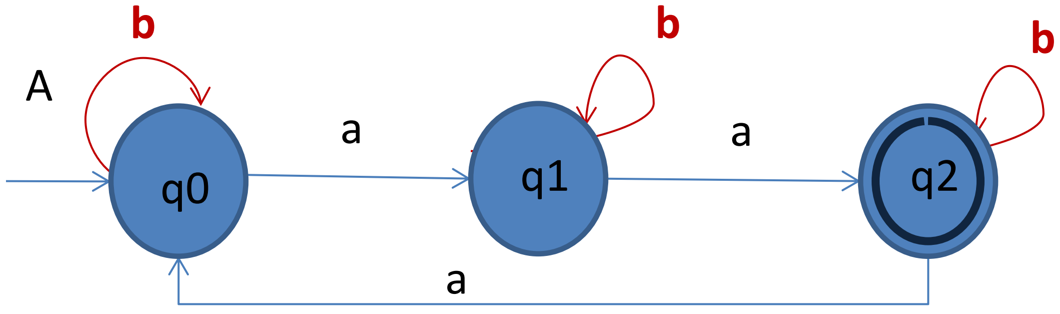


ssi

$\exists\ a \,/\, x = a\,y$ et $\delta(q, a) = p$

II.2. Langage accepté par un automate fini

Pour l'automate suivant, des configurations possibles sont:



(q0, aa) |-- (q1, a) |-- (q2, ε)

(q0, bba) |-- (q0, ba) |-- (q0, a) |-- (q1, ε)

II.2. Langage accepté par un automate fini

Définition 13. Configuration kème successeur

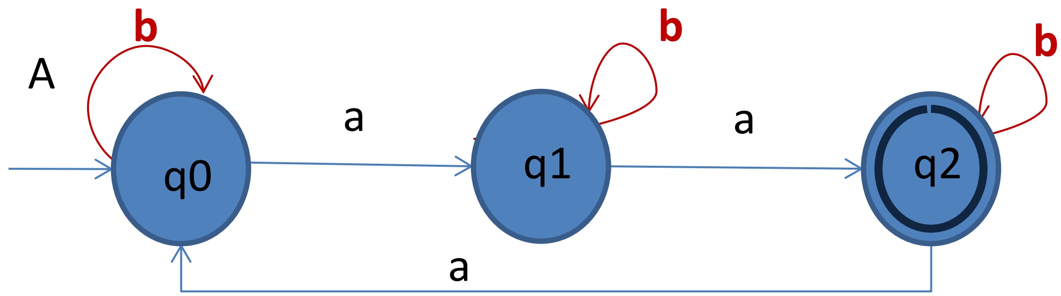
Une configuration (p, y) est kème successeur d'une configuration (q, x) qu'on note:

$$(q, x) \overset{k}{\vdash} (p, y) \text{ ssi } (q, x) \vdash (q1, x1) \vdash (q2, x2) \vdash \ldots (qk, xk) = (p, y)$$

$$(q, x) \overset{*}{\vdash} (p, y) \text{ ssi } \exists\, k \geq 0 \,/\, (q, x) \overset{k}{\vdash} (p, y)$$

II.2. Langage accepté par un automate fini

Pour l'automate suivant, nous avons :



$(q0, aa) \vdash^{*} (q2, \varepsilon)$ puisque: $\exists\, k \geq 0 \;/\; (q0, aa) \vdash^{k} (q2, \varepsilon)$

$(q0, aa) \vdash (q1, a) \vdash (q2, \varepsilon)$ $(k = 2)$

$(q0, bba) \vdash^{3} (q1, \varepsilon)$

En effet, $(q0, bba) \vdash (q0, ba) \vdash (q0, a) \vdash (q1, \varepsilon)$

II.2. Langage accepté par un automate fini

Définition 14. Langage reconnu par un automate fini

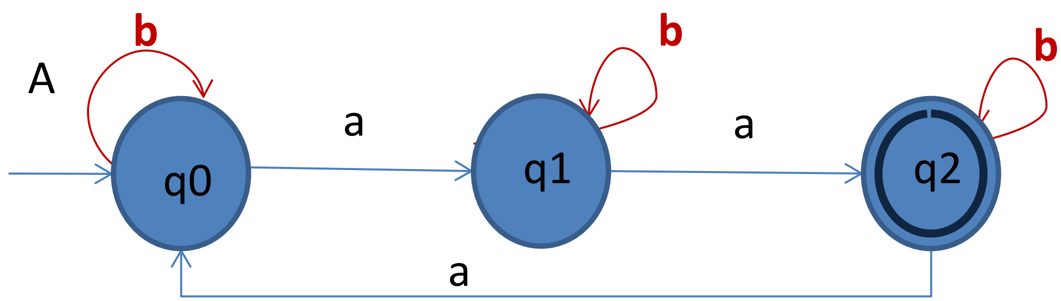
Soit A = (Q, X, δ, q0, F) un automate fini. Le langage accepté (ou reconnu par A) est noté L(A) /

$$L(A) = \{\omega / (q0, \omega) \overset{*}{\vdash\!\!-\!-} (qf, \varepsilon), qf \in F\}$$

ω est accepté par A ssi $\omega \in L(A)$

II.2. Langage accepté par un automate fini

Exercice : Soit l'automate fini A suivant :



Montrer que aa $\in$ L(A) et que bba $\notin$ L(A)

Corrigé:

aa $\in$ L(A) par ce que : (q0, aa) $\stackrel{*}{\vdash}$ (q2, ε) puisque:

(q0, aa) $\vdash$ (q1, a) $\vdash$ (q2, ε) (k = 2)

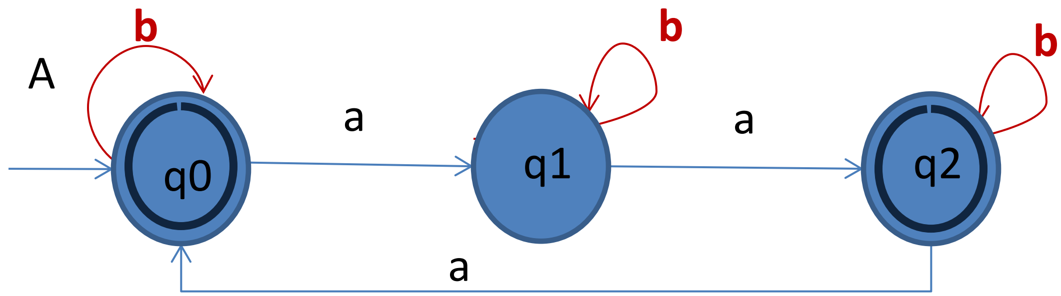
bba $\notin$ L(A) par ce que : $\neg\exists$ k / (q0, bba) $\stackrel{k}{\vdash}$ (q2, ε)

En effet, (q0, bba) $\vdash$ (q0, ba) $\vdash$ (q0, a) $\vdash$ (q1, ε)

II.2. Langage accepté par un automate fini

Exercice: Considérons l'automate A suivant :

1) Quel est le langage reconnu par A



2) Est-ce que le mot ε est accepté par A Pour l'automate suivant, nous avons :

Corrigé:

1) L(A) = **{ω ∈ {a, b}*/ |ω|a = 3k+2, k ≥ 0} + {ε}**

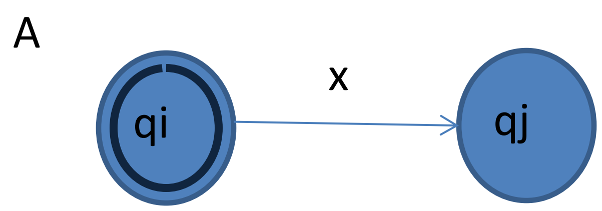
2) ε∈ L(A) par ce que : q0 ∈ F

(q0, ε) est une configuration initiale et finale.

II.2. Langage accepté par un automate fini (Lemme d'Arden)

Soit A un automate fini sur un vocabulaire X

Avec x $\in$ X



On dénote par Li, le langage reconnu par l'automate A en considérant que qi est l'état initial. Ainsi, L0 est le langage reconnu par A

$\omega_j \in Lj \Leftrightarrow \exists\ \omega_i \in Li\ /\ \omega_i = x\ \omega_j$ en conséquence Li = xLj

Si qi est un état final alors Li = x Lj + {ε} par ce que si qi est un état initial et final alors le langage Li contient le mot ε

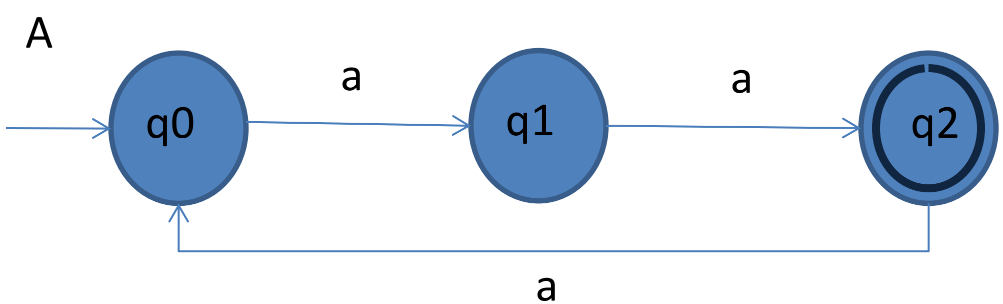
Lemme d'Arden

Pour deux langages A et B d'un vocabulaire $\sum^*$,

Les équations **L= AL+B** et **L = LA+B** admettent respectivement comme solution minimale A^*B et BA^*. Cette solution est unique si $\varepsilon \notin A$.

II.2. Langage accepté par un automate fini (Lemme d'arden)

Exercice : Soit A un automate fini sur un vocabulaire X= {a,b}, Chercher le langage L0 reconnu par l'automate A



Corrigé :

L0 = {a} L1 (1) L1 = {a} L2 (2)

L2 = {a} L0 + {ε} (3)

(1) + (2) ⇒ L0 = {a}{a}L2 = {aa}L2 (4)

(4) + (3) ⇒ L0 = {aa} L2 = {aa}({a}L0 + {ε}) ⇒ L0 = {aaa}L0 + {aa}

⇒ L0 = {aaa}*{aa} = (aaa)*aa

D'après le lemme d'Arden

II.2. Langage accepté par un automate fini (Lemme d'arden)

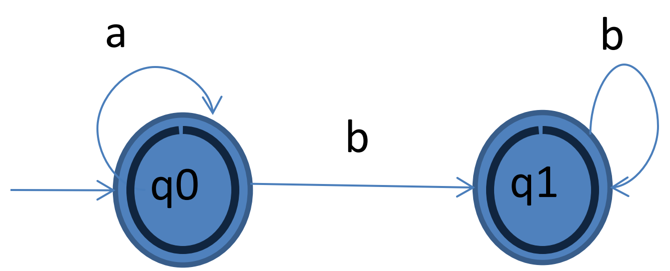
Exercice :

1) Construire un automate qui accepte le langage a*b*

2) Vérifier que l'automate construit reconnaît le langage a*b*

Corrigé :

1) A



2) Calculons L0

$L0 = \{a\} L0 + \{b\}L1 + \{\varepsilon\}$ (1)

$L1 = \{b\} L1 + [\varepsilon]$ (2)

(2) $\Rightarrow$ $L1 = \{b\}^*\{\varepsilon\}$ d'après le lemme d'Arden $\Rightarrow$ $L1 = b^*$ (3)

(1) + (3) $\Rightarrow$ $L0 = \{a\}L0 + \{b\}\, b^* + \{\varepsilon\}$

$\Rightarrow$ $L0 = \{a\}L0 + b^+ + \{\varepsilon\}$

$\Rightarrow$ $L0 = \{a\}L0 + b^*$ car $b^+ + \{\varepsilon\} = b^*$

$\Rightarrow$ $\mathbf{L0} = \{a\}^*b^* = \mathbf{a^*b^*}$ d'après le lemme d'Arden

L(A) = L0 = a*b* donc l'automate A reconnaît le langage a*b*

II.2. Langage accepté par un automate fini (Lemme d'arden)

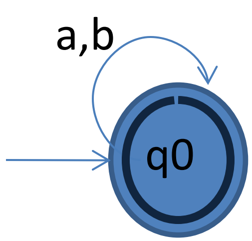
Exercice :

1) **Construire une automate qui accepte le langage (a+b)***
2) **Vérifier que l'automate construit reconnaît le langage (a+b)* (L0 est le langage accepté par l'automate puisque c'est le langage où q0 est l'état initial)**

Corrigé:

1)

A



2) $L0 = \{a\}\, L0 + \{b\}L0 + \{\varepsilon\} \qquad (1)$

$(1) \Rightarrow L0 = \{a,b\}\, L0 + \{\varepsilon\}$

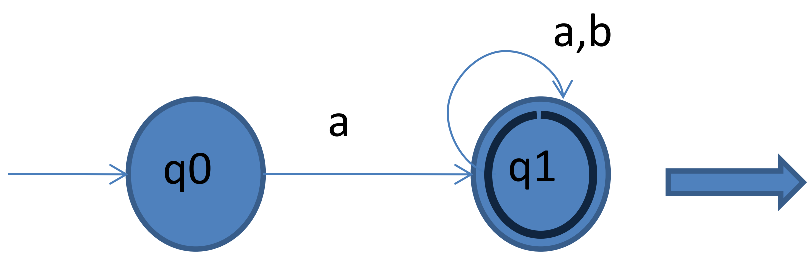
$\Rightarrow L0 = \{a,b\}^*\{\varepsilon\}$ d'après le lemme d'Arden

$\Rightarrow L0 = \{a,b\}^* = (a+b)^*$

II.3. Rendre déterministe un automate fini non déterministe

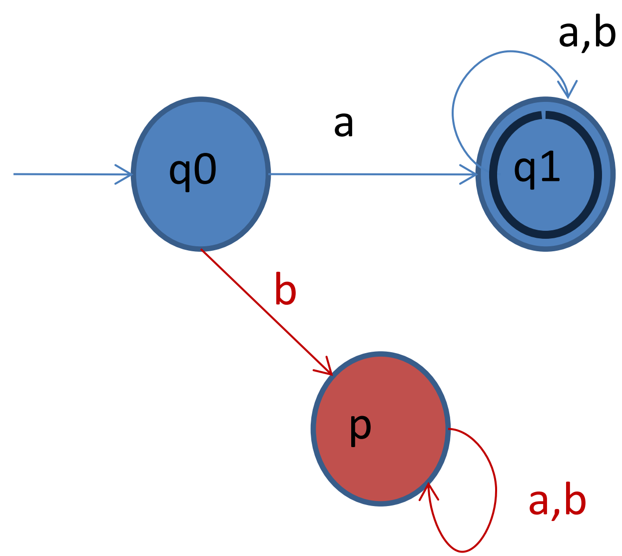
II.3.1. Automate fini non complet

Si l'automate fini est non déterministe par ce qu'il est non complet, alors pour le rendre déterministe, il suffit d'ajouter un état puit et ajouter toutes les transitions manquantes vers cet état.



Non complet sur {a, b} car
$\delta(q0, b) = \phi$

L(A) = a(a+b)*



Complet sur {a, b}

II.3. Rendre déterministe un automate fini non déterministe

II.3.2. Automate fini ambigu

Si l'automate fini est non déterministe par ce qu'il est ambigu alors on doit construire un automate en groupant les états visités par un même symbole à partir d'un état, ces états groupés forment les nouveaux états de l'automate déterministe. La construction de l'automate déterministe suit les étapes suivantes:

Etape 1. Définir les nouveaux groupes d'états

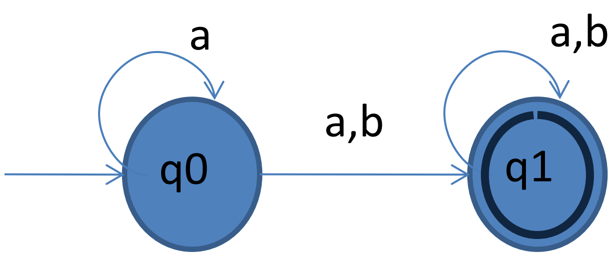
Etape 2. Renommer les nouveaux groupes et définir les états finaux. Un nouvel état (Groupe d'état) est un état final ssi il contient un ancien état final

Etape 3. Construire l'automate déterministe

II.3. Rendre déterministe un automate fini non déterministe

Exemple1

Etape 1. Définir les nouveaux groupes d'états



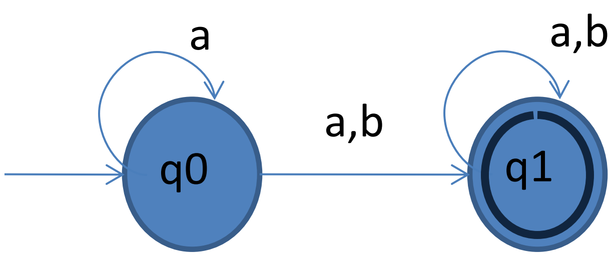
	a	**b**
{q0}	{q0, q1}	{q1}
{q0, q1}	{q0, q1}	{q1}
{q1}	{q1}	{q1}

L(A) = $a^*(a+b)^+$

II.3. Rendre déterministe un automate fini non déterministe

Exemple1

Etape 2. Renommer les états et définir les états finaux

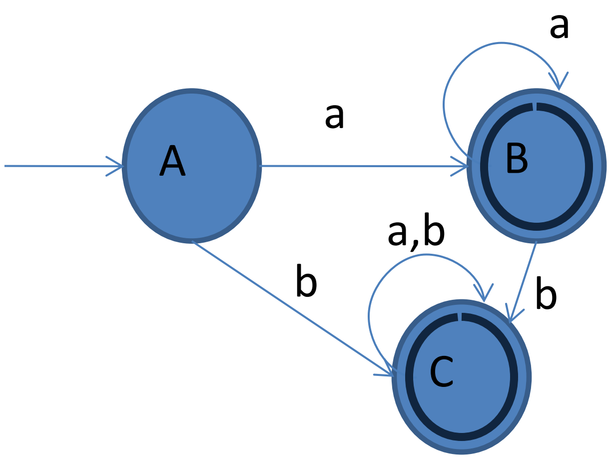


	a	b
{q0} **A**	{q0, q1} **B**	{q1} **C**
{q0, q1} **B**	{q0, q1} **B**	{q1} **C**
{q1} **C**	{q1} **C**	{q1} **C**

II.3. Rendre déterministe un automate fini non déterministe

Exemple1

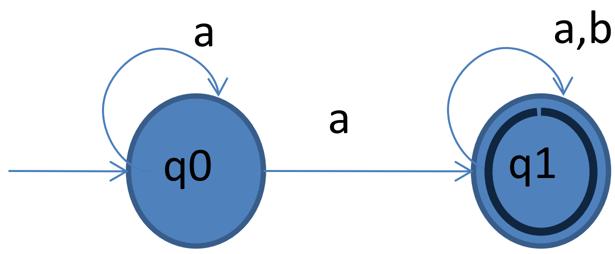
Etape 3. Construire l'automate déterministe



	a	b
{q0} **A**	{q0, q1} **B**	{q1} **C**
{q0, q1} **B**	{q0, q1} **B**	{q1} **C**
{q1} **C**	{q1} **C**	{q1} **C**

II.3. Rendre déterministe un automate fini non déterministe

Exercice : Soit l'automate A suivant, construire un automate fini déterministe qui reconnaît le même langage



Corrigé :

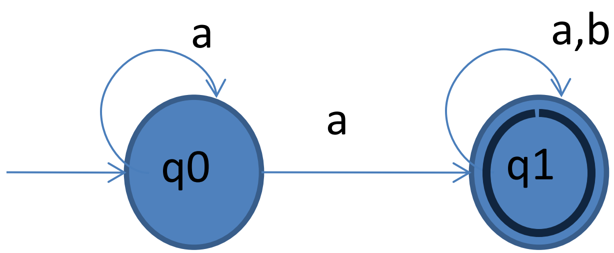
Etape1. Définir les nouveaux groupes d'états

$L(A) = a^{+}(a+b)^{*}$

	a	b
{q0}	{q0, q1}	ϕ
{q0, q1}	{q0, q1}	{q1}
{q1}	{q1}	{q1}

II.3. Rendre déterministe un automate fini non déterministe

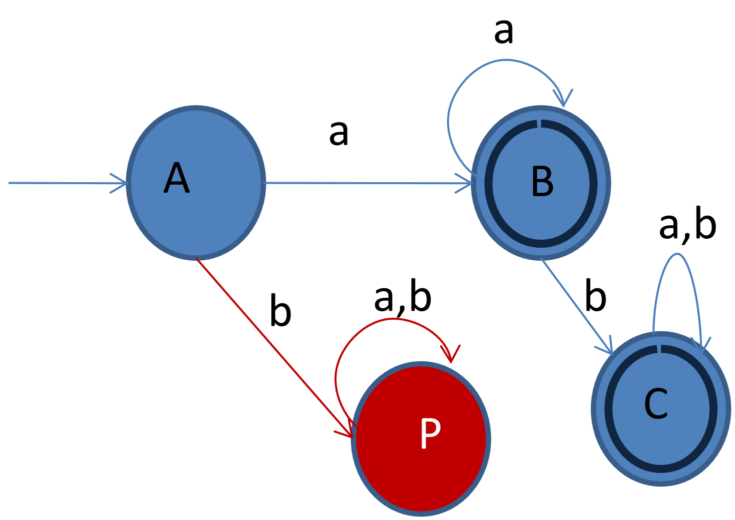
Etape 2. Renommer les états et définir les états finaux



	a	b
{q0} **A**	{q0, q1} **B**	ϕ **P**
{q0, q1} **B**	{q0, q1} **B**	{q1} **C**
{q1} **C**	{q1} **C**	{q1} **C**

II.3. Rendre déterministe un automate fini non déterministe

Etape 3. Construire l'automate déterministe



	a	b
{q0} **A**	{q0, q1} **B**	ϕ **P**
{q0, q1} **B**	{q0, q1} **B**	{q1} **C**
{q1} **C**	{q1} **C**	{q1} **C**

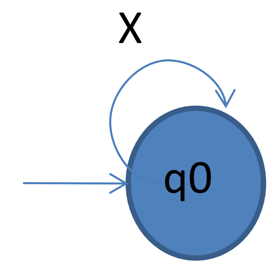
II.4. Les langages réguliers

II.4.1. Définition

Un langage L est dit régulier s'il existe un automate fini A qui l'accepte (L(A) = L)

II.4. Les langages réguliers

II.4.2. Propriétés des langages réguliers (LR)



- ϕ est un langage régulier (Il existe un automate fini qui l'accepte)
- Si L1 et L2 sont des langages réguliers alors

  L1+L2 (Union), L1.L2 (Concaténation) et L1$\cap$L2 (Intersection) sont des langages réguliers
- Si L est un langage régulier alors

  L* (Fermeture transitive), L+ (Fermeture transitive réflexive), (Complément X-L) et L (Image miroir) sont des langages réguliers

II.4. Les langages réguliers

II.4.2. Propriétés des langages réguliers (LR)

- Si L1 et L2 sont des langages réguliers alors L1+L2 (Union) est un langage régulier.

  On peut toujours construire un automate fini qui accepte L1 + L2 à partir des automates finis qui acceptent respectivement L1 et L2

Soit $A1 = (Q1, X1, \delta 1, q01, F1) / L(A1) = L1$

$A2 = (Q2, X2, \delta 2, q02, F2)/ L(A2) = L2$

On construit $A = (Q, X, \delta, q0, F)/ L(A) = L1+L2$

$Q = Q1 \cup Q2 \cup \{q0\}$

$X = X1 \cup X2$

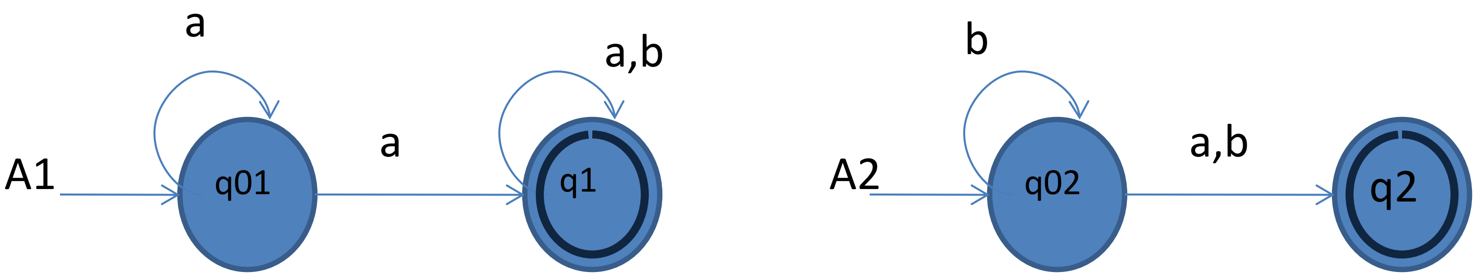
$\delta = \delta 1 \cup \delta 2 \cup \{(q0, x, q)/ (q01, x, q) \in \delta 1 \text{ ou } (q02, x, q) \in \delta 2\}$

$F =$ $F1 \cup F2 \cup \{q0\}$ si $q01 \in F1$ ou $q02 \in F2$

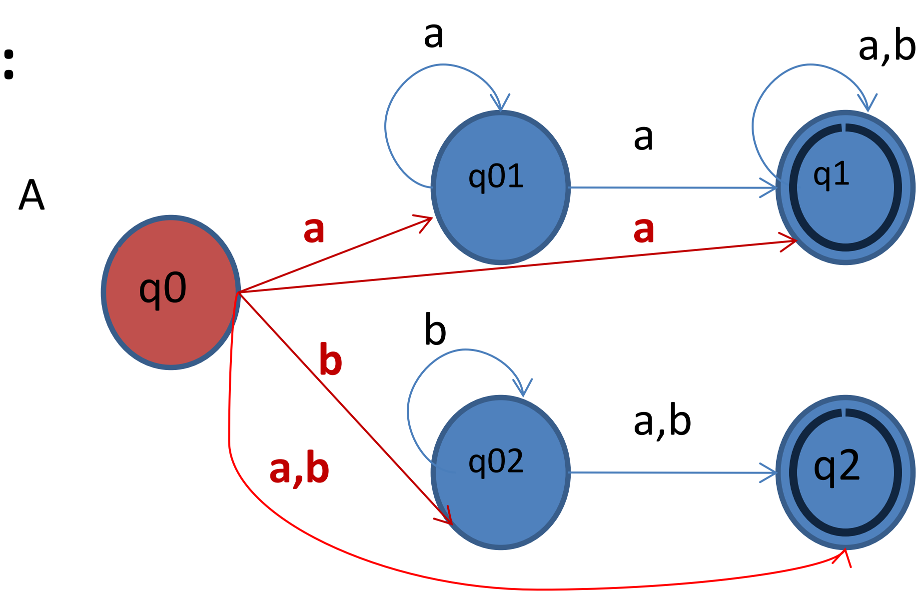
$F1 \cup F2$ sinon

II.4. Les langages réguliers

Exercice: Soient les automates A1 et A2, construire un automate A qui reconnaît L(A1) + L(A2) (l'union des deux langages reconnus par A1 et A2)



Corrigé:



II.4. Les langages réguliers

II.4.2. Propriétés des langages réguliers (LR)

- Si L1 et L2 sont des langages réguliers alors L1.L2 (concaténation) est un langage régulier.

  On peut toujours construire un automate fini qui accepte L1.L2 à partir des automates finis qui acceptent respectivement L1 et L2

Soit $A1 = (Q1, X1, \delta 1, q01, F1)$ / $L(A1) = L1$

$A2 = (Q2, X2, \delta 2, q02, F2)$/ $L(A2) = L2$

On construit $A = (Q, X, \delta, q0, F)$/ $L(A) = L1.L2$

$Q = Q1 \cup Q2$

$X = X1 \cup X2$

$Q0 = q01$

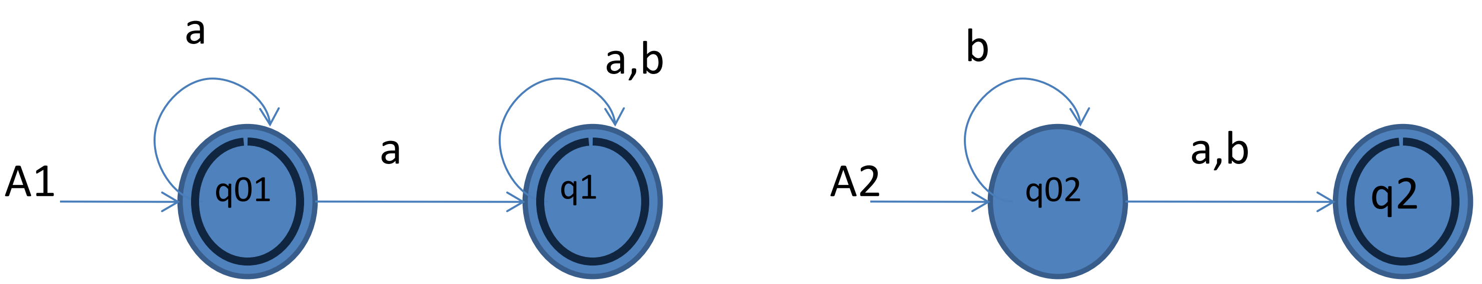
$\delta = \delta 1 \cup \delta 2 \cup \{(p, x, q) / (q02, x, q) \in \delta 2 \text{ et } p \in F1\}$

$F =$ $F1 \cup F2$ si $q02 \in F2$

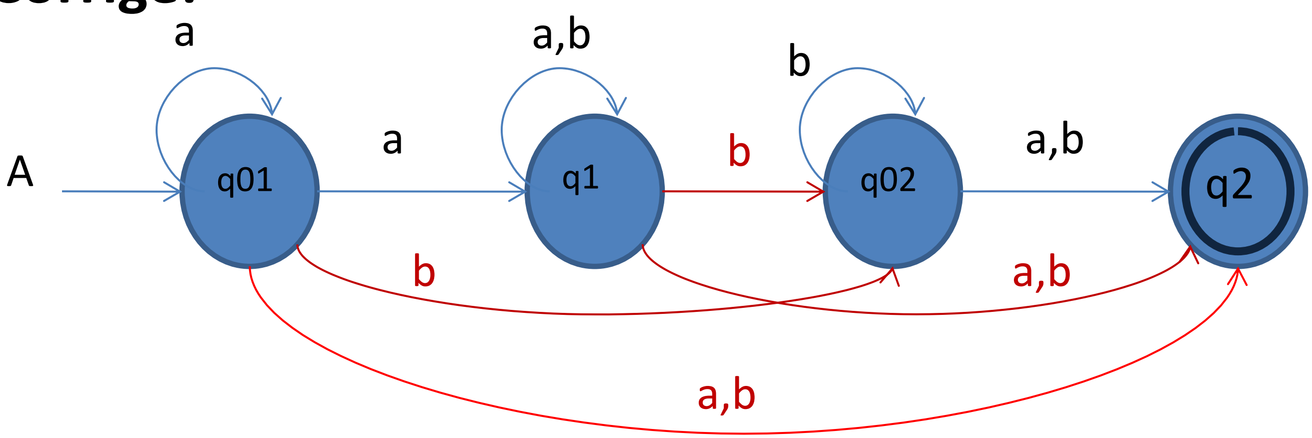
$F2$ sinon

II.4. Les langages réguliers

Exercice: Soient les automates A1 et A2 suivants, construire un automate A qui accepte L(A1) . L(A2)

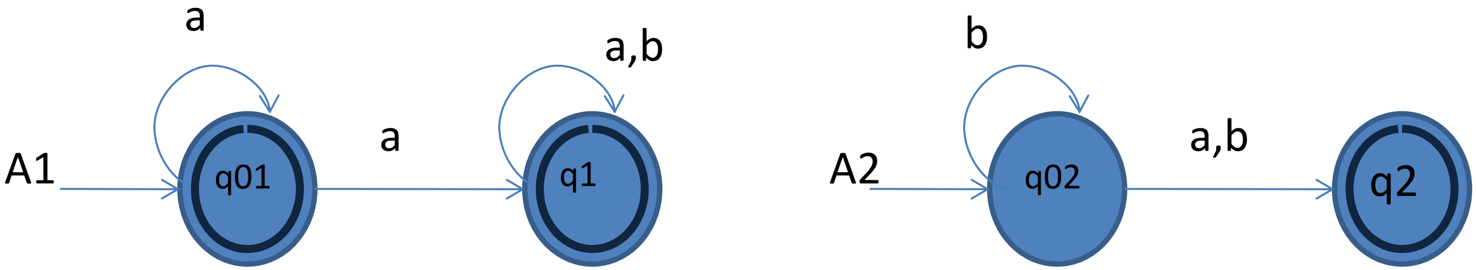


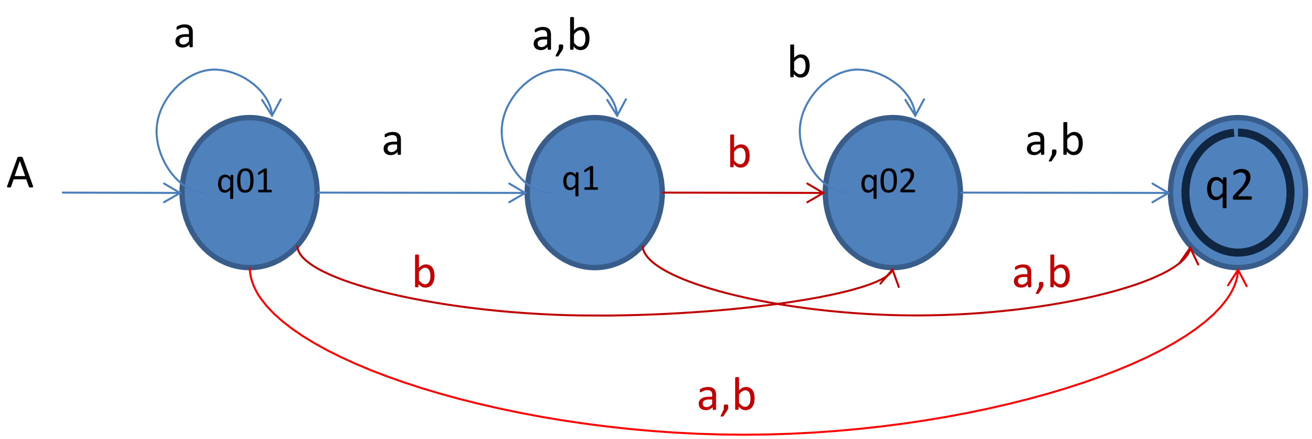
Corrigé:



II.4. Les langages réguliers

Exercice: Soient les automates A1 et A2 suivants, construire un automate A qui accepte L(A1) . L(A2)





II.4. Les langages réguliers

II.4.2. Propriétés des langages réguliers (LR)

- Si L est un langage régulier alors L* est un langage régulier

  On peut toujours construire un automate fini qui accepte L* à partir de l'automate fini qui accepte L

Soit **$A1 = (Q1, X1, \delta 1, q01, F1)$ / $L(A1) = L1$**

On construit $A = (Q, X, \delta, q0, F)$/ $L(A) = L^*$

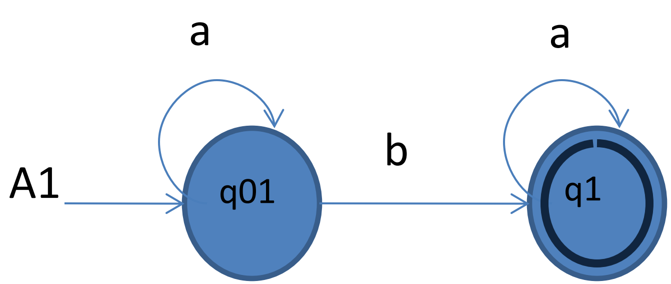
$Q = Q1 \cup \{q0\}$

$X = X1$

$\delta = \delta 1 \cup \{(q0, x, q) / (q01, x, q) \in \delta 1\} \cup \{(qf, x, q) / (q01, x, q) \in \delta 1 \text{ et } qf \in F1\}$

$F = F1 \cup \{q0\}$

Exercice: Soit l'automate A1 suivant. Construire un automate A qui accepte (L(A1))*



L(A1) = a*ba*

L(A) = (L(A1))* = (a*ba*)*

Par ce que on a :

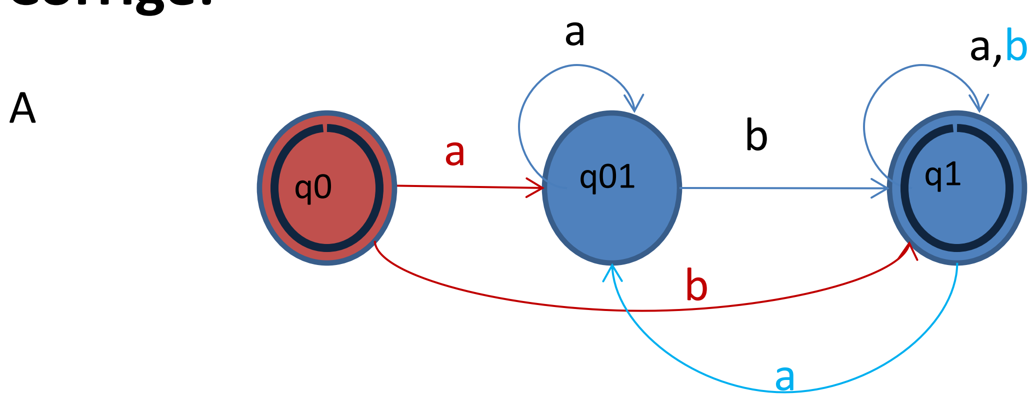
(q01, a q01) et (q01, b, q1)

dans δ1 qu'on ajoute :

(q0, a, q01) et (q0, b, q1) à δ

______

Corrigé:



Par ce que on a :

(q01, a, q01) et (q01, b, q1)

dans δ1 qu'on ajoute :

(q1, a, q01) et (q1, b, q1) à δ

II.4. Les langages réguliers

II.4.2. Propriétés des langages réguliers (LR)

- Si L est un langage régulier alors L^+ est un langage régulier
  On peut toujours construire un automate fini qui accepte L^+ à partir de l'automate fini qui accepte L

Soit $A1 = (Q1, X1, \delta 1, q01, F1)$ / $L(A1) = L1$

On construit $A = (Q, X, \delta, q0, F)$/ $L(A) = L^+$

$Q = Q1 \cup \{q0\}$

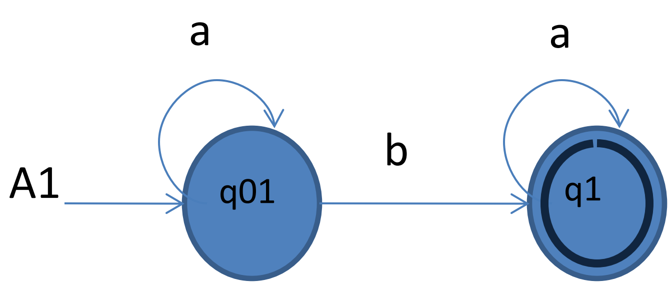
$X = X1$

$\delta = \delta 1 \cup \{(q0, x, q)/\ (q01, x, q) \in \delta 1\} \cup \{(qf, x, q)/\ (q01, x, q) \in \delta 1 \text{ et } qf \in F1\}$

$F = F1$

II.4. Les langages réguliers

Exercice: Soit l'automate A1 suivant. Construire un automate A qui accepte $(L(A1))^+$



L(A1) = a*ba*

L(A) = $(L(A1))^+$ = $(a^*ba^*)^+$

Par ce que on a :

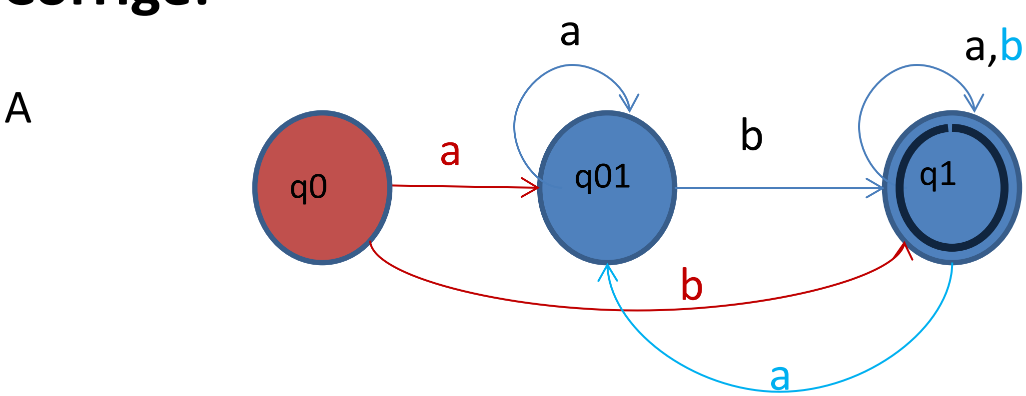
(q01, a q01) et (q01, b, q1)

dans $\delta 1$ qu'on ajoute :

(q0, a, q01) et (q0, b, q1) à δ

______

Corrigé:



Par ce que on a :

(q01, a, q01) et (q01, b, q1)

dans $\delta 1$ qu'on ajoute :

(q1, a, q01) et (q1, b, q1) à δ

II.5. Minimisation d'un automate fini déterministe

Soit A un automate fini déterministe A = (Q, X, δ, q0, F)

Pour obtenir un automate A minimal en nombre d'états, suivre les étapes suivantes:

1) Construire une partition initiale Π0 composée de deux groupes: Groupe des états finaux G1 et Groupe des états non finaux G2

2) Répéter Jusqu'à Π i+1 = Π i / i$\geq$0

- **Pour obtenir Π i+1, Partitionner chaque groupe de Πi en mettant ensemble des états p et q si pour chaque symbole s du vocabulaire, p et q transitent vers des états d'un même groupe d'états.**
- **Construire les nouveaux groupes**

3) Associer à chaque groupe un non

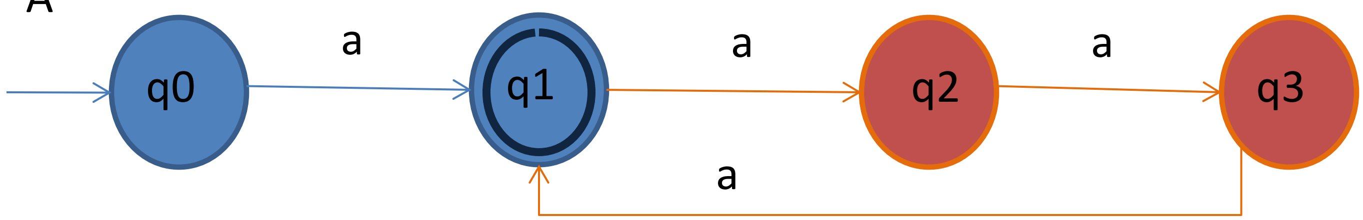
4) Construire les transitions des groupes en utilisant les transitions des états des groupes.

5) Un groupe qui contient un état final est un état final dans l'automate minimal

II.5. Minimisation d’un automate fini déterministe

Exercice : Soit l’automate A suivant. Construire un automate minimal qui accepte le même langage (Minimiser le nombre d’états de l’automate A)

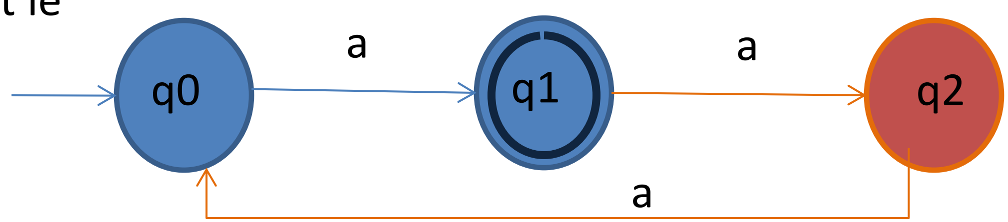
A



Corrigé:

Π0	**(q0, q2, q3)**		**(q1)**
Π1	**(q0, q3)**	**(q2)**	**(q1)**
Π2	**(q0, q3)**	**(q2)**	**(q1)**
	q0	**q2**	**q1**

On obtient le A minimal



(q2, a, q0) est dans A minimal car (q2, a, q3) était dans A et que le group d’états (q0, q3) est renommé q0

II.7. Limites des automates finis

Exercice : Prouver que le langage L suivant n'est pas un langage régulier:

$$L = \{a^i b^i,\ i \geq 0\}$$

C-a-d, il n'existe pas un automate fini qui l'accepte

Corrigé:

Supposons qu'il existe un automate fini A à n états qui accepte L (L(A) = L) (1)

Alors A accepte en particulier $a^n b^n$,

Considérons la séquence de configurations aceptables pour $a^n b^n$

$(q0, a^n b^n) \vdash (q1, a^{n-1} b^n) \vdash \ldots\ldots (q_{2n-1}, b) \vdash (q_{2n}, \varepsilon)$ **où** $q_{2n} \in F$

Cette séquence contient 2n+1 états

Donc un état au moins doit apparaître plus qu'une fois dans les n premiers mouvements. Soit q cet état, si = sj = q pour certaines valeurs de i et j telles que $0 \leq i \leq j \leq n$

Ceci entraine

$$(q0, a^n b^n) \overset{+}{\vdash} (q, a^{n-i} b^n) \overset{+}{\vdash} \ldots\ldots (q, a^{n-j} b^n) \overset{+}{\vdash} (q_{2n}, \varepsilon)$$

$$(q0, a^{n-j+i} b^n) \overset{+}{\vdash} (q, a^{n-j} b^n) \overset{+}{\vdash} \ldots\ldots (q_{2n}, \varepsilon)$$

C-a-d $a^{n-j+i} \in L(A)$

C-a-d $L \neq L(A)$ **(absurde) avec l'hypothèse (1)**