Informatique théorique 2
Langages Formels & Automates

J. Bond

bond@unice.fr
☎ 04 92 96 51 41

Erick Gallesio

erick.gallesio@unice.fr
☎ 04 92 96 51 53

Les copies des transparents et des TD seront disponible sur slack

Langages

- En mathématiques, en informatique et en linguistique, la théorie des langages a pour objectif de décrire les langages formels. Un langage formel est un ensemble de mots. L'alphabet d'un langage formel est l'ensemble des symboles, lettres ou lexèmes qui servent à construire les mots du langage ; on suppose que cet alphabet est fini.

Langages

- Les mots sont des suites d'éléments de cet alphabet ; les mots qui appartiennent à un langage formel particulier sont parfois appelés mots bien formés ou formules bien formées. Un langage formel est souvent défini par une grammaire formelle et analysé par des automates.

Langages

- La théorie des langages étudie les aspects purement syntaxiques de tels langages, c'est-à-dire leur structure interne formelle.
- En informatique, les langages formels sont souvent utilisés comme base pour la définition des langages de programmation et d'autres systèmes ; les mots d'un langage comportent alors aussi un sens, une sémantique.

Langages

- En théorie de la complexité des algorithmes, les problèmes de décision sont généralement définis comme des langages formels.
- En logique mathématique, les langages formels sont utilisés pour représenter la syntaxe des systèmes axiomatiques, et l'attitude formaliste en logique affirme qu'en principe, les mathématiques peuvent se ramener à la manipulation syntaxique de langages formels.

Langages

- L'étude des langages formels comporte l'ensemble des moyens de description et d'analyse de ces langages, comme les grammaires formelles pour la génération et les automates pour la reconnaissance, mais elle s'intéresse aussi à l'apprentissage des langages et à leur traduction. Dans le domaine de la traduction, la théorie des langages s'applique aux compilateurs de langages de programmation.

Langages

- On se donne un ensemble Σ, appelé alphabet dont les éléments sont appelés des lettres.
- Un mot de longueur k est une suite

  $u = (a_1, a_2, \ldots, a_k)$ de k lettres. En pratique, on utilise la notation condensée $u = a_1 a_2 \ldots a_k$
- L'ensemble des mots sur l'alphabet Σ est noté Σ^*.
- Le mot vide, de longueur 0, est noté ε.

Langages

- On définit sur Σ^*, une loi de composition interne appelée concaténation. Elle associe à deux mots $a_1 \ldots a_n$ et $b_1 \ldots b_m$ le mot $a_1 \ldots a_n b_1 \ldots b_m$ (de longueur $n+m$).
- Cette loi de composition interne est associative et admet le mot vide pour élément neutre. Par conséquent l'ensemble Σ^*, muni de cette loi, est un monoïde.

Langages

- Un langage formel est un ensemble de mots sur un alphabet fini, c'est-à-dire une partie du monoïde libre sur cet alphabet.

Exemples de langages formels

- l'ensemble de tous les mots sur $\Sigma = \{a,b\}$
- l'ensemble des mots de la forme a^n, où n est un nombre premier,
- l'ensemble des programmes syntaxiquement corrects dans un langage de programmation donné,
- l'ensemble des mots d'entrée sur lesquels une machine de Turing donnée s'arrête,
- l'ensemble des 1000 mots les plus fréquents dans une langue donnée.

Automate fini

- Un AF est un quintuplet

$$A=(\Sigma, Q, \delta, q_0, F)$$

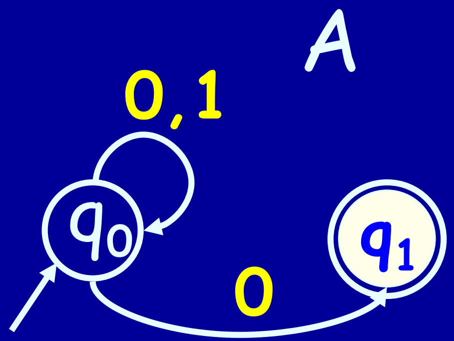
1. Σ : un alphabet fini (symboles du ruban)
2. Q : un ensemble fini d'états
3. δ : fonction de transition (règle de changement d'état)

$$\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$$

4. $q_0 \in Q$: état initial
5. $F \subseteq Q$: états de reconnaissance (ou finals)

Exemple d'automate fini

$A = (\Sigma, Q, \delta, q_0, F)$



$\Sigma = \{0,1\}$
$Q = \{q_0, q_1\}$
$\delta = \{(q_0, 0, q_0), (q_0, 0, q_1), (q_0, 1, q_0)\}$
q_0 est l'état initial
$F = \{q_1\}$

A accepte $m \in \Sigma^*$ s'il existe un chemin de q_0 à un état de F étiqueté par les lettres de m.

A reconnaît le langage $L(A)$ décrit par l'expression rationnelle : $(0+1)^*0$

Automates finis déterministes

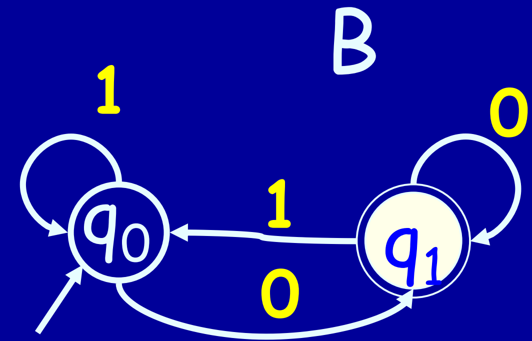
- Un automate fini est **déterministe** (**Deterministic Finite Automaton ou DFA**) si et seulement si δ est une fonction de transition telle que :

$$\delta : \; Q \times \Sigma \rightarrow Q$$

- D'un état donné, il part **au plus** une transition étiquetée par une lettre donnée.

Exemple d'automate fini déterministe

$B = (\Sigma, Q, \delta, q_0, F)$



$\Sigma = \{0,1\}$

$Q = \{q_0,q_1\}$

$\delta = \{(q_0, 1, q_0),(q_0, 0, q_1), (q_1, 1, q_0), (q_1, 0, q_1)\}$

q_0 est l'état initial

$F = \{q_1\}$

B accepte $m \in \Sigma^*$ s'il existe un chemin de q_0 à un état de F étiqueté par les lettres de m.

B reconnaît le langage L(B) décrit par l'expression rationnelle : $(0+1)^*0$

Autres représentations des automates

- déterministes

	0	1
$\rightarrow q_0$	q_1	q_0
$\leftarrow q_1$	q_1	q_0

- non-déterministes

	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	q_0
$\leftarrow q_1$	-	-

- δ : $(q_0,0)\rightarrow q_1$
  $(q_0,1)\rightarrow q_0$
  $(q_1,0)\rightarrow q_1$
  $(q_0,1)\rightarrow q_0$

Automate fini non déterministe

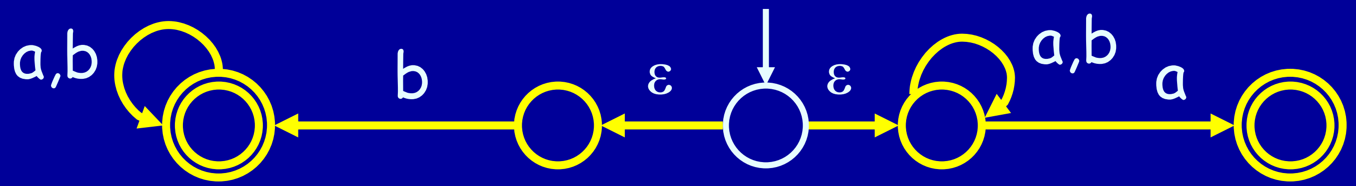
- Du même état, on peut avoir plusieurs transitions étiquetées par la même lettre
  - ce n'est plus une fonction mais
    - une relation $\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$
    - une fonction avec un autre co-domaine

$$\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow \mathbf{P}(Q)$$

- On peut avoir des transitions vides appelées

ε-transitions

- On peut avoir plusieurs états initiaux



différences

déterministe	non-déterministe
▪ fonction de transition	▪ relation de transition
▪ un seul état initial	▪ plusieurs états initiaux
▪ pas d'ε-transitions	▪ ε-transitions possibles
▪ unicité de la lecture	▪ pluralité de lecture
$\delta : Q \times \Sigma \rightarrow Q$	$\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$
	$\delta : Q x \Sigma \cup \{\varepsilon\} \rightarrow P(Q)$

Lecture/calcul de l'automate

- **réussie** ou **ratée**
- lecture = suite des états pris par l'AFD
  - commence par l'état initial
  - termine
    - par un état de reconnaissance ➡ **réussie**
    - par un autre état ➡ **ratée**
  - ne termine pas (pas de transition applicable) ➡ **ratée**

Formalisation: configuration

- Représente
  - l'état courant
  - la partie du mot qui reste à lire

    (état courant, suffixe à lire)$\in Q \times \Sigma^*$
- application d'une transition modifie la configuration: dérivation

$$(q,a.w) \rightarrow^{a} (q',w) \quad \text{si } (q,a,q') \subseteq \delta$$

$$(q,a.w) \rightarrow^{\varepsilon} (q',a.w) \quad \text{si } (q,\varepsilon,q') \subseteq \delta$$

lecture = suite des configurations

- On étend la notion de configuration aux mots:

$$(q,w) \rightarrow^* (q',\varepsilon)$$

- w reconnu si A termine la lecture dans un état final
- w pas reconnu si

  1) w a été entièrement lu et $q' \notin F$

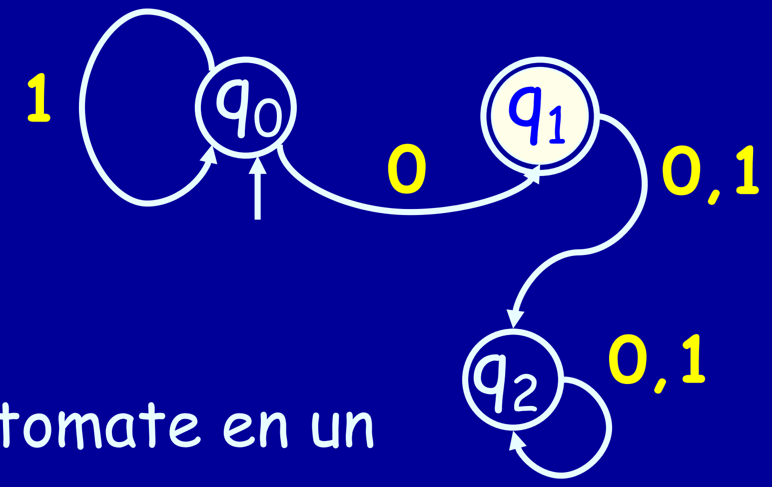
  2) w n'a pas été entièrement lu et plus de dérivation possible

Automates finis complets

Un automate fini (déterministe) est ***complet*** ssi δ est une <u>fonction totale</u> sur $Q \times \Sigma$.

De chaque état, part <u>exactement</u> une flèche étiquetée par chacune des lettres de Σ.

état	0	1
q_0	q_1	q_0
q_1	q_2	q_2
q_2	q_2	q_2



On peut toujours transformer un automate en un automate complet sans modifier le langage reconnu

Exemple

1	0	0	1	0

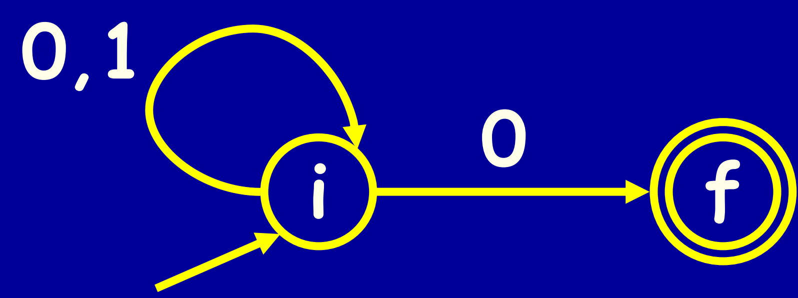
un mot à reconnaître

(i,10010) une configuration initiale

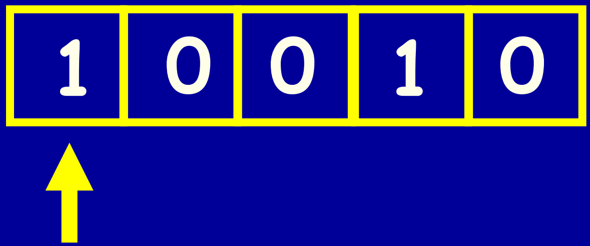
sa table de transition

	0	1
$\rightarrow$i	{i,f}	i
$\leftarrow$f	-	-

un AFND

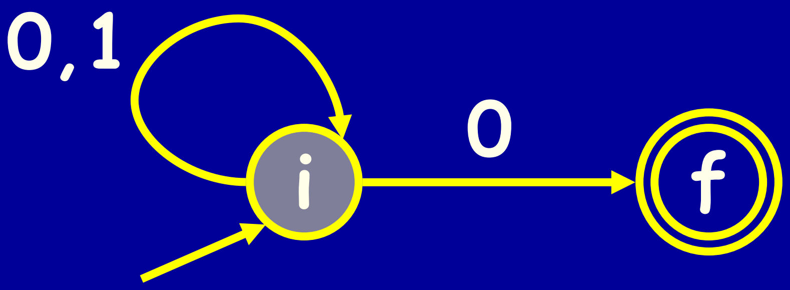


Exemple



(i,10010)

	0	1
→i	{i,f}	i
←f	-	-

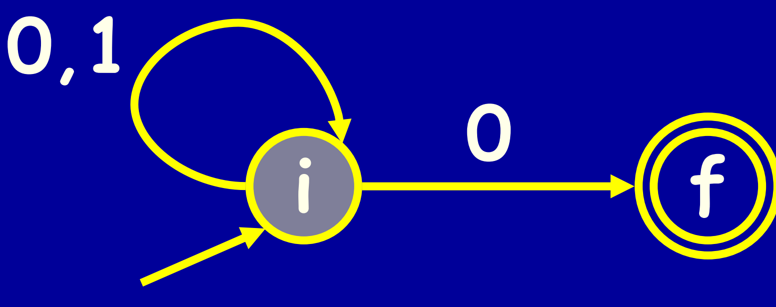


Exemple

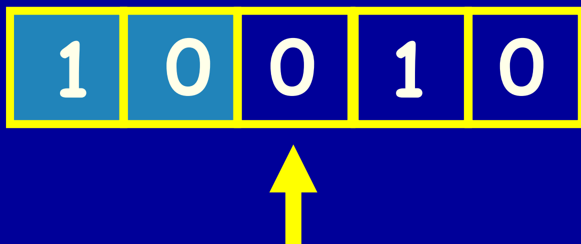
1	0	0	1	0

(i,10010)→(i,0010)

	0	1
→i	{i,f}	i
←f	-	-



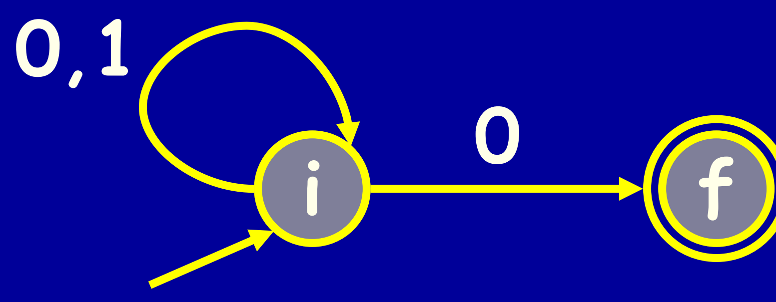
Exemple



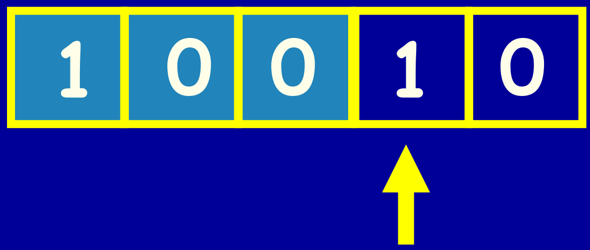
(i,10010)→(i,0010)→(i,010)

→(f,010)

	0	1
→i	{i,f}	i
←f	-	-



Exemple

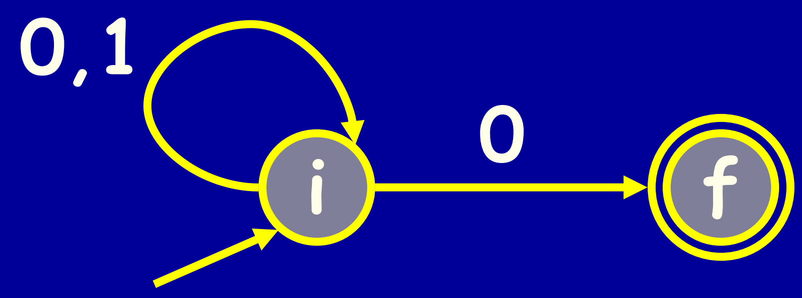


$(i,10010) \to (i,0010) \to (i,010) \to (f,10)$

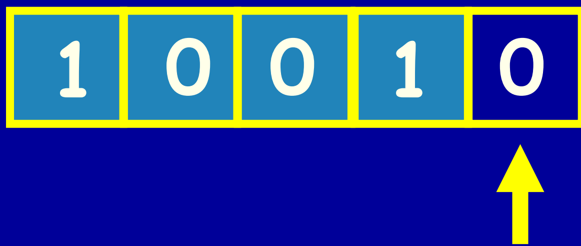
$(i,10010) \to (i,0010) \to (i,010) \to (i,10)$

$(i,10010) \to (i,0010) \to (f,010) \to \varnothing$

	0	1
$\to i$	$\{i,f\}$	i
$\leftarrow f$	-	-



Exemple

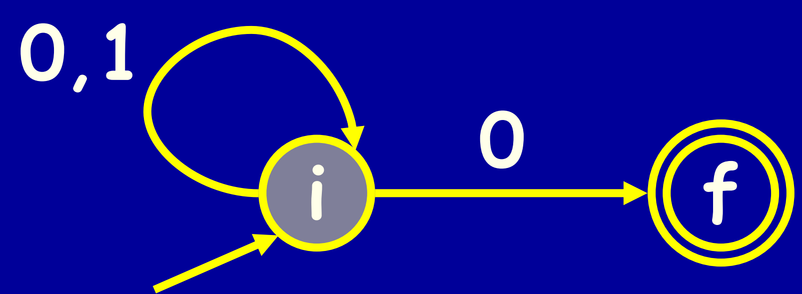


→(f,10) →ø

(i,10010)→(i,0010)→(i,010) →(i,10) →(i,0)

→(f,010)→ø

	0	1
→i	{i,f}	i
←f	-	-



Exemple

1	0	0	1	0

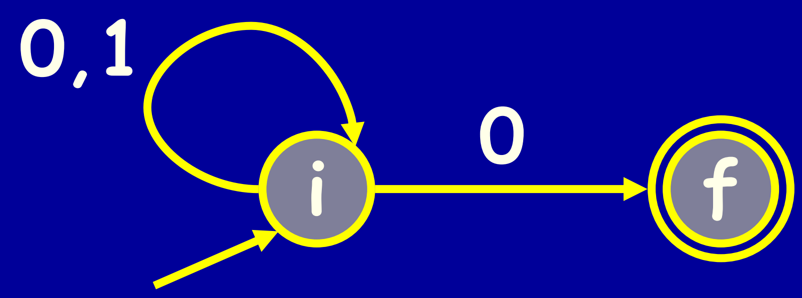
$\rightarrow(f,10)\rightarrow\varnothing$

$(i,10010)\rightarrow(i,0010)\rightarrow(i,010)\rightarrow(i,10)\rightarrow(i,0)\rightarrow(i,\varepsilon)$

$\rightarrow(f,010)\rightarrow\varnothing$

$\rightarrow(f,\varepsilon)$ **accepté**

	0	1
$\rightarrow i$	$\{i,f\}$	i
$\leftarrow f$	-	-



Equivalence AFD et AFND

Théorème : Tout langage reconnu par un automate fini l'est par un automate fini déterministe.

Preuve : si l'automate de départ est déterministe, évident;

Si l'automate de départ est non-déterministe :

- construire un AFD qui intègre tous les choix du non déterministe (**algorithme de déterminisation**)
- prouver que l 'AFD reconnaît le même langage

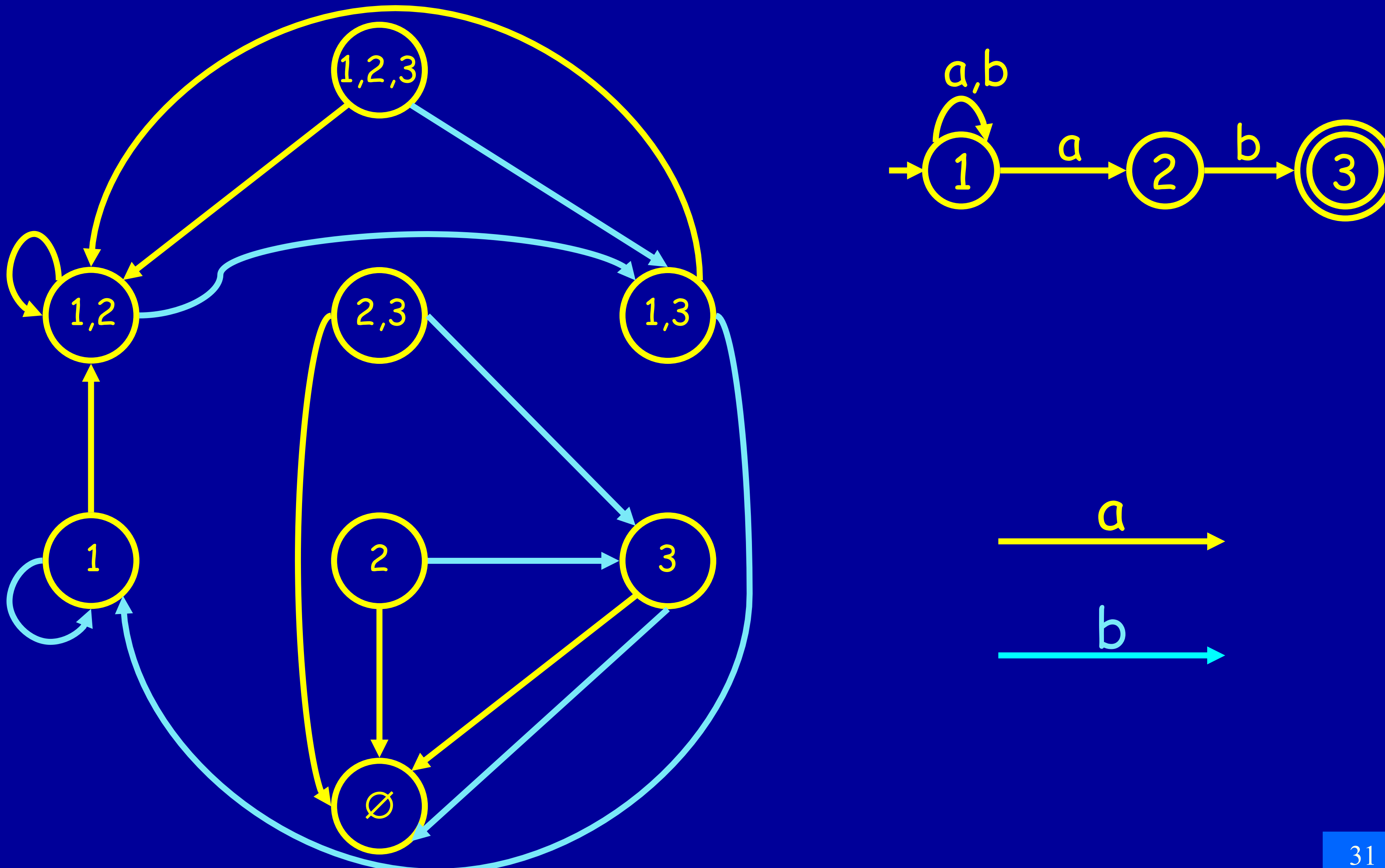
Principe

- Simulation fonctionnement AFND : mémoriser dans quel ensemble d'états on est et, en lisant une lettre, voir dans quel ensemble d'états on arrive.

- C'est exactement ce que va faire un AFD qui simule un AFND

⇒ état AFD=ens états AFND

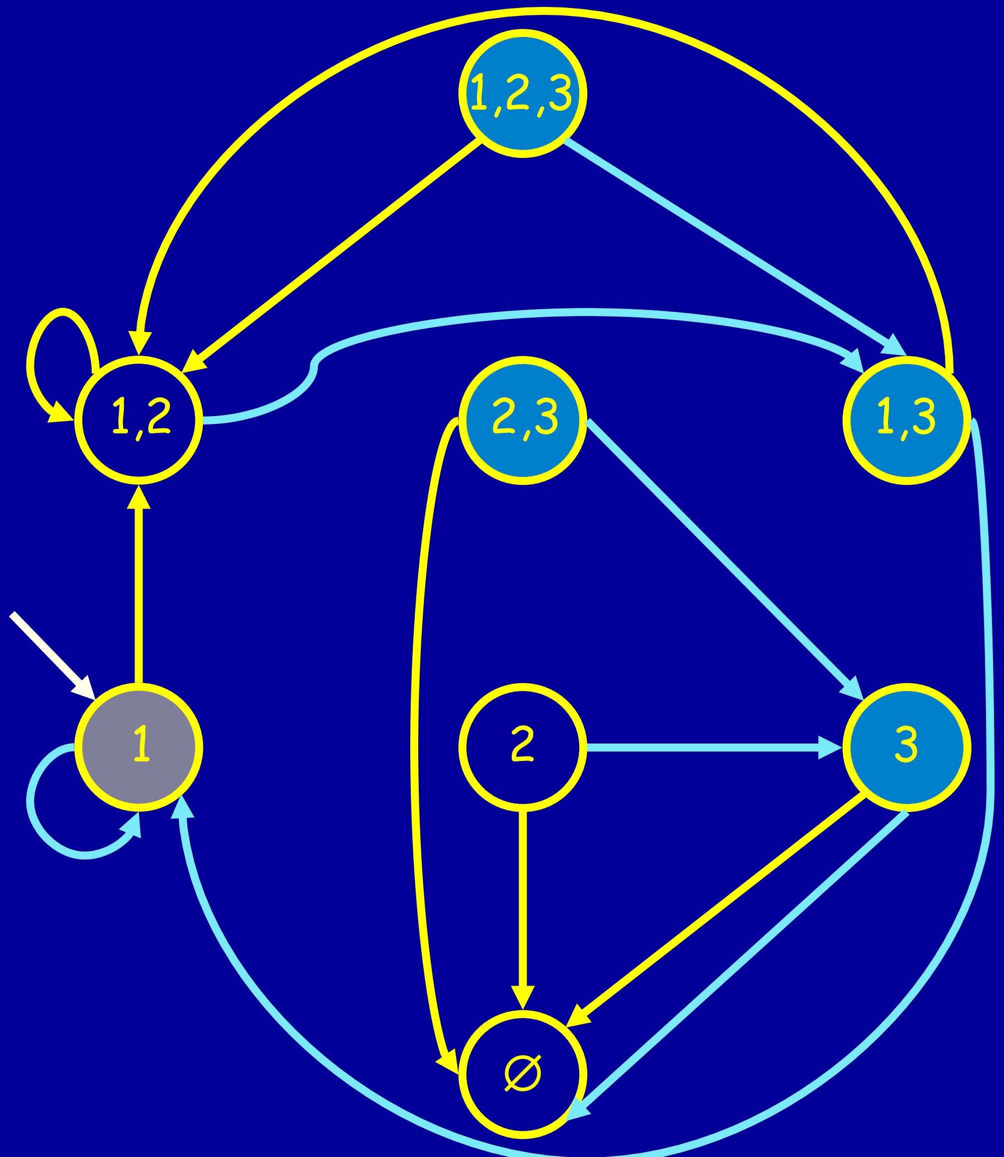
Illustration

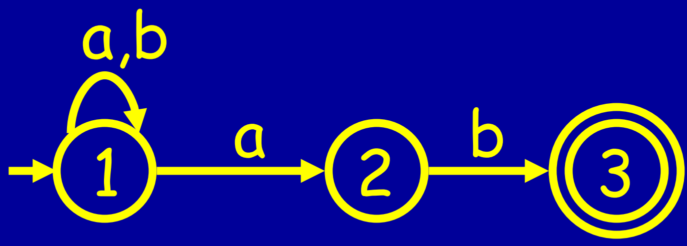


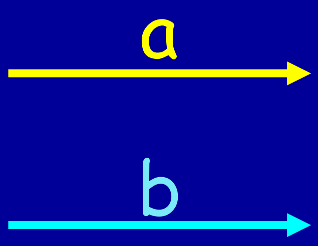
Tout ne sert pas...

- états utiles : accessibles
  - q accessible s'il existe une suite de dérivations de l'initial vers q
- états particuliers : initial et finaux
  - initial : le même que celui (ceux) de l'AFND
  - finals : toutes les parties contenant un état de reconnaissance de l'AFND

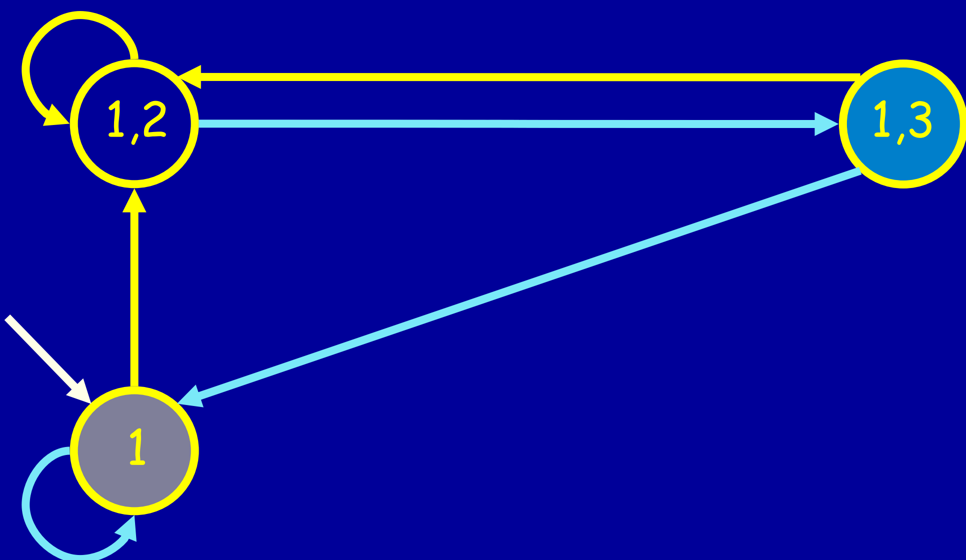
Exemple

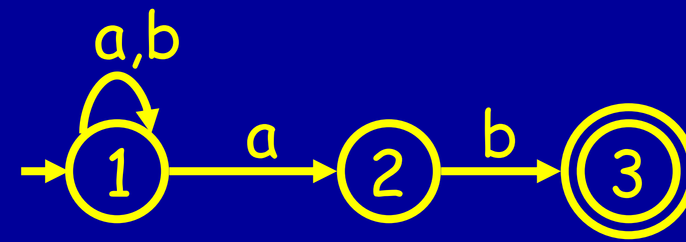


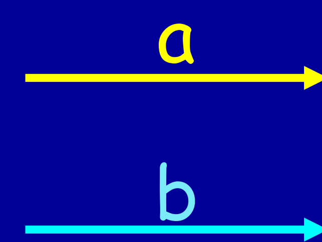


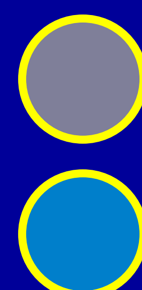


Etats utiles









état initial

états finaux

Structure de la preuve

- AFND $N = (\Sigma, Q, q_0, \delta, F) \rightarrow$
  AFD $A = (\Sigma, Q' \subseteq \mathbf{P}(Q), q_0', \delta, F')$
- $q_0' = E(q_0)$
- $\delta(R,a) = \{q \in Q \mid q \in E(\delta(r,a))$ pour $r \in R\}$
  $\delta(R,a) = \cup_{r \in R} E(\delta(r,a))$
- $F' = \{R \mid R \in Q'$ et $\{f\} \in R, \{f\} \in F\}$
- où $E(R) := \{q \mid q$ accessible de R en suivant 0 ou plusieurs ε-transitions$\}$ (ε-clôture)

Cas sans ε-transition

- $q_0' = \{q_0\}$

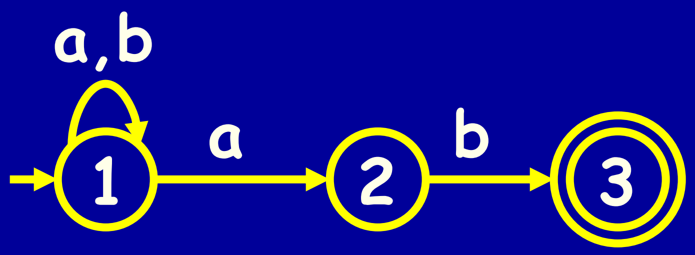
- Pour $R \in Q'$ et $a \in \Sigma$,

- $\delta(R,a) = \{q \in Q \mid q \in \delta(r,a)$ pour $r \in R\}$
  $\delta(R,a) = \cup_{r \in R} \delta(r,a)$

- $F' = \{R \in Q' \mid \{f\} \in R,\ f \in R\}$

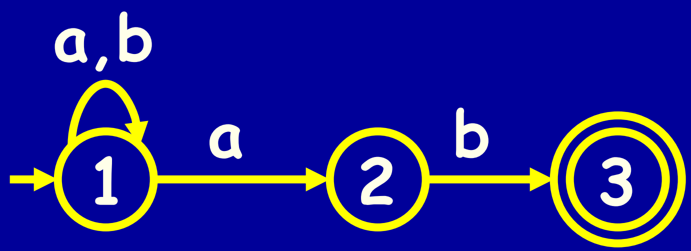
L(N) = (a+b)*ab

	a	b
1	{1,2}	1
2	-	3
3	-	-



$L(N) = (a+b)^*ab$

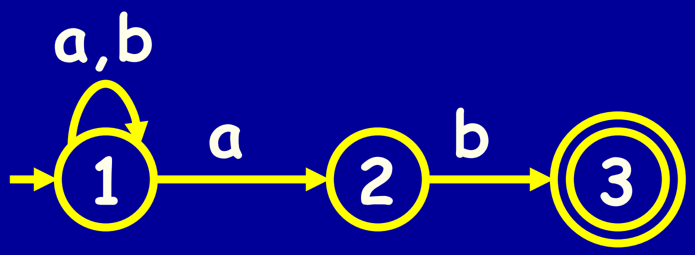
	a	b
$\rightarrow 1$	{1,2}	1
2	-	3
$\leftarrow 3$	-	-
$\rightarrow$ {1}	{1,2}	{1}



nouvel état

$L(N) = (a+b)^*ab$

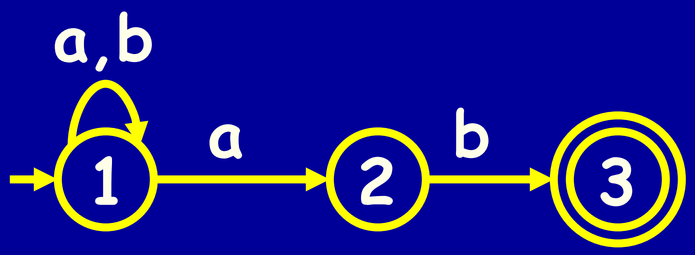
	a	b
$\rightarrow 1$	{1,2}	1
2	-	3
$\leftarrow 3$	-	-
$\rightarrow$ {1}	{1,2}	{1}
{1,2}	{1,2}	{1,3}



nouvel état

L(N) = (a+b)*ab

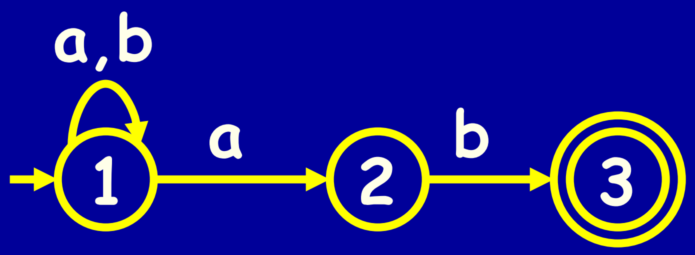
	a	b
→1	{1,2}	1
2	-	3
←3	-	-
→{1}	{1,2}	{1}
{1,2}	{1,2}	{1,3}
←{1,3}	{1,2}	{1}

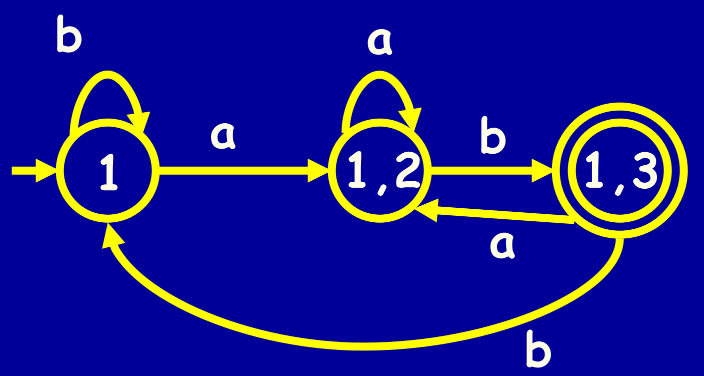


car 3 terminal dans AFND

$L(N) = (a+b)^*ab$

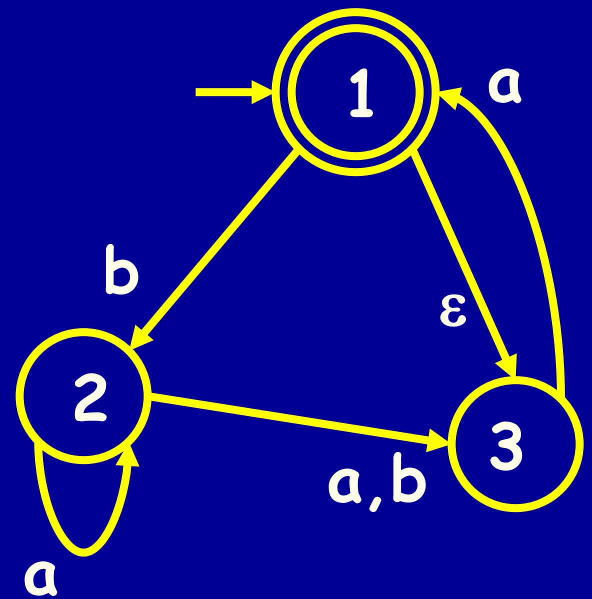
	a	b
→1	{1,2}	1
2	-	3
←3	-	-
→{1}	{1,2}	{1}
{1,2}	{1,2}	{1,3}
←{1,3}	{1,2}	{1}





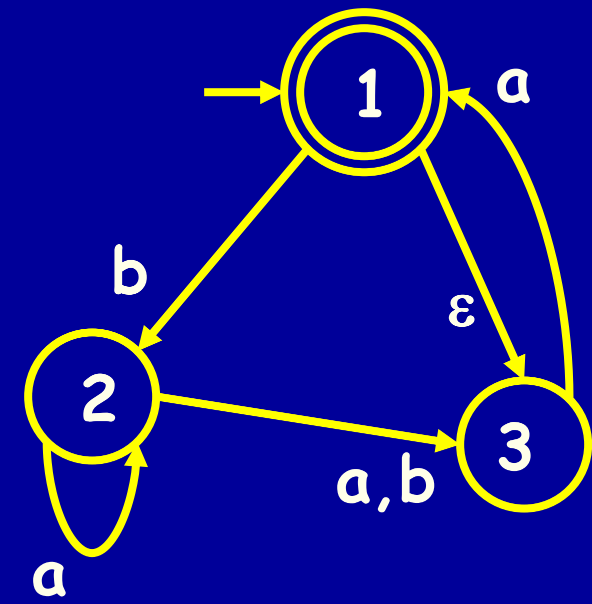
Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-



Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}			

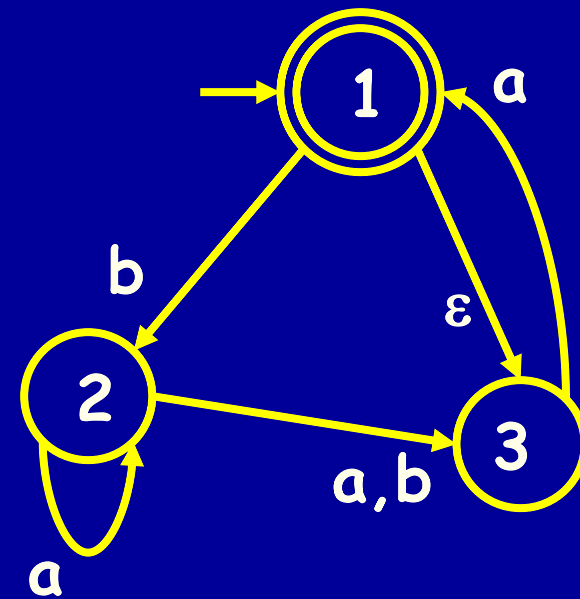


E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	E({1})	E({2})	

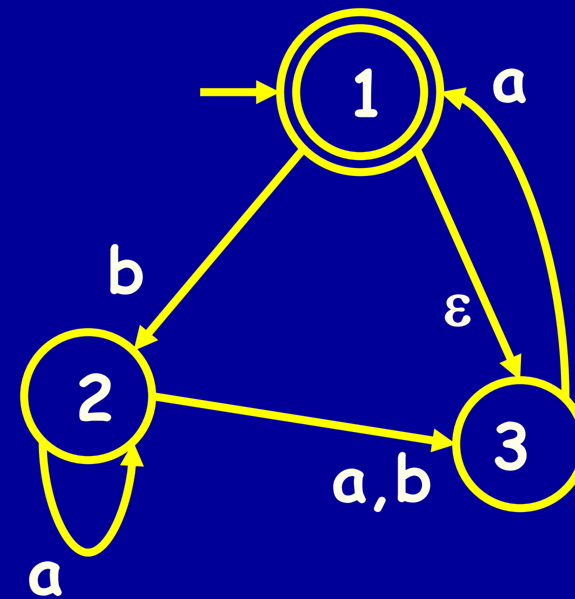


E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	



$E(\{1\})=\{1,3\}$

$E(\{2\})=\{2\}$

$E(\{3\})=\{3\}$

$E(\{1,2\})=\{1,2,3\}$

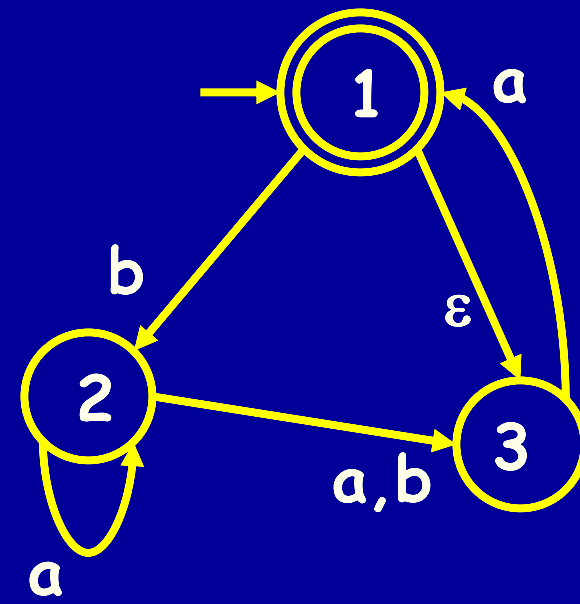
$E(\{2,3\})=\{2,3\}$

$E(\{1,3\})=\{1,3\}$

$E(\{1,2,3\})=\{1,2,3\}$

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	E({2,3})	E({3})	

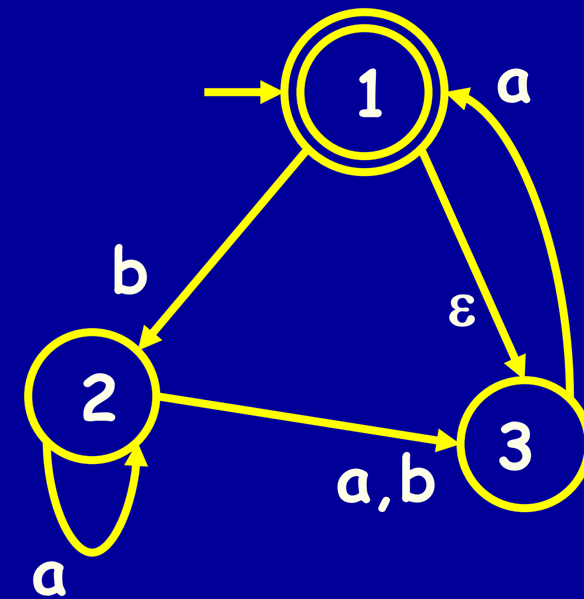


E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	

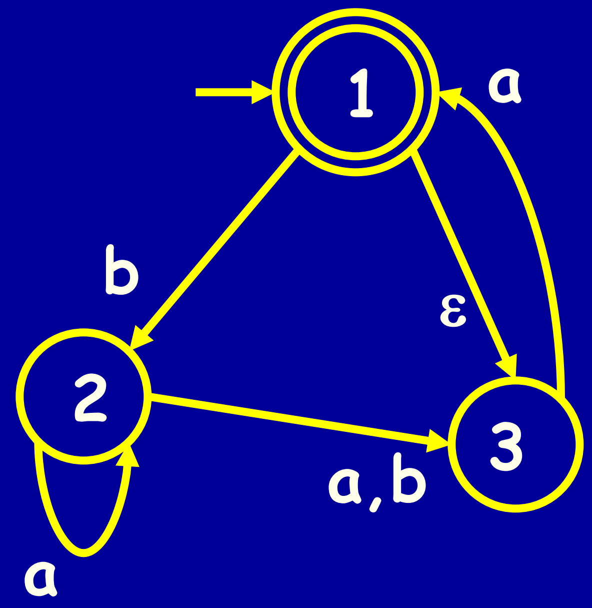


E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

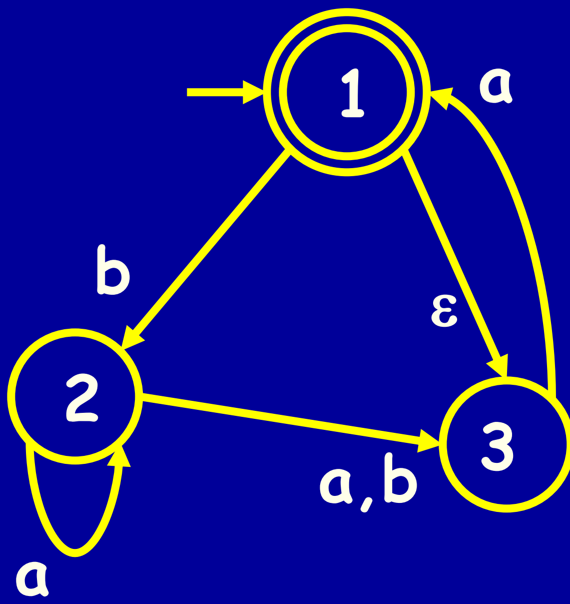
	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	
{2,3}	E({1,2,3})	E({3})	



E({1})={1,3}
E({2})={2}
E({3})={3}
E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	
{2,3}	{1,2,3}	{3}	



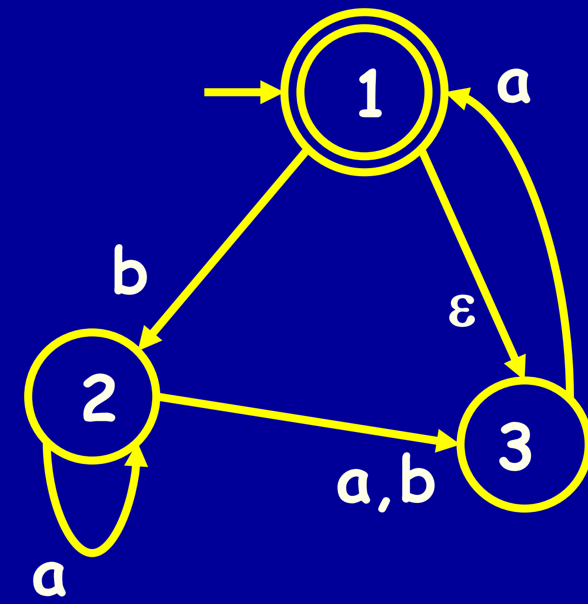
E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-

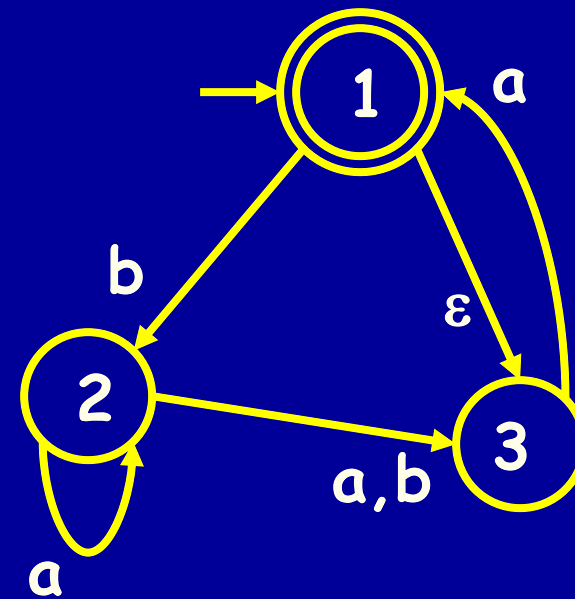
	a	b
↔{1,3}	{1,3}	{2}
{2}	{2,3}	{3}
{2,3}	{1,2,3}	{3}
{3}	E({1})	-



E({1})={1,3}
E({2})={2}
E({3})={3}
E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	
{2,3}	{1,2,3}	{3}	
{3}	{1,3}	-	

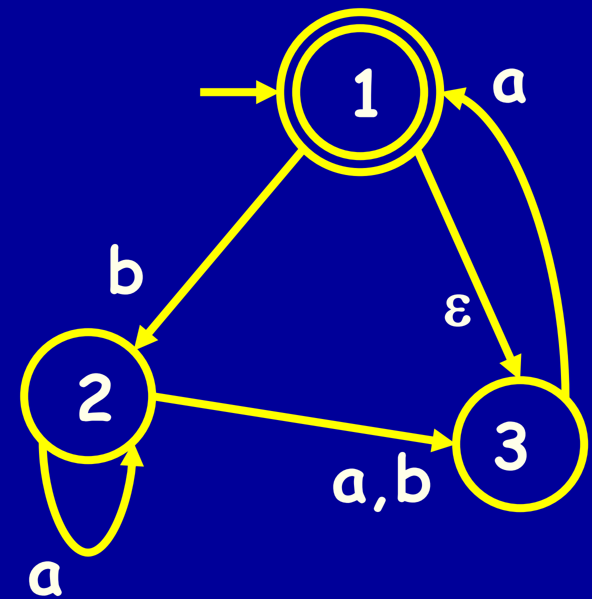


E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

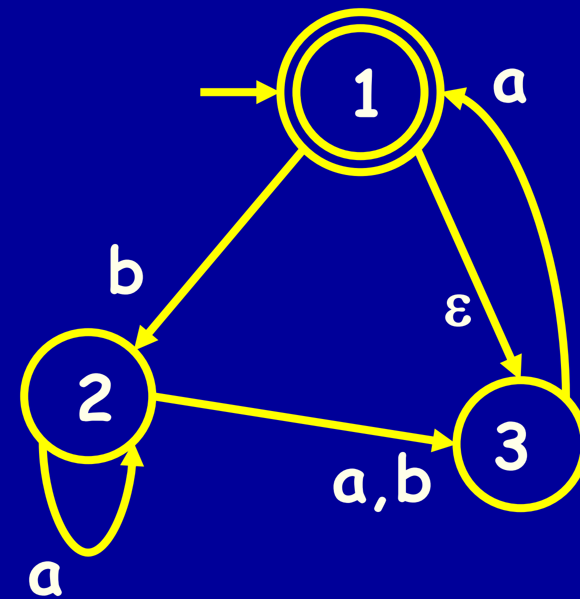
	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	
{2,3}	{1,2,3}	{3}	
{3}	{1,3}	-	
{1,2,3}	E({1,2,3})	E({2,3})	



E({1})={1,3} E({1,2})={1,2,3}
E({2})={2} E({2,3})={2,3}
E({3})={3} E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	
{2,3}	{1,2,3}	{3}	
{3}	{1,3}	-	
{1,2,3}	{1,2,3}	{2,3}	



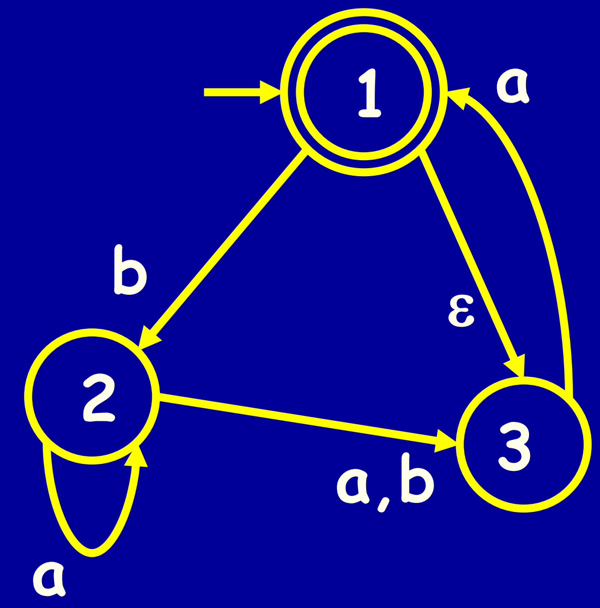
E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-

	a	b
↔{1,3}	{1,3}	{2}
{2}	{2,3}	{3}
{2,3}	{1,2,3}	{3}
{3}	{1,3}	-
←{1,2,3}	{1,2,3}	{2,3}



E({1})={1,3}
E({2})={2}
E({3})={3}

E({1,2})={1,2,3}
E({2,3})={2,3}
E({1,3})={1,3}
E({1,2,3})={1,2,3}

Exemple avec ε-transition

	a	b	ε
↔1	-	2	3
2	{2,3}	3	-
3	1	-	-
↔{1,3}	{1,3}	{2}	
{2}	{2,3}	{3}	
{2,3}	{1,2,3}	{3}	
{3}	{1,3}	-	
←{1,2,3}	{1,2,3}	{2,3}