



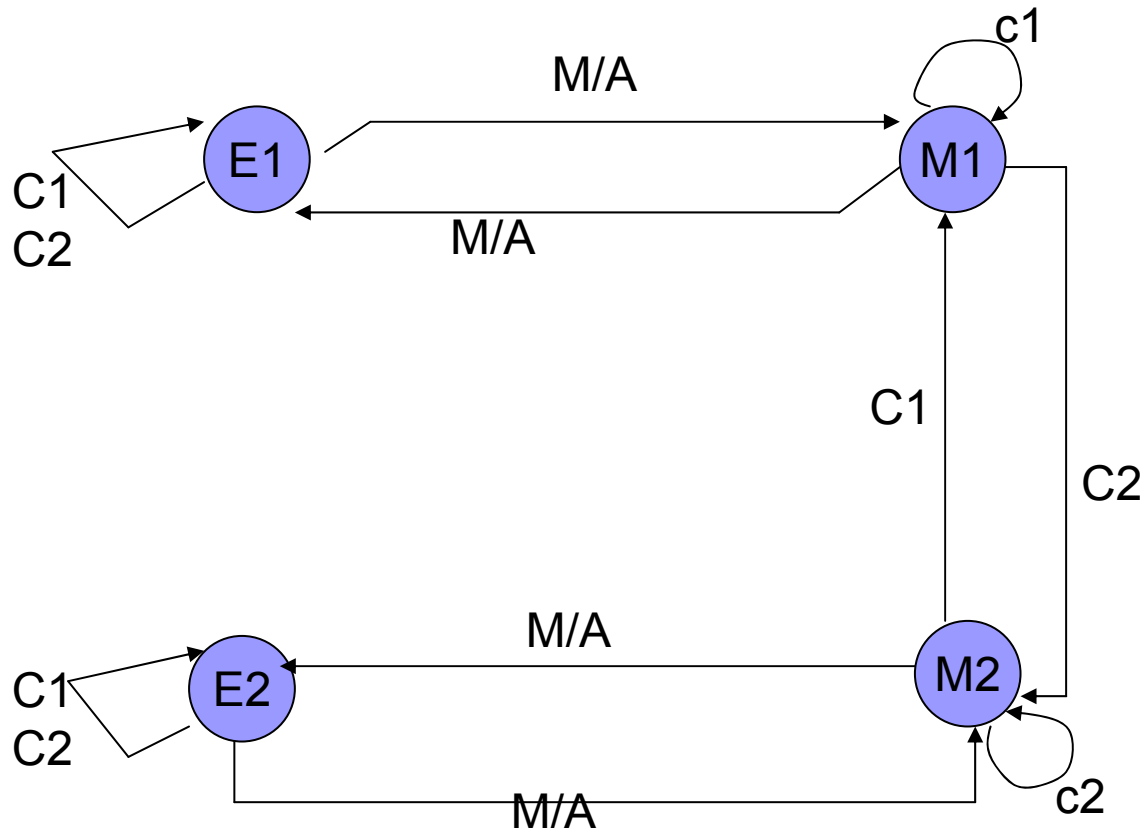
# AUTOMATES

## Exemples introductifs.

### ■ Modélisation d'une télé classique

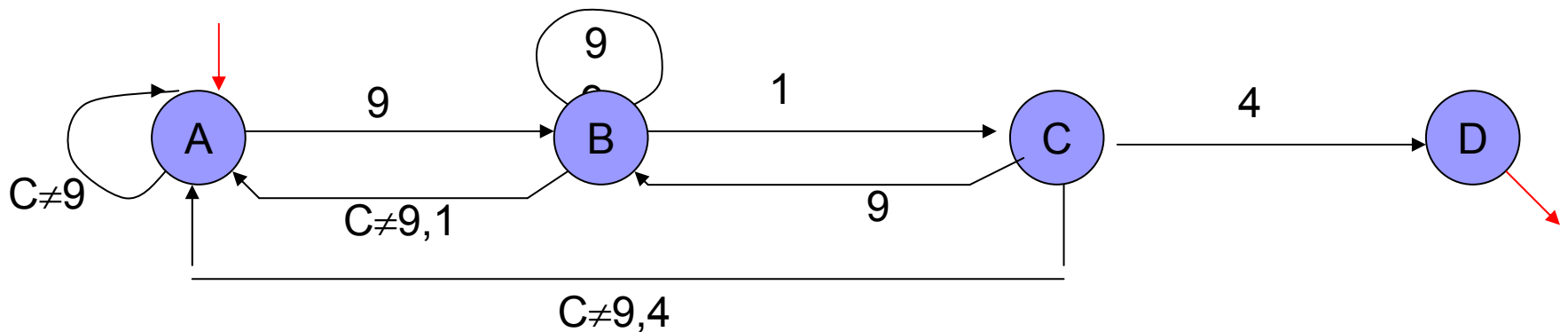
#### Description

- la télé a 3 boutons :
  - \* Marche / Arrêt → M/A
  - \* première chaîne → C1
  - \* deuxième chaîne → C2
- situation dans laquelle se trouve la télé:
  - \* Éteinte sur la chaîne 1 → E1
  - \* Éteinte sur la chaîne 2 → E2
  - \* en Marche sur chaîne1 → M1
  - \* en Marche sur chaîne2 → M2
- action sur la télé ≡ appuyer sur un bouton



## ■ Digicode

- Porte munie d'un clavier numérique
- Porte s'ouvre lorsqu'on tape le code 914
- pas de limitation sur le nombre d'essais  
(Soit  $c$  un chiffre de  $\{0, 1, \dots, 9\}$ )



B : suite se terminant par 9

C : suite se terminant par 91

D : « « « « 914

## ■ Automates finis déterministes (AFD) (AFD ou DFA).

**Définition** Un *automate fini déterministe (AFD)* est un quintuplet  $M = (Q, \Sigma, \delta, q_0, F)$  où

- $Q$  est un ensemble fini d'*états*,
- $\Sigma$  est un ensemble fini de symboles, l'*alphabet (d'entrée)*,
- $\delta : Q \times \Sigma \rightarrow Q$  est la *fonction (totale) de transition*,
- $q_0 \in Q$  est l'*état initial* (ou de *départ*),
- $F \subseteq Q$  est un sous-ensemble de  $Q$ , les *états accepteurs* (ou *finaux*).

- **Définition (Fonction de transition sur les mots)** Étant donné un AFD  $(Q, \Sigma, \delta, q_0, F)$ , on étend la fonction de transition  $\delta$  en une fonction de transition  $\delta': Q \times \Sigma^* \rightarrow Q$  sur les mots définie par :

$$\delta'(q, \varepsilon) \triangleq q$$

$$\delta'(q, wa) \triangleq \delta(\delta'(q, w), a) \quad \forall w \in \Sigma^*, \forall a \in \Sigma$$

- **Définition (Langage accepté par un automate)**

Soit  $M = (Q, \Sigma, \delta, q_0, F)$  un AFD.

1.  $M$  accepte le mot  $w \in \Sigma^*$  ssi  $\delta'(q_0, w) \in F$ . Dans le cas contraire on dit que  $M$  rejette  $w$ .

2. Le langage accepté par  $M$  est défini par :

$$L(M) \triangleq \{w \in \Sigma^* \mid \delta'(q_0, w) \in F\}$$

- *Remarque.* On identifie (souvent)  $\delta$  à  $\delta'$  en écrivant  $\delta(q, \alpha)$  où  $\alpha$  lettre ou mot.

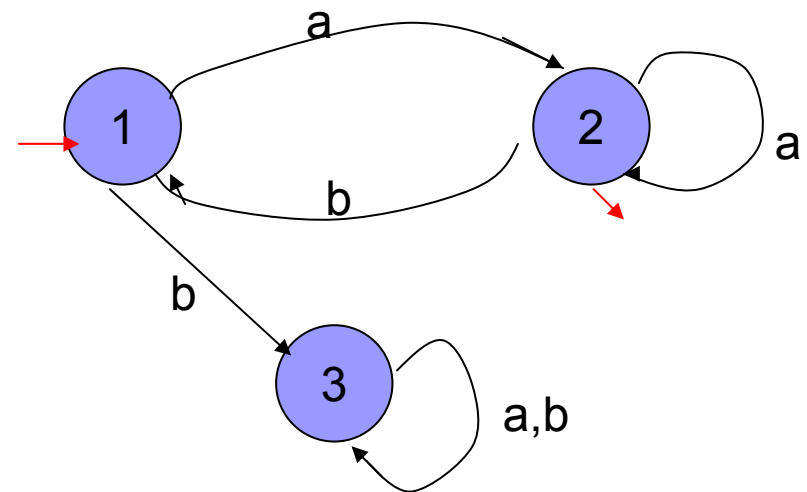
$$\begin{aligned}
[\delta(q, wa) &= \delta(\delta(q, w), a)] \\
\delta(1, aba) &= \delta(\delta(1, ab), a) \\
&= \delta(\delta(\delta(1, a), b), a) \\
&= \delta(\delta(2, b), a) \\
&= \delta(1, a) = 2
\end{aligned}$$

Ou  $[\delta(q, aw) = \delta(\delta(q, a), w)]$

$$\begin{aligned}
\delta(1, aba) &= \delta(\delta(1, a), ba) \\
&= \delta(2, ba) \\
&= \delta(\delta(2, b), a) \\
&= \delta(1, a) = 2 \\
&= \delta(\delta(\delta(1, a), b), a) = 2
\end{aligned}$$

Le mot aba est accepté ou reconnu par l'aut.

Exemple



# Algorithme de reconnaissance d'un mot

Données :

- .  $A = (Q, \Sigma, \delta, q_0, F)$  un AFD
- .  $u = u_1 u_2 \dots u_m$  ( $u_i$  est la  $i$ -ème lettre de  $u$ )

Début

$q \leftarrow q_0$

$i \leftarrow 1$

tantque ( $i \leq m$ ) faire

$q \leftarrow \delta(q, u_i)$

$i \leftarrow i + 1$

fait

si  $q \in F$  alors le mot  $u$  est accepté

sinon le mot  $u$  est rejeté

Fin



# Implémentation

- Si  $\text{card}(Q) = n$  on prend  $Q = \{0, 1, \dots, n\}$   
et  $q_0 = 0$  (par exemple)
- Si  $\text{card}(\Sigma) = m$  on code les lettres de  $\Sigma$   
par les entiers  $0, 1, \dots, m$
- $\delta$  sera représentée par une matrice  $n \times m$   
 $M[q, \ell] = \delta(q, \ell)$ ,  $0 \leq q \leq n$ ,  $0 \leq \ell \leq m$
- l'ensemble des états finaux  $F$  peut être  
représenté par un tableau de booléen :  
 $F[q] = 1$  si  $q \in F$  et  $0$  sinon

## Exemples d'automates déterministes

- $\Sigma = \{a, b\}$

Automate reconnaissant tous les mots de  $\Sigma^*$

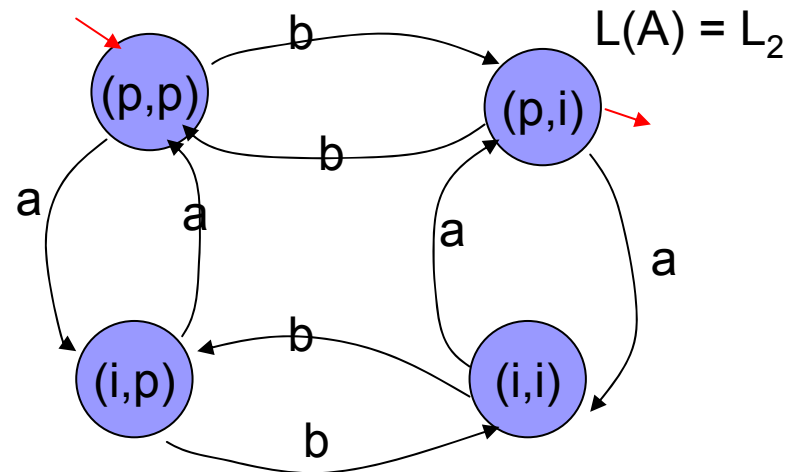
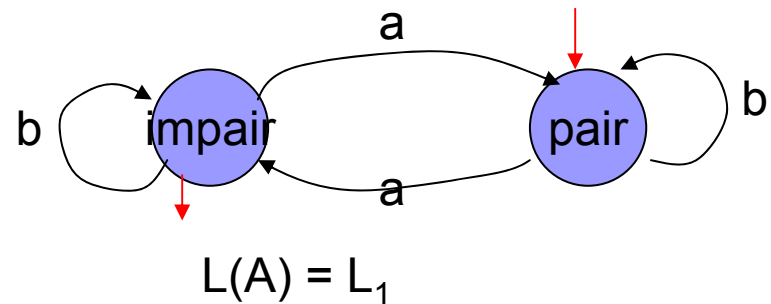
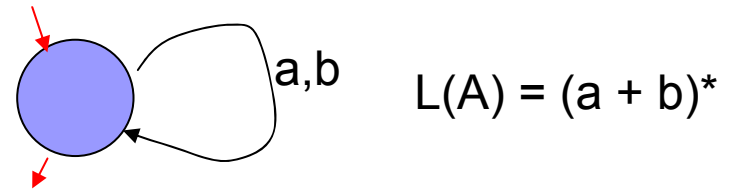
- $\Sigma = \{a, b\}$

$L_1 = \{u \in \Sigma^* / |u|_a \text{ impair}\}$

- $\Sigma = \{a, b\}$

$L_2 = \{u \in \Sigma^* / |u|_a \text{ pair et } |u|_b \text{ impair}\}$

AFD



## ■ États utiles

Soit  $A = (Q, \Sigma, \delta, q_0, F)$  un AFD et soit  $q \in Q$ .

- $q$  est accessible  $\Leftrightarrow \exists u \in \Sigma^* / \delta(q_0, u) = q$
- $q$  est co-accessible  $\Leftrightarrow \exists u \in \Sigma^* / \delta(q, u) \in F$
- $q$  est utile s'il est à la fois accessible et co-accessible
- un état qui n'est pas co-accessible est appelé état puit ou poubelle
- un automate est dit émondé s'il ne contient que des états utiles
- un automate est complet si  $\delta$  est totale i.e.  
 $\delta(q, a)$  est définie  $\forall q \in Q$  et  $\forall a \in \Sigma$

# Langages des états d'un AFD

- Définition Soit  $A = (Q, \Sigma, \delta, q_0, F)$  un AFD.  
Pour tout état  $q \in Q$ , on définit le langage  $L_q$  par :
$$\begin{aligned} L_q &\triangleq L((Q, \Sigma, \delta, q, F)) \\ &\triangleq \{u \in \Sigma^* / \delta(q, u) \in F\} \end{aligned}$$
- Lemme Soit  $A = (Q, \Sigma, \delta, q_0, F)$  un AFD.  
Alors, pour tout  $q \in Q$  :
$$L_q = \left( \bigsqcup_{a \in \Sigma} \{a\} \cdot L_{\delta(q,a)} \right) \cup \{\varepsilon / q \in F\} \quad (\text{LIN})$$
- Lemme d'Arden Soit  $\Sigma$  un alphabet. Soient  $X$ ,  $A$  et  $B$  des langages sur  $\Sigma$  satisfaisant l'équation  $X = A X + B$  avec  $\varepsilon \notin A$ .  
Alors  $X = A^* B$  est une solution unique de cette équation.

# Expression régulière définissant un AFD

## ■ Conversion d'un AFD $A$ en Exp.Rég.

1. Établir un système de  $\text{card}(Q)$  équations linéaires (format (LIN)) qui définissent les inconnus  $L_q$ , pour tout  $q \in Q$ .
2. Résoudre ce système par élimination successive des  $L_q$  (sauf  $L_{q_0}$ ) en utilisant le lemme d'Arden.
3. L'expression  $\alpha$  de  $L_{q_0}$  correspond à l'AFD  $A$ . i.e.  
$$L(A) = L_{q_0} = \alpha$$

Exemple.

$$L_0 = b L_0 + a L_1$$

$$L_1 = b L_0 + a L_2 + \varepsilon$$

$$L_2 = a L_2 + b L_2$$

Ce système est équivalent  
à:

$$L_0 = b L_0 + a L_1$$

$$L_1 = b L_0 + a L_2 + \varepsilon$$

$$L_2 = (a + b) L_2 + \emptyset$$

Par lemme d'Arden on :

$$L_2 = (a + b)^* . \emptyset = \emptyset$$

Le système devient :

$$L_0 = b L_0 + a L_1 \quad (1)$$

$$L_1 = b L_0 + \varepsilon \quad (2)$$

$$L_2 = \emptyset$$

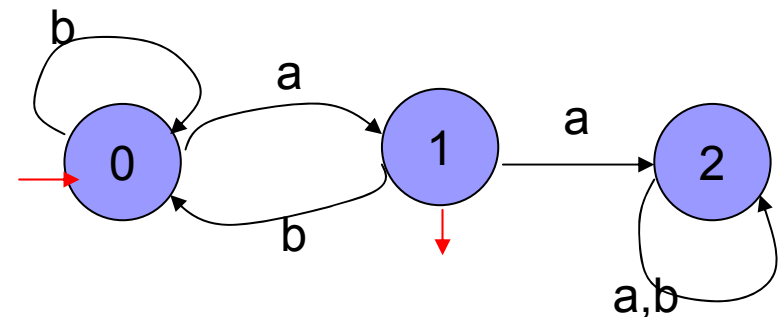
En remplaçant  $L_1$  dans (1) :

$$L_0 = b L_0 + a (b L_0 + \varepsilon)$$

$$L_0 = (b + ab) L_0 + a$$


Par le lemme d'Arden on a:

$$L_0 = (b + ab)^* a$$



## ■ Configuration d'un automate fini

- Une configuration est un couple  $(q, w) \in Q \times \Sigma^*$
- configuration initial  $(q_0, w)$  (où  $w$  est le mot à tester)
- configuration final  $(q, \varepsilon)$  ( $q \in F \Rightarrow w$  est accepté)
- Changement de configuration  
soit  $\vdash$  la relation binaire sur l'ensemble des configurations  $(\vdash \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*))$  définie par:  
 $(q, aw) \vdash (q', w)$  Ssi  $\delta(q, a) = q'$  ( $a \in \Sigma, w \in \Sigma^*$ )
- $\vdash^*$  est la fermeture réflexive – transitive de  $\vdash$
- Langage reconnu par un automate fini  $A$   
$$L(A) = \{w \in \Sigma^* / \exists q \in F : (q_0, w) \vdash^* (q, \varepsilon)\}$$



Deux automates  $A$  et  $A'$  sont  
équivalents si et seulement si ils  
reconnaissent le même langage.  
i.e.

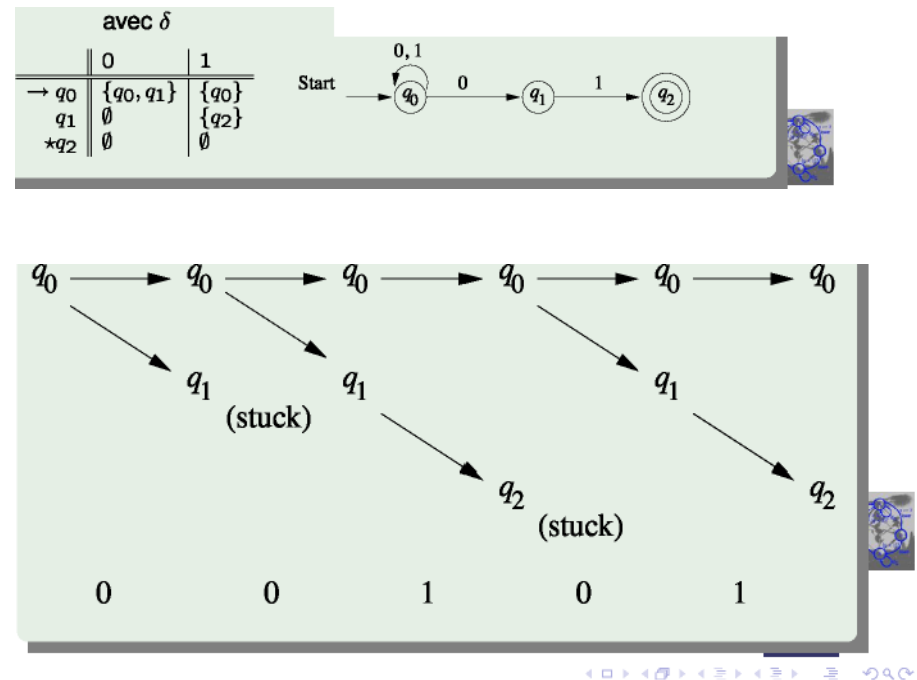
$A$  est équivalent à  $A' \iff L(A) = L(A')$



# Automate Fini Non déterministe (AFN)

## ■ Exemple

- Le AFN accepte l'ensemble des mots sur l'alphabet  $\{0, 1\}$  qui se terminent par 01.
- $\delta(q,a)$  peut être un ens.d'états au lieu d'un seul
- Pour un même mot il peut exister plusieurs chemins



## ■ Un Automate Fini Non déterministe est défini formellement par un quintuplet

$A_n = \{Q_n, \Sigma, \delta_n, Q_0, F_n\}$  où :

- $Q_n$  est un ensemble fini d'états
  - $\Sigma$  est un alphabet
  - $\delta_n : Q_n \times \Sigma \rightarrow \wp(Q_n)$  est la fonction de transition (ou  $\delta_n \subseteq (Q_n \times \Sigma \times Q_n)$  )
  - $Q_0 \subseteq Q_n$  est l'ensemble des états initiaux
  - $F_n \subseteq Q_n$  est l'ensemble des états accepteurs.
- 
- Langage reconnu par un automate fini  $A_n$   
$$L(A_n) = \{w \in \Sigma^* / \exists q_0 \in Q_0, \exists q \in F : (q_0, w) \vdash^* (q, \varepsilon) \}$$

## Extension de la fonction de transition $\delta$ aux mots

On définit pour un ensemble d'états  $S$  :

$$\delta(S, a) = \cup_{q \in S} \delta(q, a) \quad (\forall a \in \Sigma)$$

■ Extension de  $\delta$  pour les AFN  $(\delta_n : Q_n \times \Sigma \rightarrow 2^{Q_n})$

$$\delta_n^*(q, \varepsilon) = \{q\} \quad (\forall q \in Q_n)$$

$$\delta_n^*(q, wa) = \delta_n(\delta_n^*(q, w), a) \quad (\forall a \in \Sigma, \forall w \in \Sigma^*)$$

- Et on a : 
$$\begin{aligned} \delta_n^*(q, a) &= \delta_n(\delta_n^*(q, \varepsilon), a) \\ &= \delta_n(\{q\}, a) \\ &= \delta_n(q, a) \end{aligned}$$

- Langage reconnu par un AFN  $A_n$

$$L(A_n) = \{w \in \Sigma^* / \delta_n^*(q_0, w) \cap F \neq \emptyset\}$$

# Exemple.

$$\delta_n^*(q_0, 01)$$

$$= \delta_n(\delta_n^*(q_0, 0), 1)$$

$$= \delta_n(\delta_n(q_0, 0), 1)$$

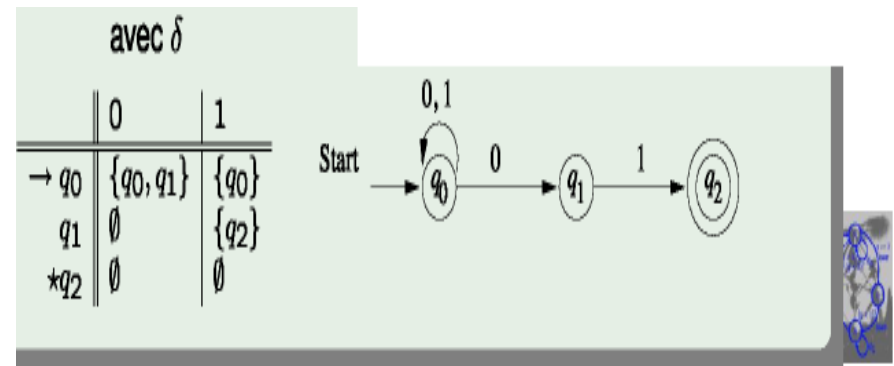
$$= \delta_n(\{q_0, q_1\}, 1)$$

$$= \delta_n(q_0, 1) \cup \delta_n(q_1, 1)$$

$$= \{q_0\} \cup \{q_2\}$$

$$= \{q_0, q_2\}$$

Le mot 01 est accepté car  $\{q_2\} \cap \{q_0, q_2\} \neq \emptyset$



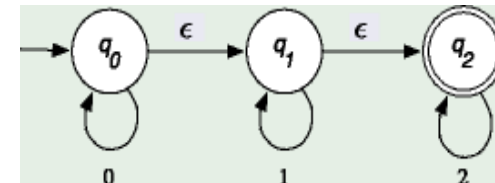
# Automate Fini Non déterministe à transitions libres ( $\text{AFN}_\epsilon$ )

Un  $\text{AFN}_\epsilon$  est un AFN auquel on ajoute des transitions étiquetées par  $\epsilon$

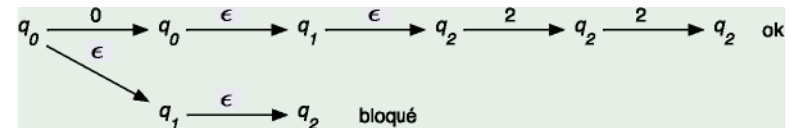
Exemple :

Le  $\text{AFN}_\epsilon$  accepte l'ensemble des mots sur l'alphabet  $\{0, 1, 2\}$  qui correspond à l'expression régulière  $0^*1^*2^*$ .

	0	1	2	$\epsilon$
$q_0$	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_1\}$
$q_1$	$\emptyset$	$\{q_1\}$	$\emptyset$	$\{q_2\}$
$q_2$	$\emptyset$	$\emptyset$	$\{q_2\}$	$\emptyset$



Reconnaissance de 022



- Un Automate  $AFN_\varepsilon$  est défini formellement par un quintuplet

$A_\varepsilon = \{Q_\varepsilon, \Sigma \cup \{\varepsilon\}, \delta_\varepsilon, s_0, F_\varepsilon\}$  où :

- $Q_\varepsilon$  est un ensemble fini d'états
- $\Sigma$  est un alphabet
- $\delta_\varepsilon : Q_\varepsilon \times (\Sigma \cup \{\varepsilon\}) \rightarrow \wp(Q_\varepsilon)$  est la fonction de transition (ou  $\delta_\varepsilon \subseteq (Q_\varepsilon \times (\Sigma \cup \{\varepsilon\})) \times Q_\varepsilon$ )
- $s_0 \in Q_\varepsilon$  est l'état initial
- $F_\varepsilon \subseteq Q_\varepsilon$  est l'ensemble des états accepteurs.
- Langage reconnu par un automate fini  $A_\varepsilon$   
$$L(A_\varepsilon) = \{w \in \Sigma^* / \exists q \in F_\varepsilon : (s_0, w) \vdash^* (q, \varepsilon)\}$$

## Extension de la fonction de transition aux mots

Extension de  $\delta$  pour les AFN $_{\varepsilon}$

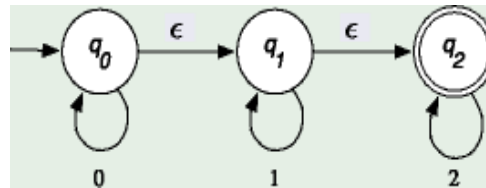
$$(\delta_{\varepsilon} : Q_{\varepsilon} \times (\Sigma \cup \{\varepsilon\}) \rightarrow \wp(Q_{\varepsilon}))$$

### Définition: ( $\varepsilon$ - clôture)

La clôture d'un état  $q \in Q$  par  $\varepsilon$ , notée  $C_{\varepsilon}(q)$ , est l'ensemble des états accessibles de  $q$  uniquement par des transitions vides.

- $C_{\varepsilon}(q) = \bigcup_{i \geq 0} \text{close}^i(q)$  où :
  - $\text{close}^0(q) = \{q\}$
  - $\text{close}^{i+1}(q) = \delta_{\varepsilon}(\text{close}^i(q), \varepsilon)$
- $C_{\varepsilon}(S) = \bigcup_{q \in S} C_{\varepsilon}(q)$
- $C_{\varepsilon}(\emptyset) = \emptyset$

## Exemple.



État $q$	$q_0$	$q_1$	$q_2$
$C_\varepsilon(q)$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$



## Extension de $\delta$ pour les AFN $_{\varepsilon}$

$$(\delta_{\varepsilon} : Q_{\varepsilon} \times (\Sigma \cup \{\varepsilon\}) \rightarrow \wp(Q_{\varepsilon}))$$

$$\cdot \delta_{\varepsilon}^*(q, \varepsilon) = C_{\varepsilon}(q)$$

$$\cdot \delta_{\varepsilon}^*(q, ua) = C_{\varepsilon}(\delta_{\varepsilon}(\delta_{\varepsilon}^*(q, u), a))$$

Et on a :

$$\begin{aligned} \delta_{\varepsilon}^*(q, a) &= C_{\varepsilon}(\delta_{\varepsilon}(\delta_{\varepsilon}^*(q, \varepsilon), a)) \\ &= C_{\varepsilon}(\delta_{\varepsilon}(C_{\varepsilon}(q), a)) \quad (\forall q \in Q, \forall a \in \Sigma) \end{aligned}$$

Exp :

$$\delta_{\varepsilon}^*(q_0, 01) = C_{\varepsilon}(\delta_{\varepsilon}(\delta_{\varepsilon}^*(q_0, 0), 1))$$

$$\begin{aligned} \delta_{\varepsilon}^*(q_0, 0) &= C_{\varepsilon}(\delta_{\varepsilon}(C_{\varepsilon}(q_0), 0)) = C_{\varepsilon}(\delta_{\varepsilon}(\{q_0, q_1, q_2\}, 0)) \\ &= C_{\varepsilon}(\delta_{\varepsilon}(q_0, 0) \cup \delta_{\varepsilon}(q_1, 0) \cup \delta_{\varepsilon}(q_2, 0)) \\ &= C_{\varepsilon}(\{q_0\} \cup \emptyset \cup \emptyset) \\ &= \{q_0, q_1, q_2\} \end{aligned}$$

$$\begin{aligned} \delta_{\varepsilon}^*(q_0, 01) &= C_{\varepsilon}(\delta_{\varepsilon}(\{q_0, q_1, q_2\}, 1)) = C_{\varepsilon}(\delta_{\varepsilon}(q_0, 1) \cup \delta_{\varepsilon}(q_1, 1) \cup \delta_{\varepsilon}(q_2, 1)) \\ &= C_{\varepsilon}(\emptyset \cup \{q_1\} \cup \emptyset) \\ &= \{q_1, q_2\} \end{aligned}$$

- $L(A_{\varepsilon}) = \{w \in \Sigma^* / \delta_{\varepsilon}(q_0, w) \cap F_{\varepsilon} \neq \emptyset\}$  (le mot 01 est donc accepté)

## ■ Équivalence entre automates finis (FA) et langages réguliers

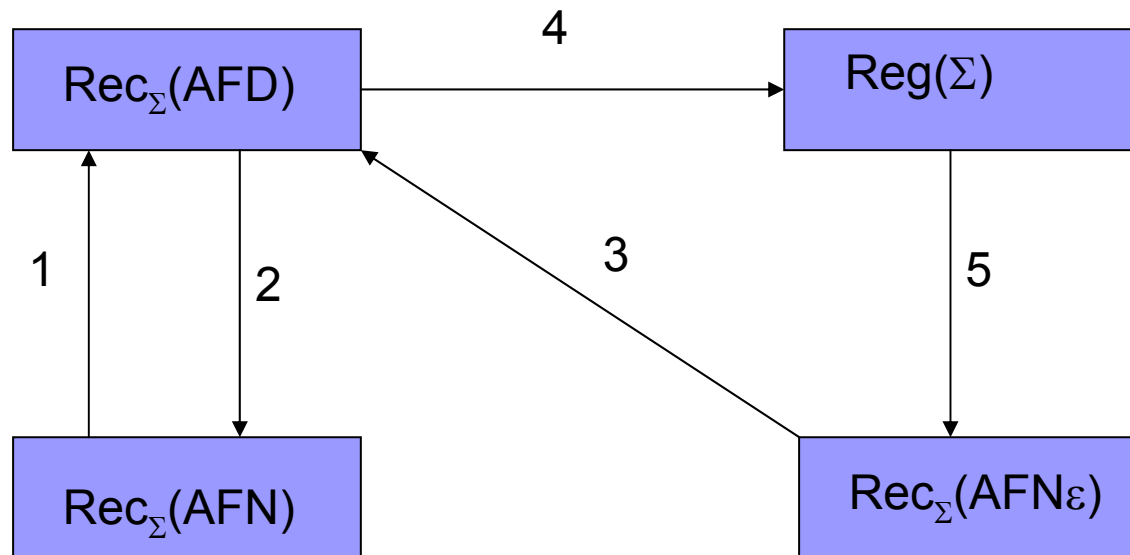
On note par  $\text{Rec}_\Sigma(\text{FA})$  la classe des langages reconnaissables par le type d'automate FA sur un alphabet  $\Sigma$ .

On montre que les 3 formalismes AFD, AFN et  $\text{AFN}_\varepsilon$  définissent la même classe de langages: les **langages réguliers**.

$$( \text{Rec}_\Sigma(\text{AFD}) = \text{Rec}_\Sigma(\text{AFN}) = \text{Rec}_\Sigma(\text{AFN}) = \text{Reg}(\Sigma) )$$

$$\text{Rec}_{\Sigma}(\text{AFN}) \subseteq \text{Rec}_{\Sigma}(\text{AFD})$$

(flèche 1)



Théorème (Pour tout AFN  $A_n$ , il existe un AFD  $A_d$  avec  $L(A_n) = L(A_d)$ )

*Preuve :*

Soit un AFN  $A_n = \{Q_n, \Sigma, \delta_n, Q_0, F_n\}$

définissons (construisons) le AFD  $A_d = \{Q_d, \Sigma, \delta_d, q_0^d, F_d\}$

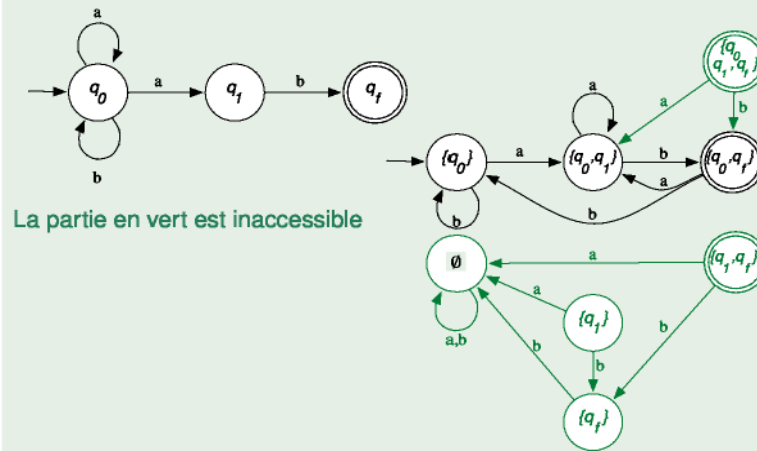
avec

- $Q_d = \{S \mid S \subseteq Q_n\}$  (càd  $Q_d = \wp(Q_n)$ )
- $F_d = \{S \subseteq Q_n \mid S \cap F_n \neq \emptyset\}$
- $q_0^d = Q_0$
- Pour tout  $S \subseteq Q_n$  et  $a \in \Sigma$ ,

$$\delta_d(S, a) = \delta_n(S, a) (= \cup_{q \in S} \delta_n(q, a))$$

- Notons que  $|Q_d| = 2^{|Q_n|}$  (bien qu'en général, beaucoup d'états soient inutiles car inaccessibles)

Exemple (NFA  $N$  et DFA  $D$  équivalent)



Théorème (Pour tout AFN  $A_n$ , il existe un AFD  $A_d$  avec  $L(A_n) = L(A_d)$ )

*Preuve (suite) :*

*On va montrer que  $L(A_d) = L(A_n)$*

*Il suffit de montrer par induction que :*

$$\delta_d^*(Q_0, w) = \delta_n^*(Q_0, w)$$

- *Base* : ( $w = \varepsilon$ ) : **OK** par définition des  $\delta^*$
- *Induction* : ( $w = ua$ )

$$\begin{aligned}\delta_d^*(Q_0, ua) &=_{\text{def}} \delta_d(\delta_d^*(Q_0, u), a) \\ &=_{hi} \delta_d(\delta_n^*(Q_0, u), a) \\ &=_{cst} \delta_n(\delta_n^*(Q_0, u), a) \\ &=_{\text{def}} \delta_n^*(Q_0, ua)\end{aligned}$$

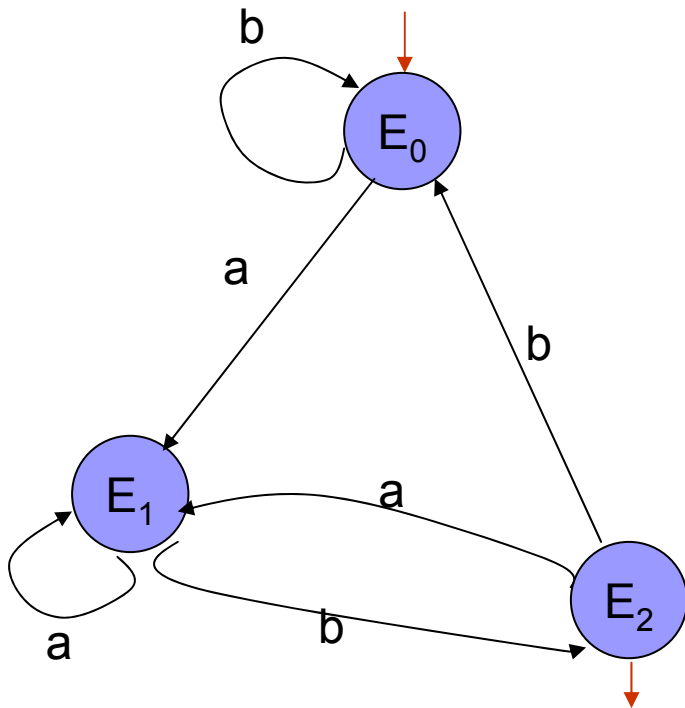
# Algorithme de détermination d'un AFN

(Conversion d'un AFN en un AFD équivalent)

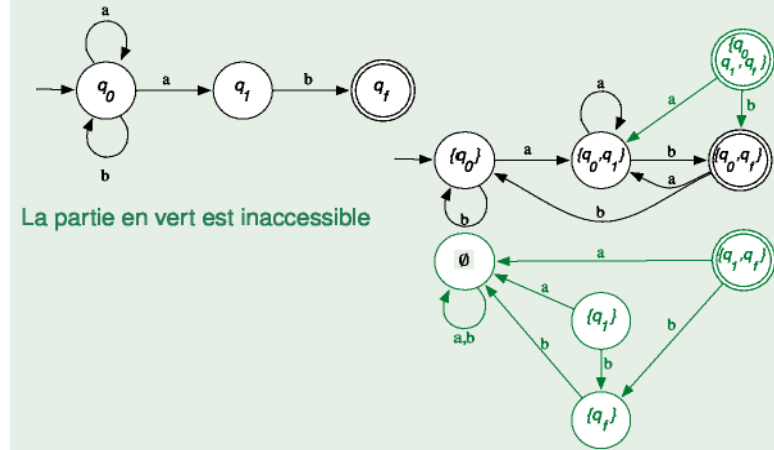
Principe : considérer des ensembles d'états successeurs de  $\{q_0\}$ .

1. Partir de l'état initial  $E_0 = \{q_0\}$
2. Construire  $E_1$  l'ensemble des états obtenus à partir de  $E_0$  par la transition  $a$ :  $E_1 = \delta_n(E_0, a)$
3. Recommencer 2. pour toutes les transitions possibles et pour chaque nouvel ensemble  $E_i$
4. Tous les ensembles  $E_i$  contenant au moins un état final deviennent terminaux
5. Renommer les ensembles  $E_i$  par de simples états

# Exemple1.



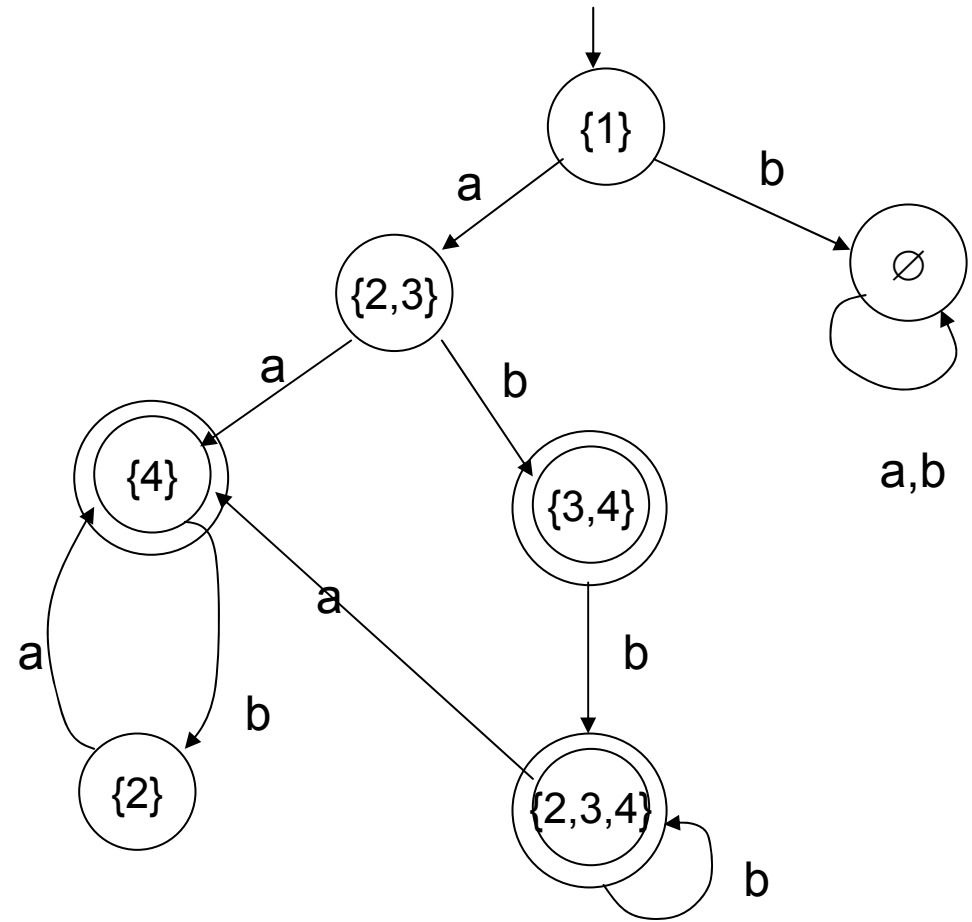
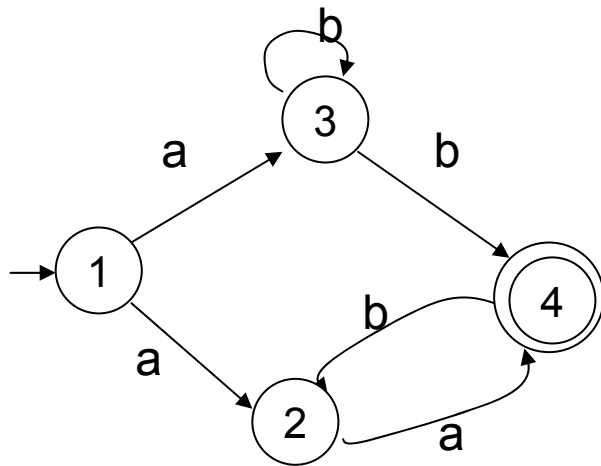
Exemple (NFA  $N$  et DFA  $D$  équivalent)



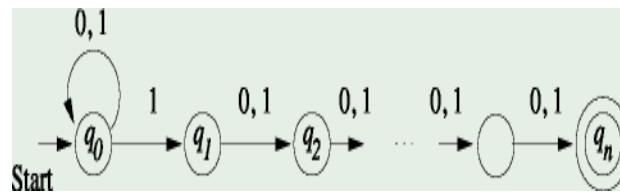
	a	b
$E_0 = \{q_0\}$	$\{q_0, q_1\} = E_1$	$\{q_0\} = E_0$
$E_1 = \{q_0, q_1\}$	$\{q_0, q_1\} = E_1$	$\{q_0, q_2\} = E_2$
$E_2 = \{q_0, q_2\}$	$\{q_0, q_1\} = E_1$	$\{q_0\} = E_0$



## Exemple2.

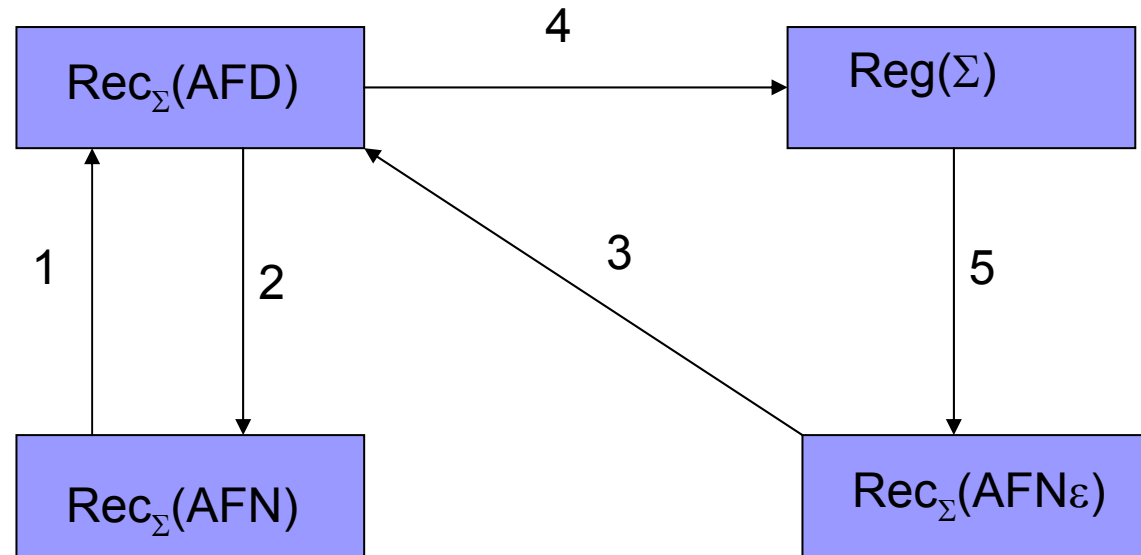


Exemple3. (AFN ayant  $n+1$  états dont l'AFD équivalent a  $2^n$  états)



# $\text{Rec}_\Sigma(\text{AFD}) \subseteq \text{Rec}_\Sigma(\text{AFN})$

(flèche 2)



Théorème (Pour tout AFD  $A_d$ , il existe un AFN  $A_n$  avec  $L(A_d) = L(A_n)$ )

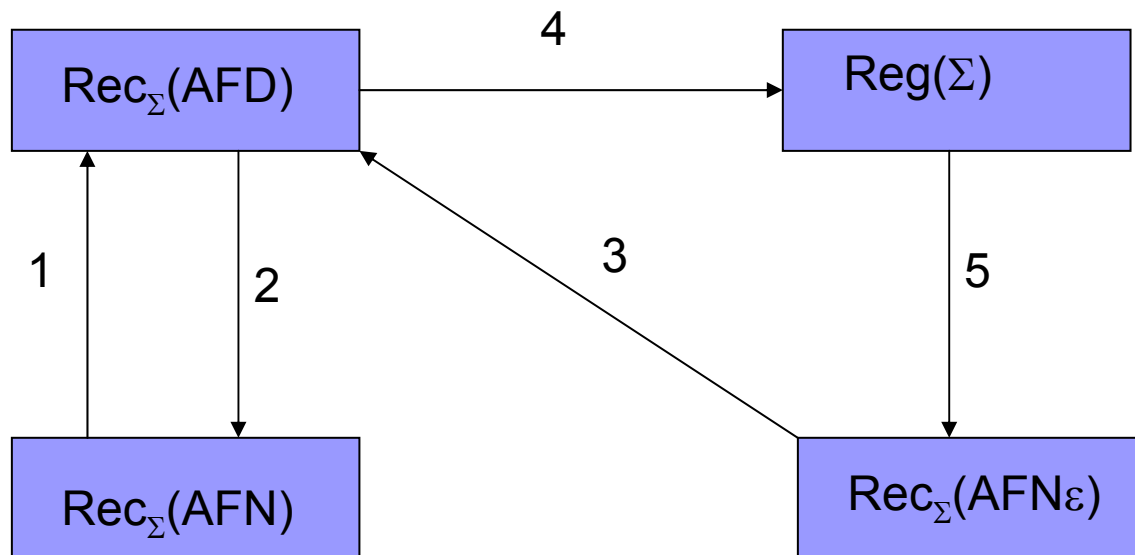
*Preuve :* Soit un AFD  $A_d = \{Q_d, \Sigma, \delta_d, q_0, F_d\}$

On prend le AFN  $A_n = \{Q_d, \Sigma, \delta_n, \{q_0\}, F_d\}$  où :

$$\delta_d(q, a) = p \Rightarrow \delta_n(q, a) = \{p\}$$

$$\text{Rec}_{\Sigma}(\text{AFN}_{\varepsilon}) \subseteq \text{Rec}_{\Sigma}(\text{AFD})$$

Flèche 3



**Théorème (Pour tout AFN $_{\varepsilon}$   $A_{\varepsilon}$  il existe un AFD  $A_d$  avec  $L(A_{\varepsilon}) = L(A_d)$ )**

*Preuve :*

Soit un AFN $_{\varepsilon}$   $A_{\varepsilon} = \{Q_{\varepsilon}, \Sigma \cup \{\varepsilon\}, \delta_{\varepsilon}, q_0^{\varepsilon}, F_{\varepsilon}\}$

définissons (construisons) le AFD  $A_d = \{Q_d, \Sigma, \delta_d, \{q_0^d\}, F_d\}$

Avec :

- $Q_d = \{ C_{\varepsilon}(S) / S \subseteq Q \}$
- $q_0^d = C_{\varepsilon}(q_0^{\varepsilon})$
- $F_d = \{S \in Q_d / S \cap F_{\varepsilon} \neq \emptyset \}$
- Pour tout  $S \in Q_d$  et  $a \in \Sigma$ ,  
 $\delta_d(S, a) = C_{\varepsilon}(\delta_{\varepsilon}(S, a))$

**Théorème (Pour tout AFN $_{\varepsilon}$   $A_{\varepsilon}$  il existe un AFD  $A_d$  avec  $L(A_{\varepsilon}) = L(A_d)$ )**

Preuve (suite)

*On va montrer que  $L(A_d) = L(A_{\varepsilon})$ . Il suffit de montrer par induction que :*

$$\delta_{\varepsilon}^*(\{q_0^{\varepsilon}\}, w) = \delta_d^*(q_0^d, w)$$

- *Base : ( $w = \varepsilon$ ) :*

$$\delta_{\varepsilon}^*(\{q_0^{\varepsilon}\}, \varepsilon) = C_{\varepsilon}(q_0^{\varepsilon}) = q_0^d = \delta_d^*(q_0^d, \varepsilon)$$

- *induction : ( $w = ua$ )*

$$\begin{aligned} \delta_{\varepsilon}^*(\{q_0^{\varepsilon}\}, ua) &=_{\text{def. } \delta_{\varepsilon}^*} C_{\varepsilon}(\delta_{\varepsilon}(\delta_{\varepsilon}^*(\{q_0^{\varepsilon}\}, u), a)) \\ &=_{hi} C_{\varepsilon}(\delta_{\varepsilon}(\delta_d^*(q_0^d, u), a)) \\ &=_{cst} \delta_d(\delta_d^*(q_0^d, u), a) \\ &=_{\text{def. } \delta_d^*} \delta_d^*(q_0^d, w) \end{aligned}$$

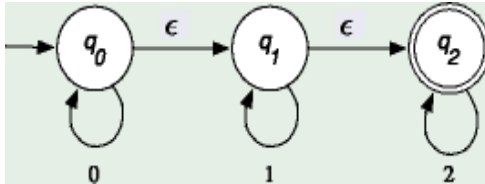
# Algorithme de détermination d'un $AFN_\varepsilon$

(Conversion d'un  $AFN_\varepsilon$  en un AFD équivalent)

Principe : considérer des ensembles d'états successeurs de  $C_\varepsilon(\{q_0\})$ .

1. Partir de l'état initial  $E_0 = C_\varepsilon(\{q_0\})$
2. Pour toute transition  $a$ , construire  $E_i$  l'ensemble des états obtenus à partir de  $E_{i-1}$  (à l'étape précédente) par :  $E_i = C_\varepsilon(\delta_\varepsilon(E_{i-1}, a))$
3. Recommencer 2. pour chaque nouvel ensemble obtenu.
4. Tous les ensembles  $E_i$  contenant au moins un état final deviennent terminaux
5. Renommer les ensembles  $E_i$  par de simples états

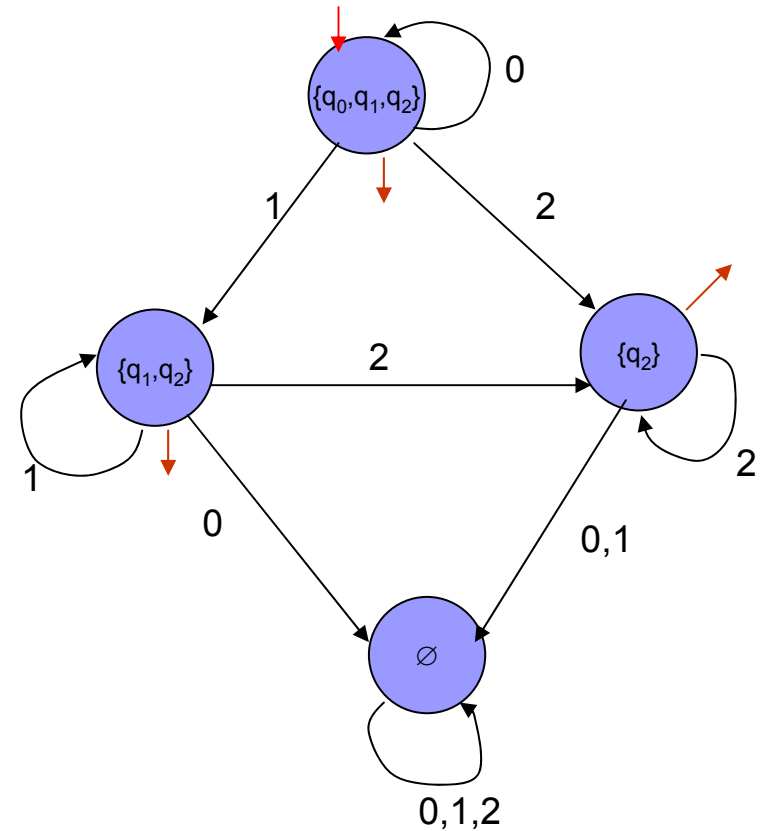
## Exemple1



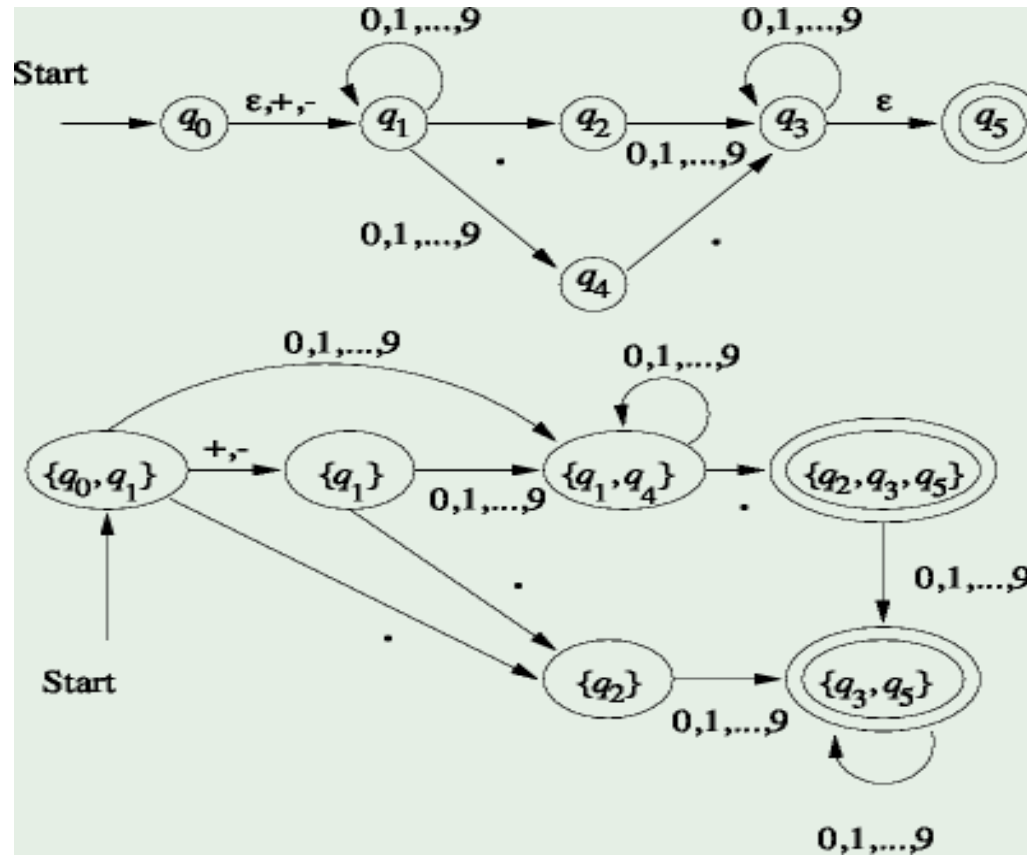
État q	$q_0$	$q_1$	$q_2$
$C_\epsilon(q)$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$

	0	1	2
$E_0 = C_\epsilon(q_0) = \{q_0, q_1, q_2\}$	$C_\epsilon(q_0) = E_0$	$E_1 = C_\epsilon(q_1) = \{q_1, q_2\}$	$E_2 = C_\epsilon(q_2) = \{q_2\}$
$E_1 = \{q_1, q_2\}$	$\emptyset$	$E_1 = C_\epsilon(q_1)$	$E_2 = C_\epsilon(q_2)$
$E_2 = \{q_2\}$	$\emptyset$	$\emptyset$	$E_2 = C_\epsilon(q_2)$

Automate déterministe équivalent



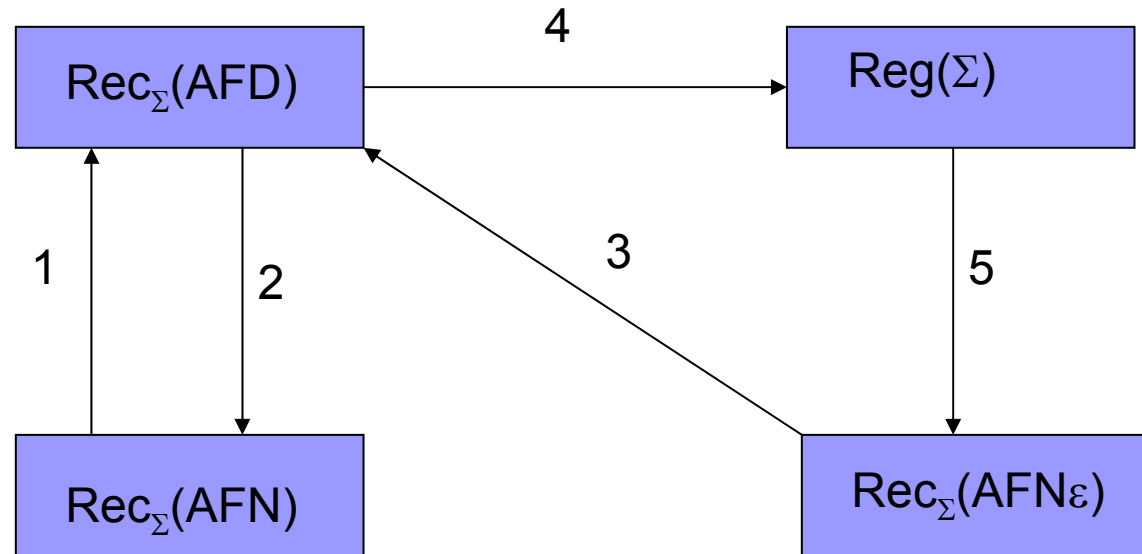
## Exemple2





$$\text{Rec}_\Sigma(\text{AFD}) \subseteq \text{REG}(\Sigma)$$

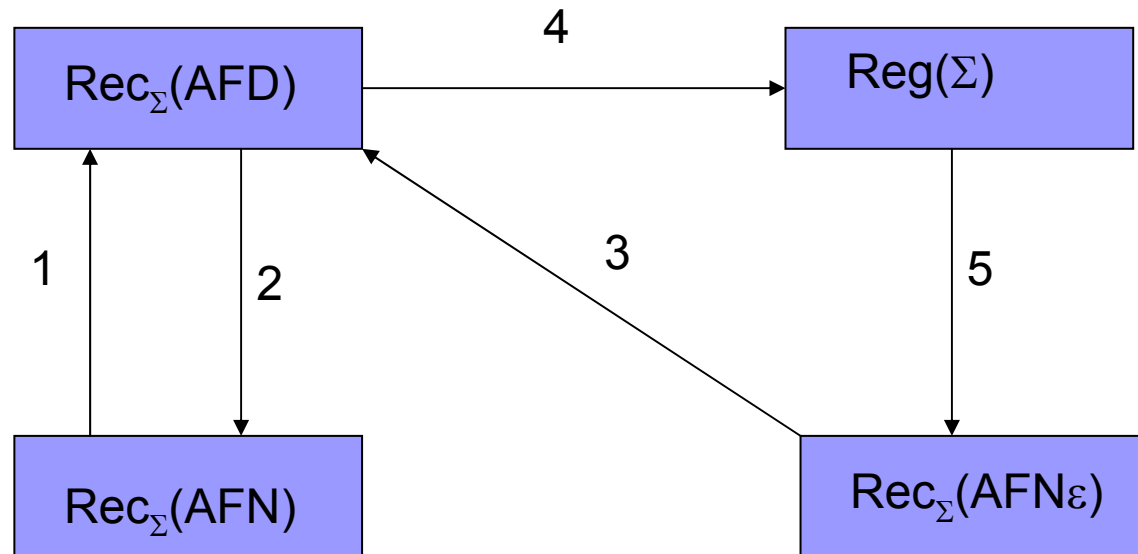
Flèche 4



**Théorème :** Tout langage reconnaissable par un AFD est défini par une expression régulière.

(Pour le calcul de l'exp. Régulière on peut résoudre un système de  $\text{card}(Q_d)$  équations et  $\text{card}(Q_d)$  inconnues (pp. 12-13))

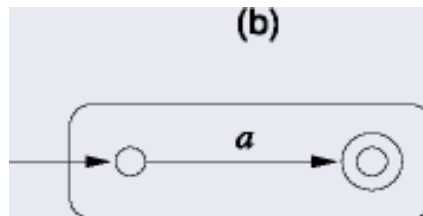
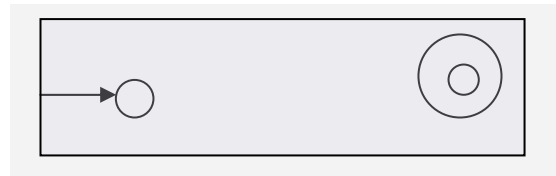
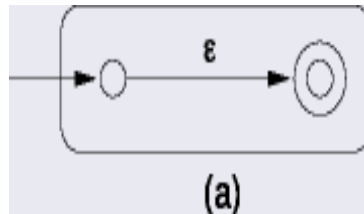
$\text{REG}(\Sigma) \subseteq \text{Rec}_{\Sigma}(\text{AFN}_{\varepsilon})$   
Flèche 5



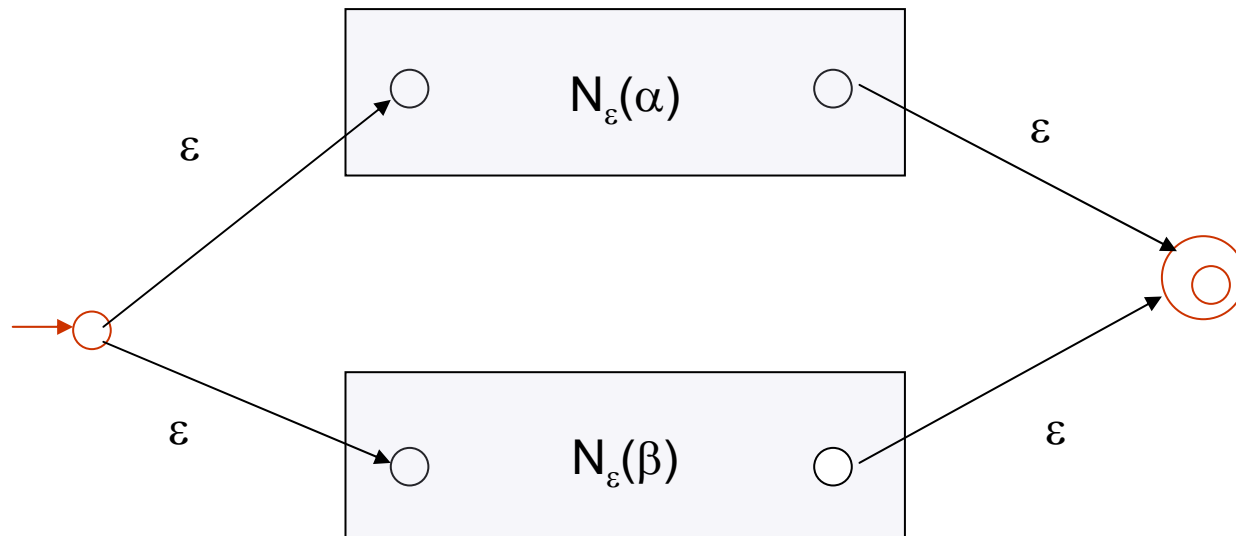
**Théorème :** Pour toute expression régulière  $\alpha$  , il existe un  $\text{AFN}_\varepsilon$   $N_\varepsilon$  Tel que  $L(\alpha)=L(N_\varepsilon)$

*Preuve : par induction.*

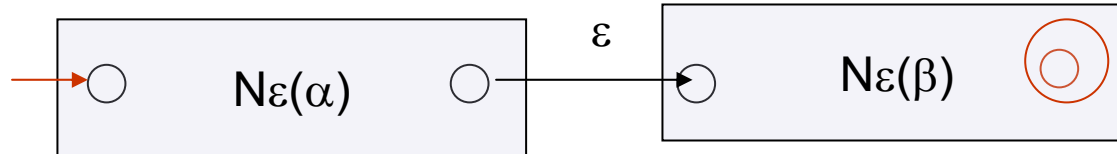
- *Cas de base :* automates pour  $\varepsilon$ ,  $\emptyset$  et  $a$  ( $a \in \Sigma$ )



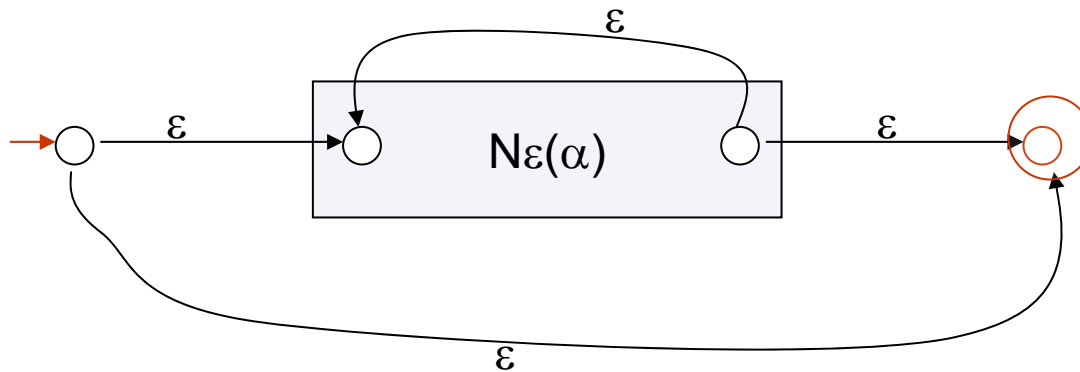
- **Induction** : automate pour  $\alpha + \beta$



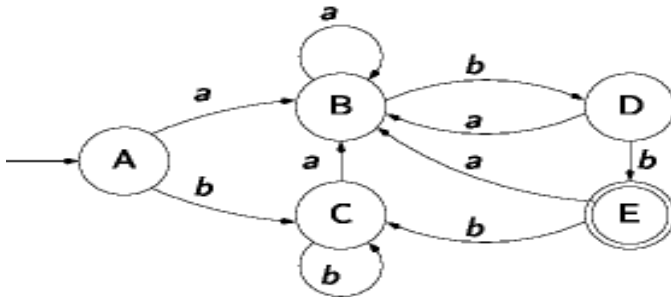
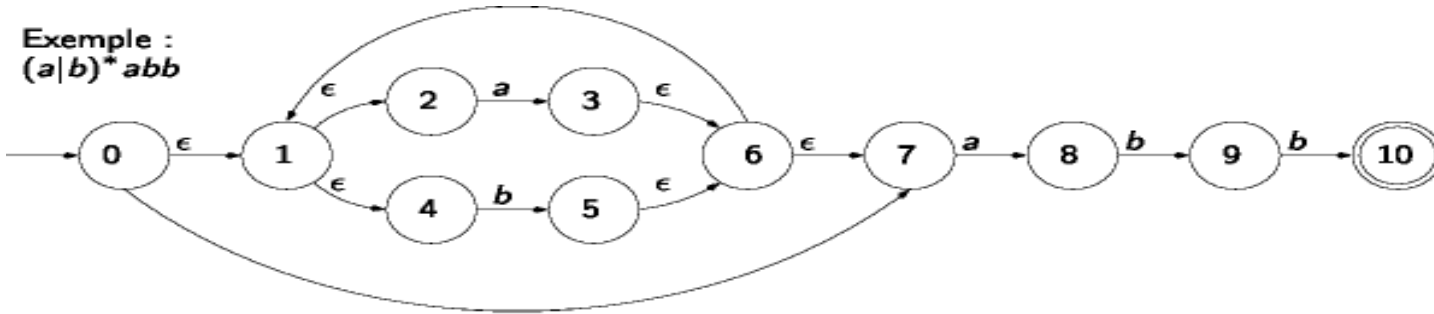
**Indiction** : automate pour  $\alpha . \beta$



**Induction** : automate pour  $\alpha^*$



Exemple :  
 $(a|b)^*abb$



$$A = \{0, 1, 2, 4, 7\}$$

$$B = \{1, 2, 3, 4, 6, 7, 8\}$$

$$C = \{1, 2, 4, 5, 6, 7\}$$

$$D = \{1, 2, 4, 5, 6, 7, 9\}$$

$$E = \{1, 2, 4, 5, 6, 7, 10\}$$

Conclusion: les 3 formalismes d'automates sont équivalents et définissent la classe des langages réguliers



# Automate minimal

## • Congruence de Myhill – Nerode

Soit  $L$  un langage sur  $\Sigma$  ( $L \subseteq \Sigma^*$ ) et soient  $u$  et  $v$  deux mots de  $\Sigma^*$ .

On définit une relation d'équivalence  $\sim_L$  ( $\sim_L \subseteq \Sigma^* \times \Sigma^*$ ) par :

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^*, (u w \in L \Leftrightarrow v w \in L)$$

- $\sim_L$  **est une congruence à droite** ( $u \sim_L v \Rightarrow u \alpha \sim_L v \alpha$ ) ( $\alpha \in \Sigma^*$ )  
( $u \sim_L v \Rightarrow \forall \gamma \in \Sigma^*, (u \gamma \in L \Leftrightarrow v \gamma \in L)$ )  
( $u \alpha \sim_L v \alpha \Rightarrow \forall \beta \in \Sigma^*, (u \alpha \beta \in L \Leftrightarrow v \alpha \beta \in L)$ ) (prendre  $\gamma = \alpha \beta$ )

- **Automate quotient dans le cas où  $\sim_L$  est d'index fini**

$A / \sim_L = \{Q_{\sim}, \Sigma, \delta_{\sim}, q_{0\sim}, F_{\sim}\}$  où :

- $Q_{\sim} = \{[u]_{\sim}, u \in \Sigma^*\}$
- $q_{0\sim} = [\varepsilon]_{\sim}$
- $F_{\sim} = \{[u]_{\sim}, u \in L\}$
- $\delta_{\sim} : Q_{\sim} \times \Sigma \rightarrow Q_{\sim}$  définie par :  
 $\delta_{\sim}([u]_{\sim}, a) = [ua]$



- L'automate quotient  $A / \sim_L$  reconnaît les mots de  $L$ . ( $L(A / \sim_L) = L$ )

En effet :

$$\begin{aligned}
 w \in L(A / \sim_L) &\Leftrightarrow \delta_{\sim}(q_{0\sim}, w) \in F_{\sim} \\
 &\Leftrightarrow \delta_{\sim}([\varepsilon]_{\sim}, w) \in F_{\sim} \\
 &\Leftrightarrow [w]_{\sim} \in F_{\sim} \\
 &\Leftrightarrow w \in L
 \end{aligned}$$

**Exemple :**  $L = a^* b a^* = \{a^i b a^j, i, j \geq 0\}$

$$\square \varepsilon \sim_L a$$

$$\varepsilon a^i b a^j \in L \text{ et } a^{i+1} b a^j \in L$$

$$\varepsilon b b \notin L \text{ et } a b b \notin L$$

$$(\text{de même } a^i \sim_L a^j \sim_L \varepsilon \forall i, j \geq 0)$$

$$[\varepsilon] = a^*$$

$$\square \varepsilon \sim_L b \text{ (} \varepsilon b \in L \text{ et } b b \notin L \text{)}$$

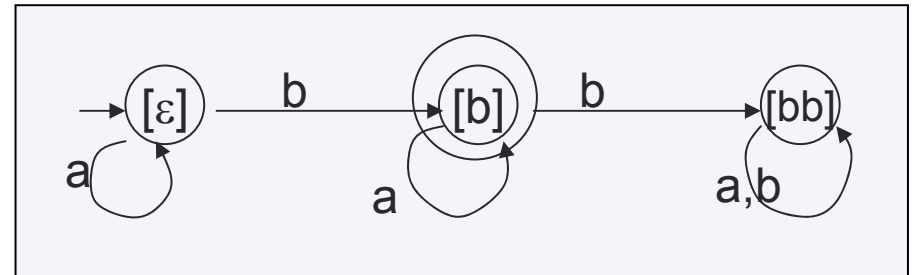
$$\square b \sim_L b a \text{ (} b a^j \in L \text{ et } b a a^j \in L \text{)}$$

$$(\text{de même } a^k b \sim_L b a^l \sim_L a^k b a^l \sim_L b \forall k, l \geq 0)$$

$$[b] = a^* b a^*$$

$$\square b \sim_L b b \text{ (} b \in L \text{ et } b b \notin L \text{)}$$

$$[bb] = a^* b a^* b (a+b)^*$$



## Théorème de Myhill - Nerode

Soit  $L$  un langage quelconque sur  $\Sigma$ . ( $L \subseteq \Sigma^*$ )

$L$  est régulier  $\Leftrightarrow$  la congruence  $\sim_L$  est d'index fini

## ● Cas où L est un langage régulier

Si L est régulier il est reconnaissable par un AFD  $A = (\Sigma, Q, \delta, q_0, F)$

Soient p et q deux états quelconque de Q, accessible de  $q_0$ . On note par :

$$p = \delta(q_0, u) \quad \text{et} \quad q = \delta(q_0, v) \quad (u, v \in \Sigma^*)$$

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^* : (u w \in L \Leftrightarrow v w \in L)$$

$$\Leftrightarrow \forall w \in \Sigma^* : (\delta(q_0, uw) \in F \Leftrightarrow \delta(q_0, vw) \in F)$$

$$\Leftrightarrow \forall w \in \Sigma^* : (\delta(\delta(q_0, u), w) \in F \Leftrightarrow \delta(\delta(q_0, v), w) \in F)$$

$$\Leftrightarrow \forall w \in \Sigma^* : (\delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F)$$

$$\Leftrightarrow \forall w \in \Sigma^* : (w \in L_p \Leftrightarrow w \in L_q)$$

$$\Leftrightarrow L_p = L_q$$

Et on écrit :

$$p \approx q \Leftrightarrow \forall w \in \Sigma^* : (\delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F)$$

On a :

$A / \sim_L$  est isomorphe à  $A / \approx$

Les 2 automates quotients sont identiques: ils ont même nombre d'états et Ils ne diffèrent que par les noms des états

Remarque: les langages  $L_p$  (resp.  $L_q$ ) est appelé résiduel de  $L$  par rapport à  $u$  (res.  $v$ )  
 $u^{-1} L = \{w \in \Sigma^* / uw \in L\}$   
 (un résiduel par rapport à  $u$  est l'ensemble des mots de  $L$  dont on efface le préfixe  $u$ )

Exemple.

$$L1 = a L2 + b L3$$

$$L2 = a L4 + b L5$$

$$L3 = a L4 + b L6 + \varepsilon$$

$$L4 = a L4 + b L5$$

$$L5 = a L4 + b L6 + \varepsilon$$

$$L6 = a L4 + b L6$$

On en déduit :

$$L2 = L4 \Rightarrow 2 \approx 4$$

$$L3 = L5 \Rightarrow 3 \approx 5$$

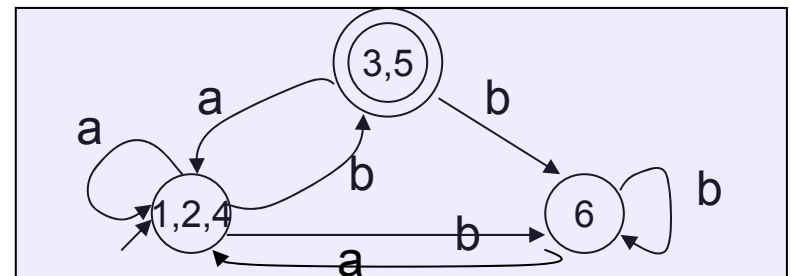
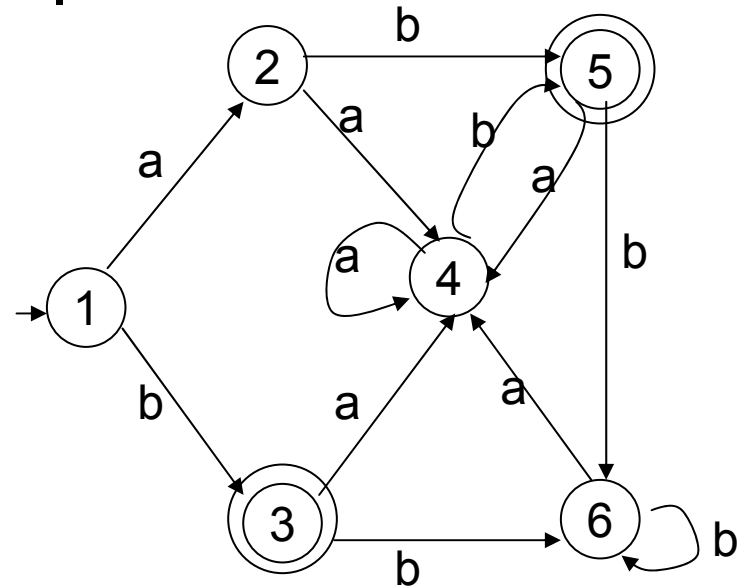
Ces deux résultats font que  $L1 = L2$

Et donc  $1 \approx 2 \approx 4$ .

$L1 \neq L6$  ( $\varepsilon \in L3$ ,  $b \in L1$  mais  $b \notin L6$ )  
 car  $\varepsilon \notin L6$ )

Les classes:  $\{1,2,4\}$ ,  $\{6\}$  et  $\{3,5\}$

Automate minimal équivalent



- $p \approx q \iff \forall w \in \Sigma^* : (\delta(p,w) \in F \iff \delta(q,w) \in F)$

- $p \not\approx q \iff \exists w \in \Sigma^* : (\delta(p,w) \in F \text{ et } \delta(q,w) \notin F)$

ou

$$(\delta(p,w) \notin F \text{ et } \delta(q,w) \in F)$$

On dit, dans ce cas que, w **sépare** p et q.

- Pour  $n \geq 0$ , on note  $\Sigma^{\leq n} = \Sigma^0 \cup \Sigma^1 \cup \dots \cup \Sigma^n$   
(est l'ensemble des mots de longueur  $\leq n$ ) et on définit l'équivalence  $\approx_n$  sur Q par :

$$p \approx_n q \iff \forall w \in \Sigma^{\leq n} (\delta(p,w) \in F \iff \delta(q,w) \in F)$$

### Remarques :

- $\approx_0$  a pour classes d'équivalence F et  $Q \setminus F$ .
- $\approx_{n+1}$  est plus fine que  $\approx_n$ , ( $p \approx_{n+1} q \Rightarrow p \approx_n q$ )
- $\approx = \bigcap_{n \geq 0} \approx_n$ , ( $p \approx q \iff \forall n \geq 0, p \approx_n q$ )

On montre que :

- $p \approx_{n+1} q \iff p \approx_n q \text{ et } \forall a \in \Sigma, \delta(p,a) \approx_n \delta(q,a)$
- Si  $\approx_{n+1} = \approx_n$  alors  $\approx = \approx_n$

On utilise ce principe pour calculer les partitions  $\pi_n$  (relatives à  $\approx_n$ ) de  $Q$  par raffinements successifs en partant de  $\pi_0 = \{ F, Q \setminus F \}$ .

L'automate minimal est l'automate quotient

$$A / \approx = (\Sigma, Q / \approx, \delta_{\approx}, [q_0]_{\approx}, F_{\approx}) \text{ où}$$

$$Q / \approx = \{ [q]_{\approx}, q \in Q \}$$

$$F_{\approx} = \{ [q]_{\approx}, q \in F \}$$

$$\delta_{\approx} : Q / \approx \times \Sigma \rightarrow Q / \approx \text{ définie par :}$$

$$\delta_{\approx}([q]_{\approx}, a) = [\delta(q, a)]_{\approx}$$

- L'automate quotient  $A / \approx = (\Sigma, Q/\approx, \delta_{\approx}, [q_0]_{\approx}, F_{\approx})$  est équivalent à  $A$

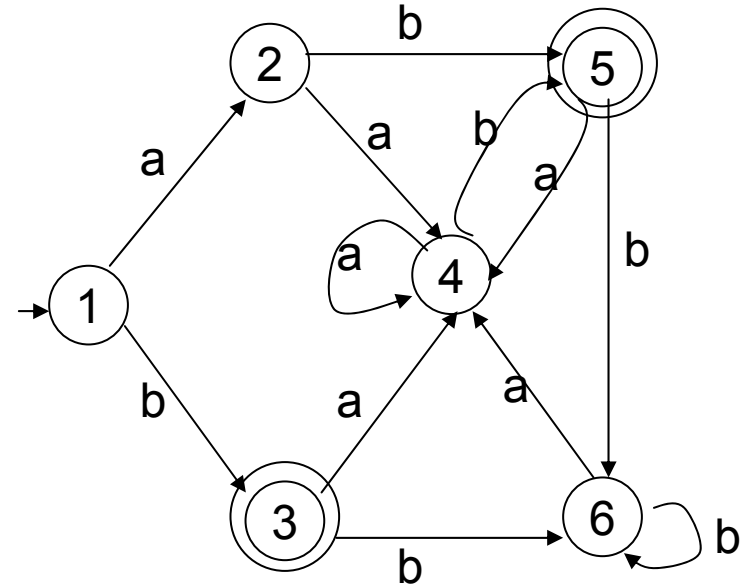
En effet :

$$\begin{aligned}
 w \in L(A / \approx) &\Leftrightarrow \delta_{\approx}([q_0]_{\approx}, w) \in F_{\approx} \\
 &\Leftrightarrow [\delta(q_0, w)]_{\approx} \in F_{\approx} \\
 &\Leftrightarrow \delta(q_0, w) \in F \\
 &\Leftrightarrow w \in L(A)
 \end{aligned}$$

#### Algorithme1 de minimisation d'un AFD $A$

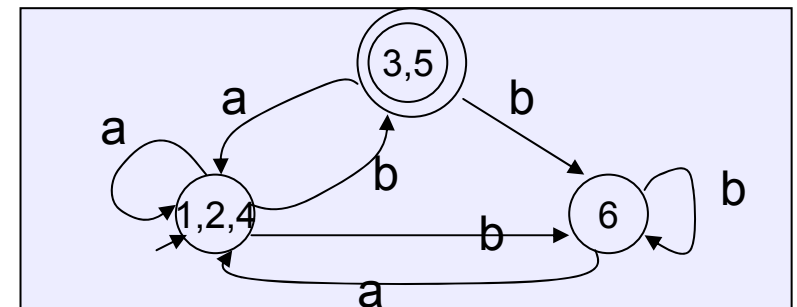
- 1- Éliminer de  $A$  tous les états non accessibles
- 2- Faire deux classes : une classe pour les terminaux et une autre pour les non terminaux.
- 3- S'il existe une lettre  $a$  qui sépare deux états  $p$  et  $q$  ( $\delta(p, a)$  et  $\delta(q, a)$  ne sont pas dans la même classe), alors créer une nouvelle classe et séparer  $p$  et  $q$ . On laisse dans la même classe tous les états qui donnent un état d'arrivée dans la même classe.
- 4- Recommencer en 3 jusqu'à ce qu'il n'y ait plus de classe à séparer.

long0		long1		longueur2			
		a	b	aa	ab	ba	bb
1		2	<u>3</u>	4	<u>5</u>	4	6
2		4	<u>5</u>	4	<u>5</u>	4	6
4		4	<u>5</u>	4	<u>5</u>	4	6
6		4	6				
<u>3</u>		4	6	4	<u>5</u>	4	6
<u>5</u>		4	6	4	<u>5</u>	4	6



Deux états sont séparés si leurs lignes corresp.  
n'ont pas le même soulignement

$\pi_0 : \{1,2,4,6\}, \{3,5\}$   
 $b$  sépare 6 de  $\{1,2,4\}$   
 $\pi_1 : \{1,2,4\}, \{6\}, \{3,5\}$   
 $\pi_2 : \{1,2,4\}, \{6\}, \{3,5\}$





## Algorithme 2 pour minimiser un AFD

Début

//  $A(\text{card}(Q), \text{card}(Q))$  matrice booléenne initialisée à faux

fini  $\leftarrow$  faux;

pour tout  $(p,q) \in Q \times Q$  faire

  si  $(p,q) \in F \times F \cup (Q \setminus F) \times (Q \setminus F)$   $A(p,q) \leftarrow$  vrai

  sinon  $A(p,q) \leftarrow$  faux;

fait;

tant que non fini faire

  fini  $\leftarrow$  vrai;

  pour tout  $(p,q) \in Q \times Q$  faire

    si  $A(p,q) =$  vrai alors

      pour tout  $a \in \Sigma$  faire

        si  $A(\delta(p,a), \delta(q,a)) =$  faux alors  $A(p,q) \leftarrow$  faux; fini  $\leftarrow$  faux fsi;

      fait;

    fsi;

  fait;

fait;

Fin



# Propriétés des langages réguliers

*Soit  $L$  un langage quelconque sur  $\Sigma$ .*

- $L$  est-il régulier ?
- Pour quels opérateurs les langages réguliers sont-ils fermés ?
- $w \in L$  ?
- $L$  est-il vide, fini, infini ?
- $(L_1 \subseteq L_2, L_1 = L_2 ?)$



Pour montrer qu'un langage est régulier,

Il faut :

- trouver un automate fini (ou une expression rég.)  $M$  et
- montrer que  $L = L(M)$  càd
  - que  $L \subseteq L(M)$  (tout mot de  $L$  est accepté par  $M$ )
  - que  $L(M) \subseteq L$  (tout mot accepté par  $M$  est dans  $L$ )

## Lemme de l'étoile ( ou lemme de pompage)

Soit  $L$  un langage sur  $\Sigma$ .

Si  $L$  est régulier alors

$\exists n$  (entier),  $\forall w$  (mot) ( $w \in L \wedge |w| \geq n$ )  $\Rightarrow$

(  $\exists x, y, z \in \Sigma^*$ ,

$$(1) \quad w = x y z$$

$$(2) \quad y \neq \varepsilon \quad \wedge \quad |xy| \leq n$$

$$(3) \quad \forall k \geq 0 \quad x y^k z \in L )$$

## Preuve.

Supposons que  $L$  est régulier. Il existe un AFD (minimal)

$A = (\Sigma, Q, \delta, p_0, F)$  t.q  $L(A) = L$ .

On prend  $n = \text{card}(Q)$ . Soit  $w = a_1 \dots a_i \dots a_m$  un mot de  $L$  de long.  $m \geq n$ .

On note par  $w_i$  le préfixe de  $w$  de longueur  $i$  ( $w_i = a_1 \dots a_i$ )

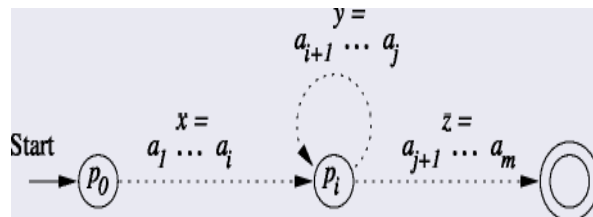
(il y a  $(n + 1)$  préfixes de  $w$  de longueur  $\leq n$ )

Soit  $f : \{0, 1, \dots, n\} \rightarrow Q$  définie par :  $f(i) = \delta(p_0, w_i) = p_i$

- $f$  est non injective ( $\text{card} \{0, 1, \dots, n\} = n + 1 > \text{card}(Q) = n$ )
- Il existe  $i, j \in \{0, 1, \dots, n\}$  t.q  $i \neq j$  et  $f(i) = f(j)$  ( $\delta(p_0, w_i) = \delta(p_0, w_j)$ )

Supposons  $j > i$ . On prend :

$x = w_i = a_1 \dots a_i$ ,  $y = a_{i+1} \dots a_j$  ( $w_j = x y$ ) et  $z = a_{j+1} \dots a_m$  ( $w = w_j z$ )



Pour ces mots  $x$ ,  $y$  et  $z$  on vérifie (facilement) les trois propriétés du lemme.

**Remarque:** le lemme de l'étoile donne une condition nécessaire pour qu'un langage soit régulier. Il est utilisé pour prouver qu'un langage n'est pas régulier. (par contraposée)

### FORMALISONS LE LEMME:

$$\begin{aligned} L \text{ est régulier} &\Rightarrow \\ \exists n : \forall w \in \Sigma^* : & \\ (w \in L \wedge |w| \geq n) & \\ &\Rightarrow \\ (\exists x, y, z \in \Sigma^* : & \\ w = x y z \wedge & \\ (|xy| \leq n \wedge |y| \geq 1) & \\ \wedge & \\ \forall i \geq 0 : x y^i z \in L ) & \end{aligned}$$

### Formalisme du lemme:

$$\begin{aligned} L \text{ est régulier} &\Rightarrow \\ \exists n : \forall w \in \Sigma^* : & \\ \neg (w \in L \wedge |w| \geq n) & \\ \vee & \\ (\exists x, y, z \in \Sigma^* : & \\ w = x y z \wedge & \\ (|xy| \leq n \wedge |y| \geq 1) & \\ \wedge & \\ \forall i \geq 0 : x y^i z \in L ) & \end{aligned}$$

## Formalisme du lemme:

L est régulier  $\Rightarrow$

$\exists n : \forall w \in \Sigma^* :$

$\neg (w \in L \wedge |w| \geq n)$

$\vee$

$(\exists x, y, z \in \Sigma^* :$

$w = x y z \wedge$

$(|xy| \leq n \wedge |y| \geq 1)$

$\wedge$

$\forall i \geq 0 : x y^i z \in L )$

## CONTRAPOSEE

$\forall n : \exists w \in \Sigma^* :$

$(w \in L \wedge |w| \geq n)$

$\wedge$

$\forall x, y, z \in \Sigma^* :$

$\neg [w = x y z \wedge$

$(|xy| \leq n \wedge |y| \geq 1)]$

$\vee$

$\neg (\forall i \geq 0 : x y^i z \in L )$

$\Rightarrow$  L est non régulier



## CONTRAPOSEE

$$\forall n : \exists w \in \Sigma^* :$$

$$(w \in L \wedge |w| \geq n)$$

$\wedge$

$$\forall x, y, z \in \Sigma^* :$$

$$\neg [w = x y z \wedge (|xy| \leq n \wedge |y| \geq 1)]$$

$\vee$

$$(\exists i \geq 0 : x y^i z \notin L)$$

$\Rightarrow$  L est non régulier

## CONTRAPOSEE

$$\forall n : \exists w \in \Sigma^* :$$

$$(w \in L \wedge |w| \geq n)$$

$\wedge$

$$\forall x, y, z \in \Sigma^* :$$

$$[w = x y z \wedge (|xy| \leq n \wedge |y| \geq 1)]$$

$\Rightarrow$

$$(\exists i \geq 0 : x y^i z \notin L)$$

$\Rightarrow$  L est non régulier

Remarque: On peut faire une démonstration par l'absurde en supposant Que L est régulier et la négation du conséquent du lemme.

Exemples :

1)  $L_1 = \{ a^m b^m, m \geq 1 \}$  n'est pas régulier.

Fixons  $n$  et soit  $w = a^n b^n \in L$  ( $|w| = 2n > n$ ).

- $w = x y z = a^n b^n$  et  $|xy| \leq n$  impose que le choix, de  $x$  et de  $y$ , est parmi les 'a'.

$$w = x y z = a^{n-j} a^j b^n \quad (y = a^j) \quad (1 \leq j \leq n)$$

$$x y^k z = a^{n-j} a^{kj} b^n = a^{n+(k-1)j} b^n$$

- Choix de  $k$  pour que  $x y^k z \notin L$ .

Il suffit de prendre  $k > 1$  (par exemple  $k = 2$ ) pour conclure que le langage  $L$  est non régulier.

**Autre démonstration par la congruence Myhill – Nerode :**

On a :  $a^i \sim_{L_1} a^j \quad \forall i \neq j$ .

en effet  $a^i a^k b^{i+k} \in L_1$  ( $\forall K$ ) mais  $a^j a^k b^{i+k} \notin L_1$  du fait que  $j \neq i$ , donc il y a une infinité de classes d'équivalence  $[a^i]$  ( $i \geq 0$ ).

Le langage  $L_1$  est non régulier (selon le théorème de M - Nerode).

2)  $L_2 = \{ a^{n^2}, n \geq 1 \}$  est un langage non régulier.

- Fixons  $n$  et choisissons  $w = a^{n^2}$  avec  $n^2 > n$ .

- $w = x y z = a^{n^2}$

$$= a^{n^2-j} a^j \text{ (où } y = a^j, 1 \leq j \leq n)$$

- $x y^k z = a^{n^2 + (k-1)j}$

prenons  $k = 2$  et montrons que  $x y^2 z \notin L_2$ .

$n^2 < |xy^2z| \leq n^2 + n < (n+1)^2$  ;  $x y^2 z$  ne peut être un carré parfait,  
donc  $x y^2 z \notin L$ .

# Propriétés de fermeture des langages réguliers

**Théorème** : Si  $L$  et  $L'$  sont réguliers alors, sont réguliers :

- union :  $L \cup L'$
- produit :  $LL'$
- étoile :  $L^*$
- complément :  $C_{\Sigma^*} L$
- intersection :  $L \cap L'$
- différence :  $L \setminus L'$
- image miroir :  $L^R$
- image par homomorphisme :  $h(L)$
- image par homomorphisme inverse :  $h^{-1}(L)$

Preuve :

- les 3 premières propriétés (union, produit et étoile) sont par définition des langages réguliers.
- le complément d'un régulier est régulier :  
L régulier  $\Rightarrow$  L est reconnu par un AFD  $A = (\Sigma, Q, \delta, q_0, F) : L(A) = L$   
L'automate  $B = (\Sigma, Q, \delta, q_0, Q \setminus F)$  reconnaît le complément de L.  
(Pour complémenter un automate, tout état non final devient final et vice versa. L'automate (à complémenter) doit être complet).
- Pour l'intersection (comme pour l'union) on fait une construction directe :

**Produit de deux automates :**

$A_i = (\Sigma, Q_i, \delta_i, q_0^i, F_i)$   $i = 1$  et  $2$  sont deux AFDs  $L_1 = L(A_1)$  et  $L_2 = L(A_2)$ .

Le produit de  $A_1$  et  $A_2$  est un AFD  $A_x = (\Sigma, Q_x, \delta_x, q_0^x, F_x)$  où :

$$Q_x = Q_1 \times Q_2, \quad q_0^x = (q_0^1, q_0^2)$$

$$\delta_x \text{ est définie par : } \delta_x((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

Si  $F_x = F_1 \times F_2$  alors  $A_x$  reconnaît  $L_1 \cap L_2$ . ( $L(A_x) = L_1 \cap L_2$ )

Si  $F_x = (F_1 \times Q) \cup (Q \times F_2)$  alors  $A_x$  reconnaît  $L_1 \cup L_2$

-  $L$  régulier  $\Rightarrow L^R$  est régulier

$L$  reconnu par un AFD  $A = (\Sigma, Q, \delta, q_0, F)$ . On construit un  $AFN_\varepsilon$

$B = (\Sigma, Q \cup \{p_0\}, \delta_B, p_0, F_B)$  où :

- $p_0$  est l'état initial ( $p_0 \notin Q$ )
- $\delta_B$  est obtenue en inversant le sens de tous les arcs de  $A$
- ajouter  $(p_0, \varepsilon) = F$
- $F_B = \{q_0\}$

et on a  $L(B) = L^R$

$h : \Sigma^* \rightarrow \Sigma'^*$  un homomorphisme

- $L$  est régulier sur  $\Sigma \Rightarrow h(L)$  est régulier sur  $\Sigma'$   
(Preuve par induction structurelle des réguliers)
- $L'$  est régulier sur  $\Sigma' \Rightarrow h^{-1}(L')$  est régulier sur  $\Sigma$   
 $A' = (\Sigma', Q', \delta', q_0', F')$  un AFD reconnaissant  $L'$ .  
On construit  $A = (\Sigma, Q, \delta, q_0, F)$  où :  
 $\delta(q, a) = \delta'(q, h(a))$   
et on vérifie que  $L(A) = h^{-1}(L')$

# Propriétés décidables des réguliers

$L$ ,  $L_1$  et  $L_2$  sont des réguliers sur  $\Sigma$ .

- $w$  est mot de  $\Sigma^*$ .  
 $w \in L ?$  :  $w \in L \Leftrightarrow w$  est accepté par un AF  
A t.q.  $L = L(A)$ .
- $L = \emptyset ?$  :  $L \neq \emptyset \Leftrightarrow \exists$  un état final est accessible  
de l'état initial. ( $L = L(A)$ )
- $L$  fini ou infini ? :  $L$  est infini  $\Leftrightarrow \exists$  un chemin de l'état  
initial à un état final passant par un cycle.
- $L_1 \subseteq L_2 ?$  :  $L_1 \subseteq L_2 \Leftrightarrow L_1 \cap C_{\Sigma^*}(L_2) = \emptyset$ .
- $L_1 = L_2 ?$  :  $L_1 \subseteq L_2$  et  $L_2 \subseteq L_1$