X5I0050 - Langages et automates
Les automates finis
ou machines avec un nombre fini d'états

D. Béchet & T. Sadiki

Université de Nantes & Université Internationale de Rabat

20 octobre 2014

Introduction

Machine de Turing
Automates finis ou machines avec un nombre fini d'états

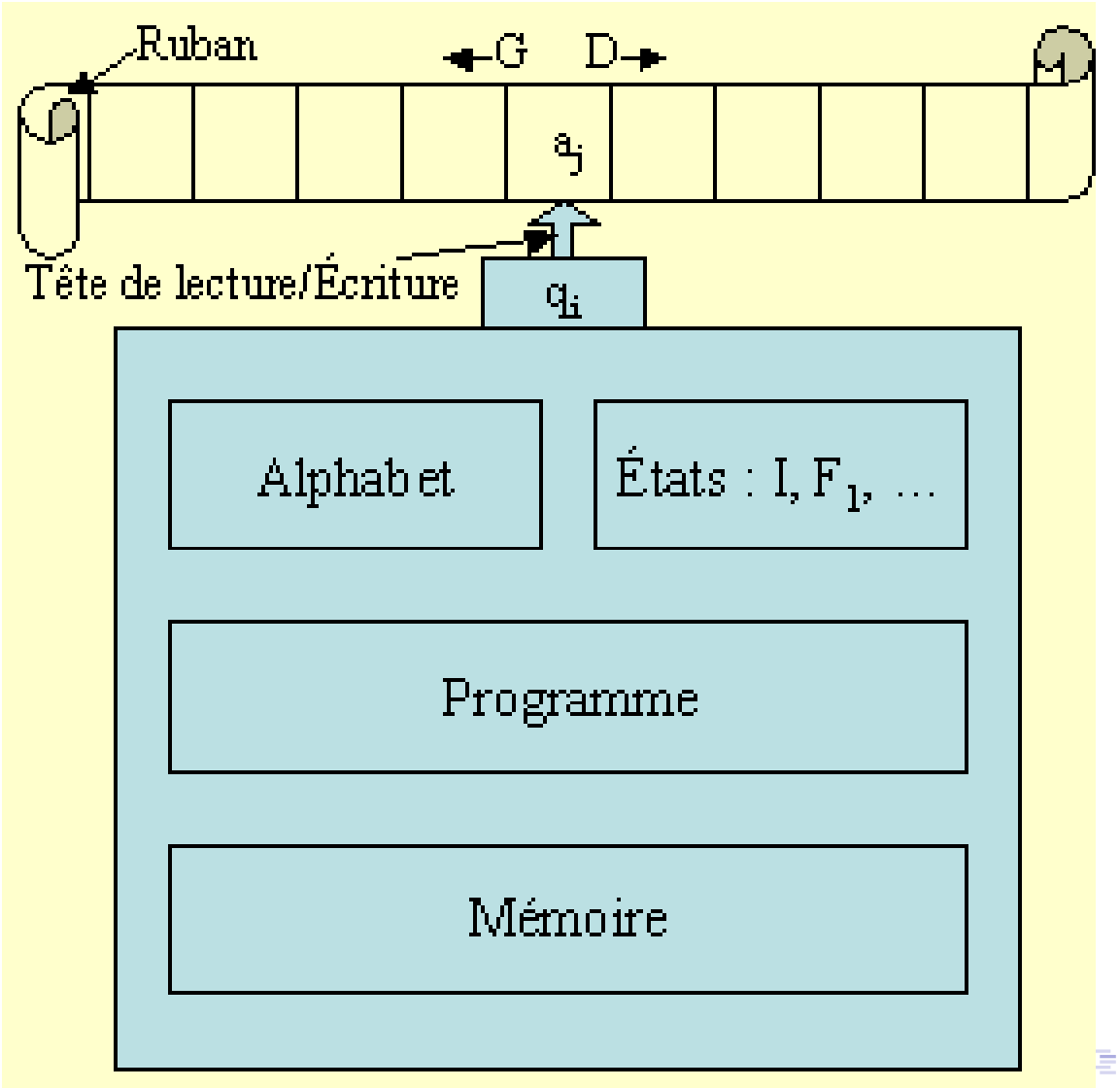
Intérêt
Formalisation du concept d'algorithme et de calculabilité
Structure des automates
Intérêt pour la théorie des langages :
- Machine de Turing : modèle pour les langages récursivement énumérables
- Automate fini : modèle pour les langages rationnels

Plan Chapitre 4

1. La machine de Turing
2. Définition des automates finis
3. Les automates finis généralisés
4. AFN et langages
5. Propriétés des AFNs
6. Rendre un automate déterministe (AFD)
7. Minimisation d'un AFD
8. Combinaisons d'AFNs

Présentation de la machine de Turing

Structure d'une machine de Turing



Règles de transition

États : $Q = \{q_1, q_2, \ldots, q_n\}$
Alphabet : $A = \{a_0, a_1, a_2, \ldots, a_m\}$
Règles de transition : $(q_i, a_j, q_k, a_l, d_m)$ ou $\mu(q_i, a_j) \mapsto (q_k, a_l, d_m)$ avec :

- $q_i \in Q$ est l'état initial ;
- $a_j \in A$ le symbole lu ;
- $q_k \in Q$ le nouvel état ;
- $a_l \in A$ le symbole à écrire ;
- $d_m \in \{D, G, C\}$ le sens du déplacement de la tête avec respectivement D pour aller d'une case vers la droite, G d'une case vers la gauche et C pour ne pas se déplacer.

Définition des configurations

Définition 4.1.a - configuration

On appelle **configuration** tout couple symbole / état. Une règle détermine donc pour une configuration courante, l'action à effectuer et donc la prochaine configuration.

Définition 4.1.b - configuration initiale

On appelle **configuration initiale** tout couple symbole / état où l'état est un état initial.

Définition 4.1.c - configuration finale (ou terminale)

On appelle **configuration finale** tout couple symbole / état où l'état est un état final.

Définition d'une machine de Turing déterministe

Définition 4.2 - Machine de Turing déterministe

La machine de Turing est dite **déterministe** si pour toute configuration, il n'existe qu'une seule transition applicable, c'est-à-dire si et seulement si, si $\mu(x, y) \mapsto (z_1, t_1, d_1)$ et $\mu(x, y) \mapsto (z_2, t_2, d_2)$, alors $z_1 = z_2$, $t_1 = t_2$ et $d_1 = d_2$ (une règle au plus par configuration).
Dans le cas contraire, la machine est dite **non déterministe** (pour une configuration donnée, plusieurs règles sont applicables).

Définition d'une machine de Turing

Définition 4.3 - Machine de Turing

Une machine de Turing est définie par un quintuplet $(\mathbf{A}, \mathbf{Q}, \mathbf{I}, \mathbf{F}, \mu)$ tel que :

- **A** est l'**alphabet** (fini, non vide) ;
- **Q** est l'ensemble des **états** possibles pour la machine (fini, non vide) ;
- $\mathbf{I} \in Q$ l'**état initial** ;
- $\mathbf{F} \subseteq Q$ l'ensemble des **états finaux** ;
- μ la **fonction de transition** telle que $\mu : Q \times A \rightarrow Q \times A \times \{D, G, C\}$. $\mu(q_i, a_j) \mapsto (q_k, a_l, d_m)$ signifie qu'on effectue les actions de remplacement (de a_j par a_k), de déplacement (à gauche, à droite ou au centre suivant d_m) et de changement d'état en q_k pour la configuration courante d'état q_i et de symbole a_j.

Reconnaissance d'un mot par une machine de Turing

Définition - Mot reconnu par une machine de Turing

Un mot m est **reconnu** par une machine de Turing M (sur état final) si à partir d'un ruban sur lequel le mot m est placé, la machine de Turing M, partant de la configuration où le symbole lu est le premier symbol de m et où l'état est l'état initial, peut arriver dans une configuration avec un état final par une suite de transitions

Définition - Langage reconnu par une machine de Turing

L'ensemble des mots reconnus par une machine de Turing M forme le **langage reconnu** par cette machine noté **L(M)A**

Classe des langages reconnus par une machine de Turing

Propriétés - Langage reconnu par les machine de Turing

Les langages reconnus par une machine de Turing sont les langages de type 0 (langages récursivement énumérables)

Démonstration :

1. Une machine de Turing peut-être simulée par une grammaire de type 0 : une transition correspond globalement à une dérivation directe à l'envers. L'emplacement de la tête de lecture/écriture et l'état de la machine de Turing sont codé dans la chaine de symboles de la dérivation
2. Les étapes d'une dérivation (à l'envers) de m vers l'axiome S d'une grammaire de type 0 peuvent être simulées par une machine de Turing (qui ne dépend que de la grammaire)

Définition des automates finis

Définition 4.4 - Automate fini - Machine avec un nombre fini d'états - AFN

Un **automate fini** (**AFN**) est défini par un quintuplet $\mathbf{A}, \mathbf{Q}, \mathbf{I}, \mathbf{F}, \mu$ tel que :

- **A** un alphabet, ensemble fini, non vide de symboles
- **Q** l'ensemble des états possibles pour la machine (fini et non vide) ;
- **I** $I \subseteq Q$
- **F** $F \subseteq Q$ l'ensemble des états finaux ou états d'acceptation ;
- μ la fonction de transition telle que : $\mu : A \cup \{\varepsilon\} \times Q \rightarrow Q$. Pour la configuration courante d'état et de symbole, $\mu(a_j, q_i) \mapsto q_k$ signifie qu'on passe dans l'état q_k et que le ruban est déplacé d'une case vers la gauche.

Un AFN est une sorte de machine de Turing qui avance toujours dans le même sens et ne modifie jamais le ruban

Représentation matricielle d'un AFN

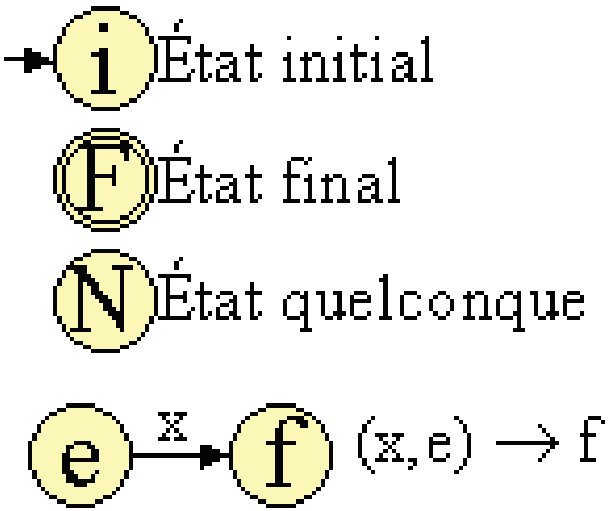
$T = \{\{a, b, c\}, \{1, \ldots, 9\}, \{1\}, \{6, 9\}, \mu\}$.
Soit μ : $\mu(a, 1) \mapsto 2$; $\mu(b, 1) \mapsto 4$; $\mu(a, 2) \mapsto 5$; $\mu(b, 2) \mapsto 3$;
$\mu(c, 2) \mapsto 2$; $\mu(a, 3) \mapsto 6$; $\mu(b, 3) \mapsto 4$; $\mu(c, 3) \mapsto 9$; $\mu(c, 4) \mapsto 5$;
$\mu(a, 5) \mapsto 5$; $\mu(a, 5) \mapsto 6$; $\mu(c, 7) \mapsto 4$

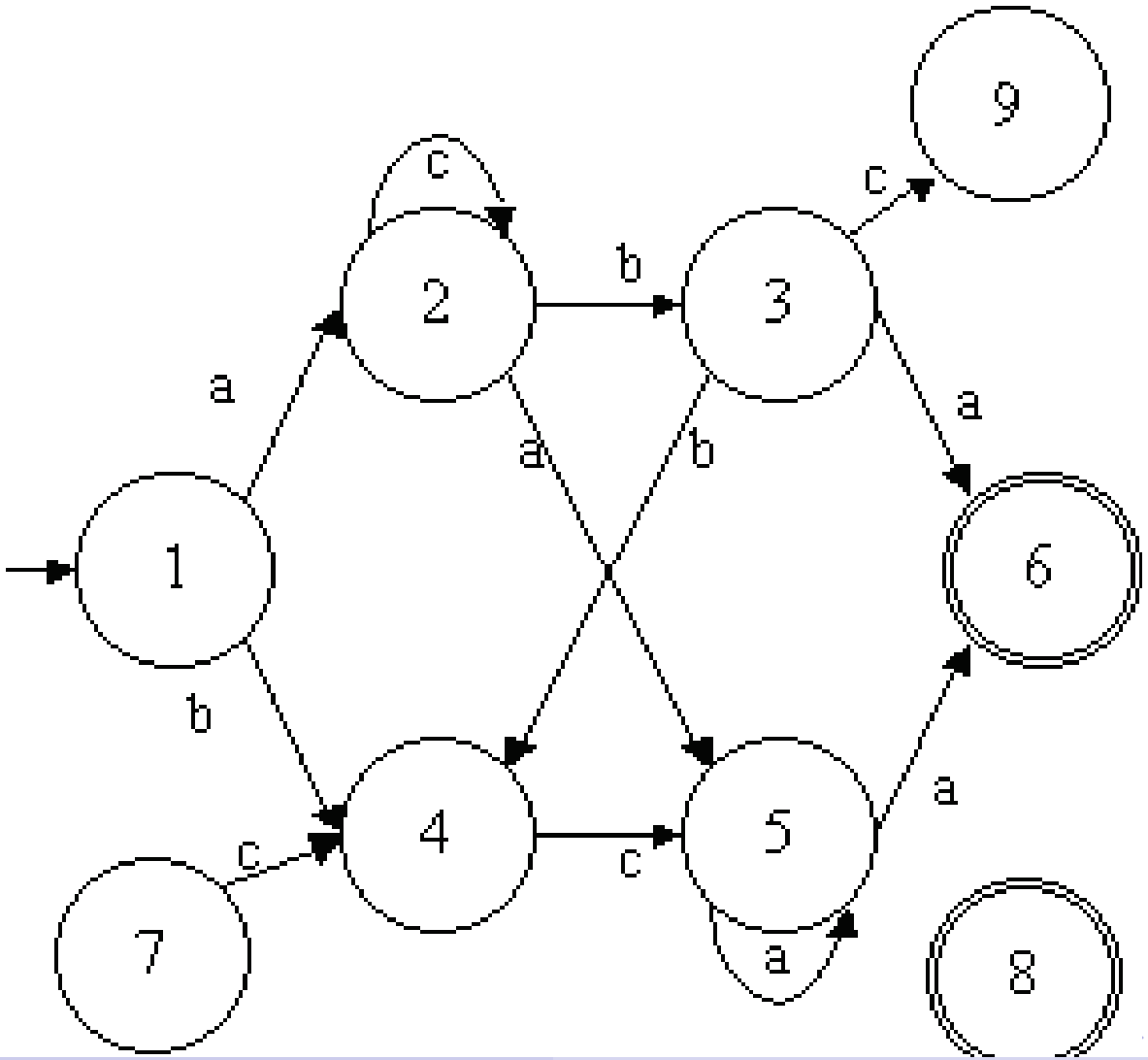
Table des transitions

	Entrée		
État	*a*	*b*	*c*
1	2	4	0
2	5	3	2
3	6	4	9
4	0	0	5
5	$\{5, 6\}$	0	0
6 :	0	0	0
7	0	0	4
8	0	0	0
9 :	0	0	0

Représentation graphique d'un AFN

Légende :





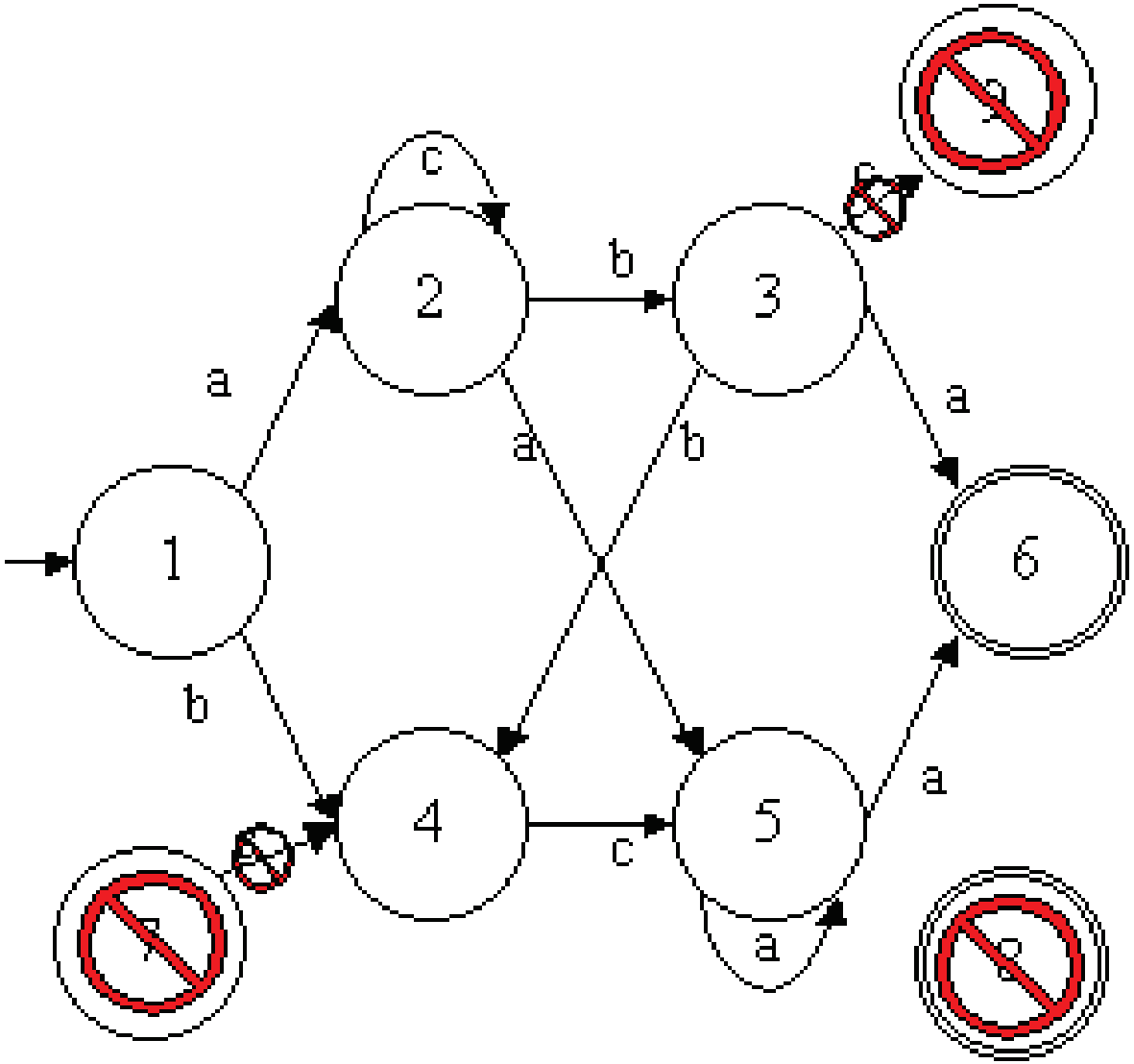
Représentation matricielle d'un AFN

$T = \{\{a, b, c\}, \{1, \ldots, 9\}, \{1\}, \{6, 9\}, \mu\}$. Soit μ :
$\mu(a, 1) \mapsto 2$; $\mu(b, 1) \mapsto 4$; $\mu(a, 2) \mapsto 5$; $\mu(b, 2) \mapsto 3$; $\mu(c, 2) \mapsto 2$;
$\mu(a, 3) \mapsto 6$; $\mu(b, 3) \mapsto 4$; $\mu(c, 3) \mapsto 9$; $\mu(c, 4) \mapsto 5$; $\mu(a, 5) \mapsto 5$;
$\mu(a, 5) \mapsto 6$; $\mu(c, 7) \mapsto 4$
Automate simplifié

	Entrée		
État	*a*	*b*	*c*
1	2	4	0
2	5	3	2
3	6	4	0
4	0	0	5
5	$\{5, 6\}$	0	0
6 :	0	0	0

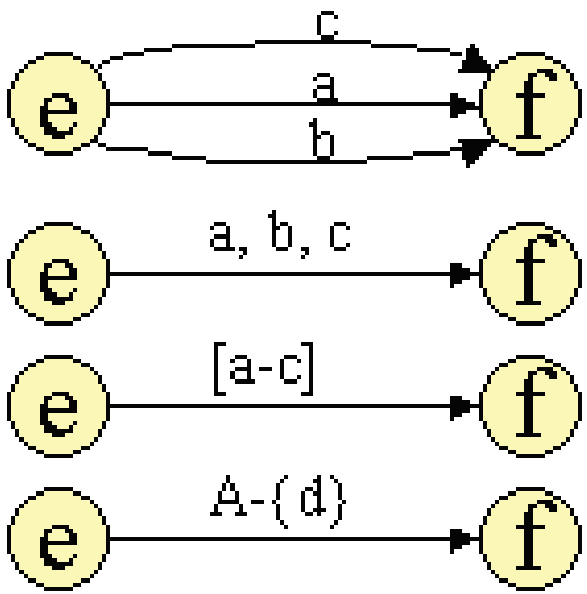
Représentation graphique d'un AFN

Automate simplifié

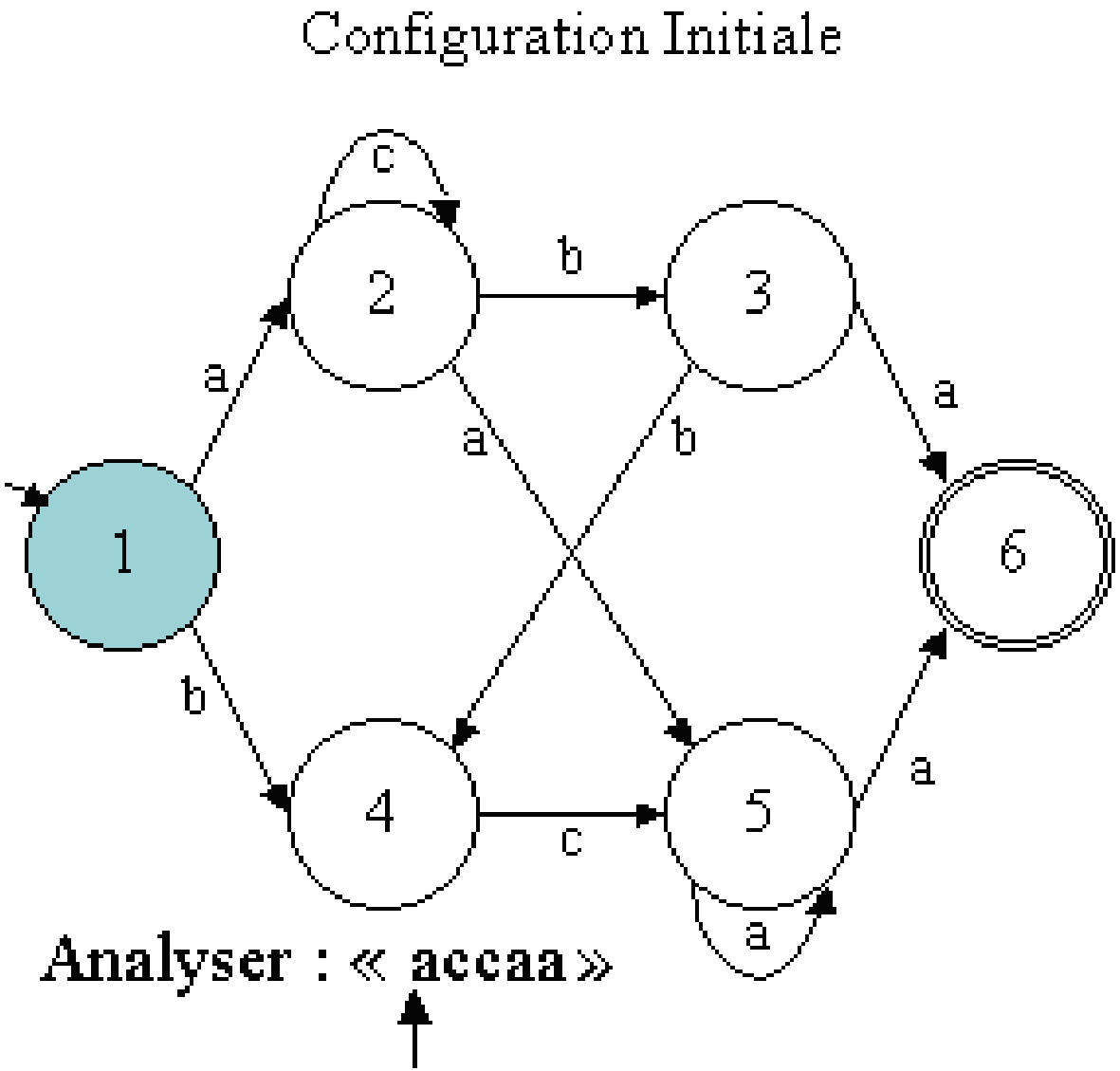


Représentation graphique d'un AFN

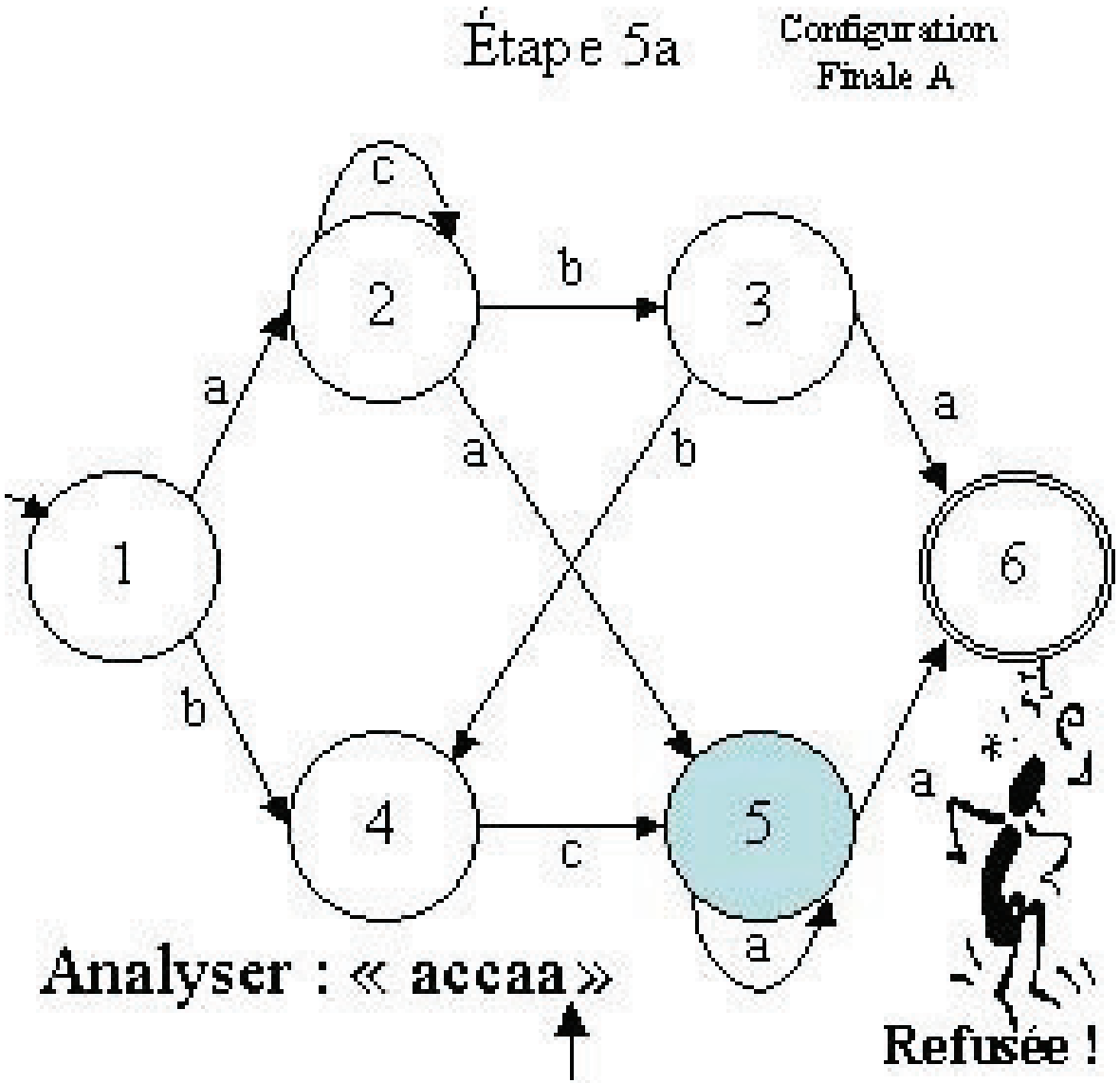
Simplification au niveau du diagramme d'états



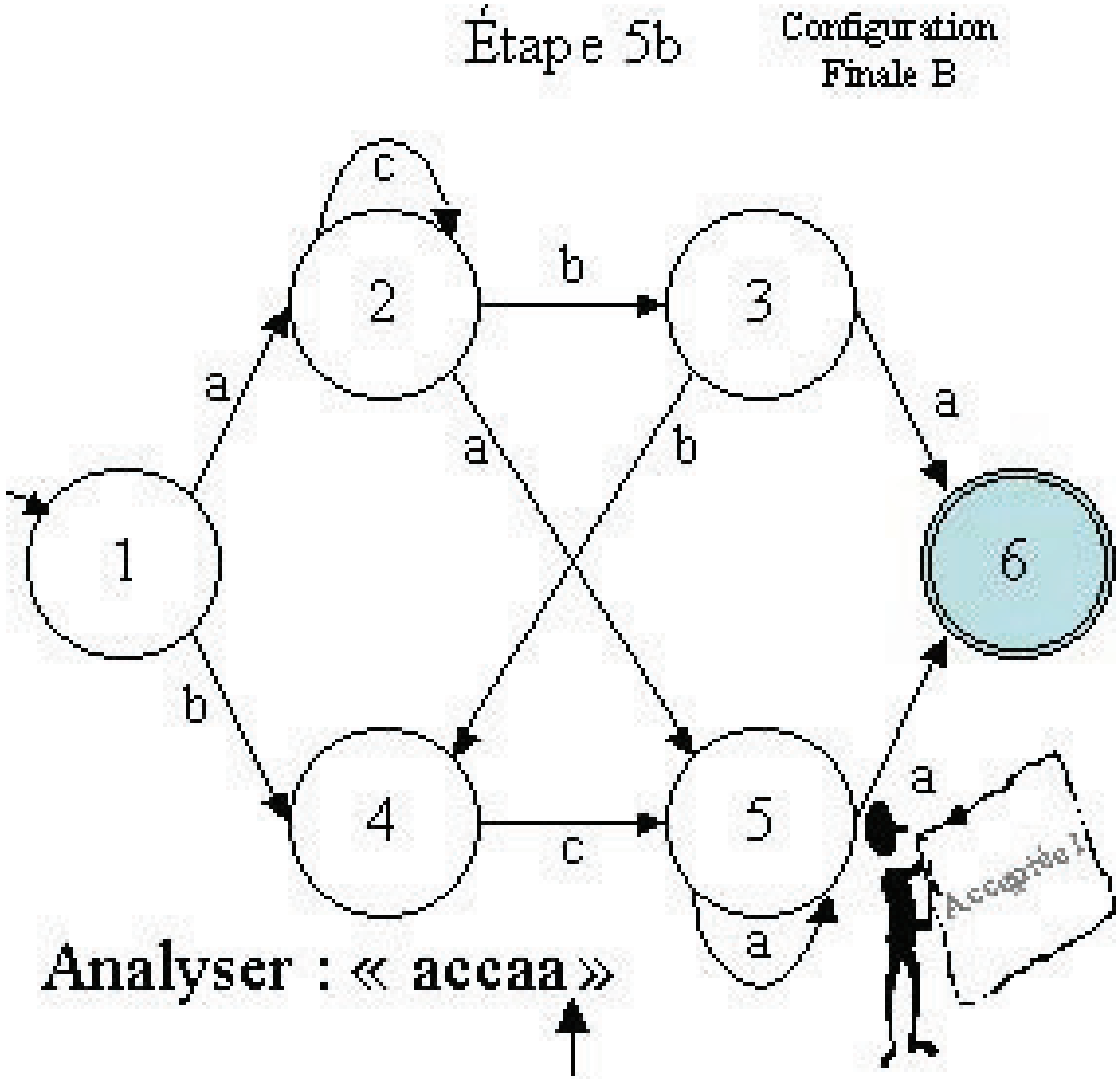
Exécution d'un AFN



Exécution d'un AFN



Exécution d'un AFN



Définition des configurations et actions d'un AFN

Définition 4.5.a - configuration d'un AFN

Si $T = (A, Q, I, F, \mu)$ est un AFN, alors tout couple (a, q) avec $a \in A^*$ et $q \in Q$ est une **configuration** de T.

Définition 4.5.b - configuration d'arrêt d'un AFN

Une **configuration d'arrêt** est une configuration :

- soit de la forme (ε, q) avec $q \in Q$,
- soit de la forme (a, q) avec $q \in Q$ et $a \in A^*$ telles qu'il n'existe pas de transition $(a, q) \rightarrow q' \ \forall q' \in Q$.

Définition 4.5.c - configuration non déterministe/configuration déterministe

Une **configuration non déterministe** est une configuration à partir de laquelle plusieurs transitions sont possibles. Dans le cas contraire, la configuration est dite **configuration déterministe**.

Définition action et suite d'actions

Définition 4.6 - Définition action (ou calcul)

Une **action** (ou un **calcul**, un **déplacement**) de l'automate $T = (A, Q, I, F, \mu)$ est représentée par une relation binaire sur les configurations notée : $\xrightarrow{T}$ ou $\xrightarrow{1/T}$
($\longrightarrow$ s'il n'y a pas d'ambiguïté sur l'automate)

La transition $\mu(a, q_i) \mapsto q_j$ s'écrit alors $(aw, q_i) \xrightarrow{1/T} (w, q_j)$

$C \xrightarrow{0/T} C$ (même configuration)

$C \xrightarrow{k/T} C', k \geq 1$ si et seulement si $\exists C_1, \ldots, C_{k-1}$ tels que

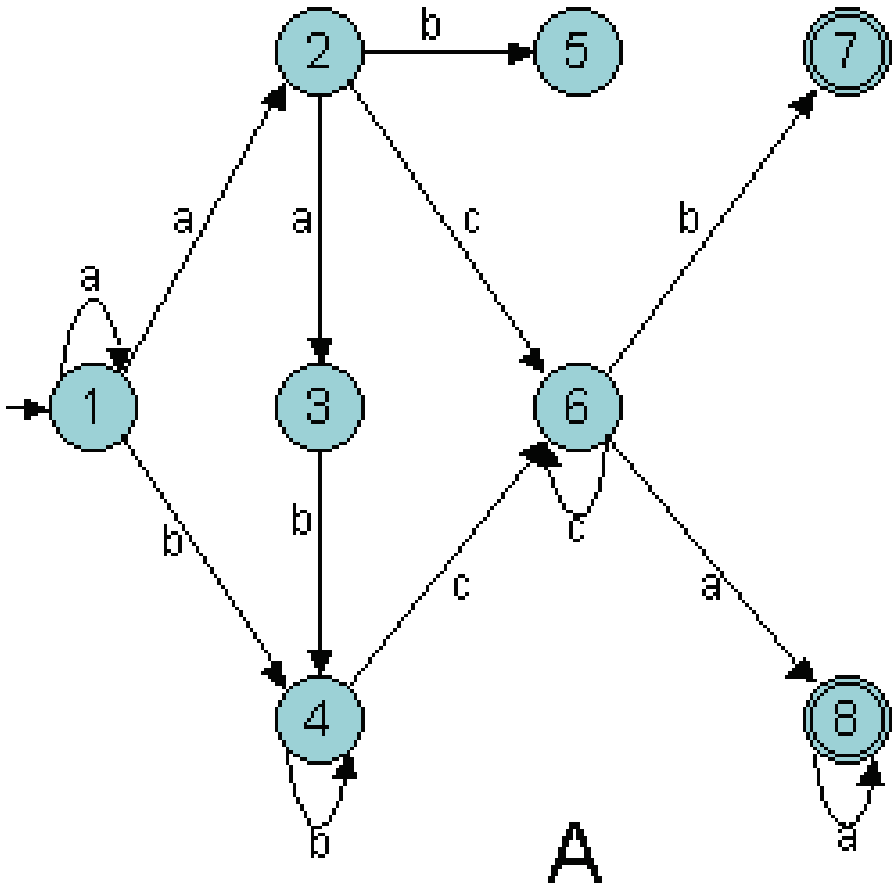
$$C_i \xrightarrow{1/T} C_{i+1}, \forall i, 0 \leq i < k \text{ avec } C_0 = C \text{ et } C_k = C'$$

Définition 4.7 - Définition suite d'actions/chemin d'actions

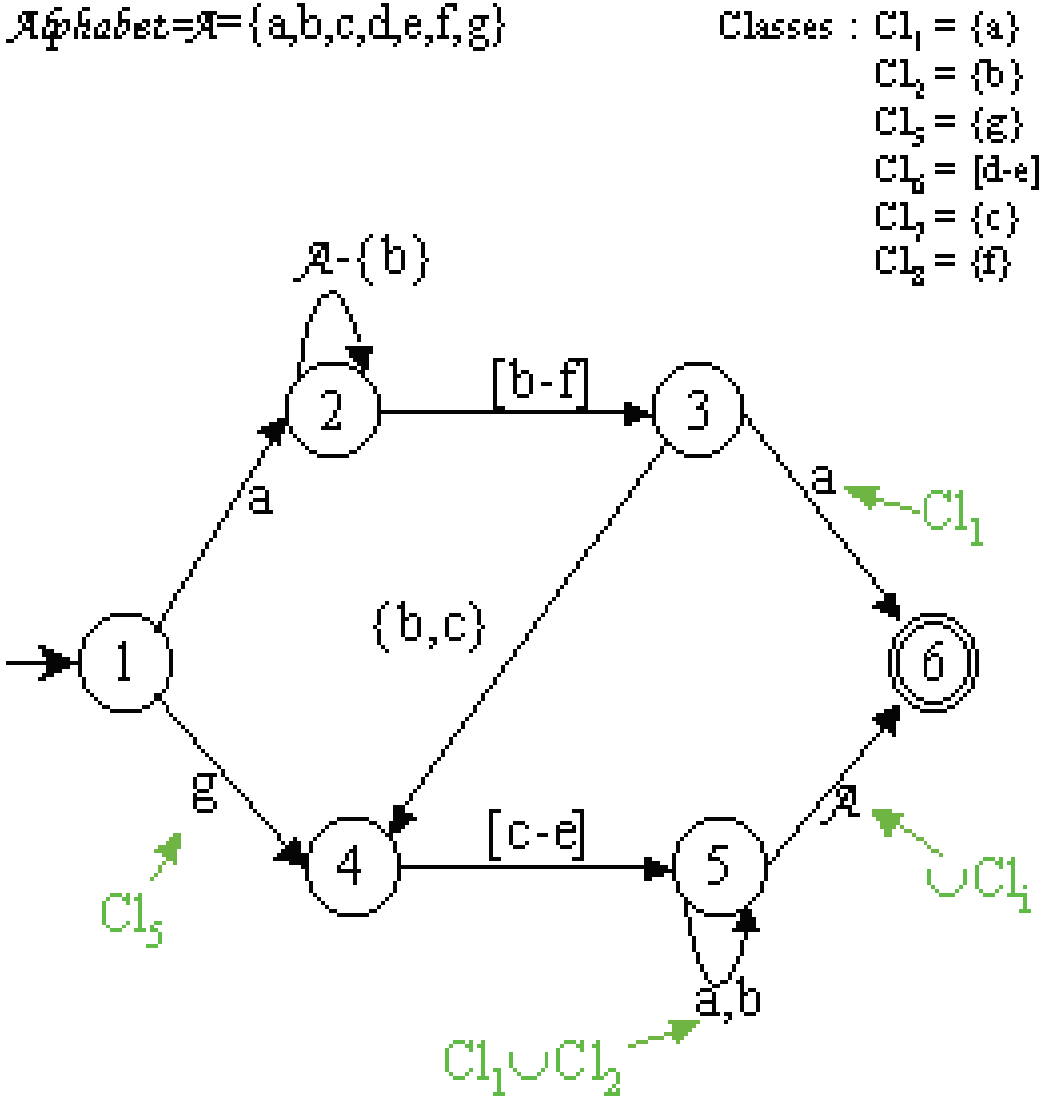
Nous appellerons **suite d'actions** (ou de configurations) ou **chemin d'actions** les différentes actions consécutives permettant de passer d'une configuration à une autre.

Tests définition d'un automate

Donnez toutes les suites d'actions possibles pour chacun des mots suivants avec l'automate fini ci-dessous, l'alphabet $\{a, b, c\}$ et en déduire s'ils sont reconnus : "ε", "a", "aab", "bcb", "abca", "acac" et "aabbcb".



Automate avec classes de symboles



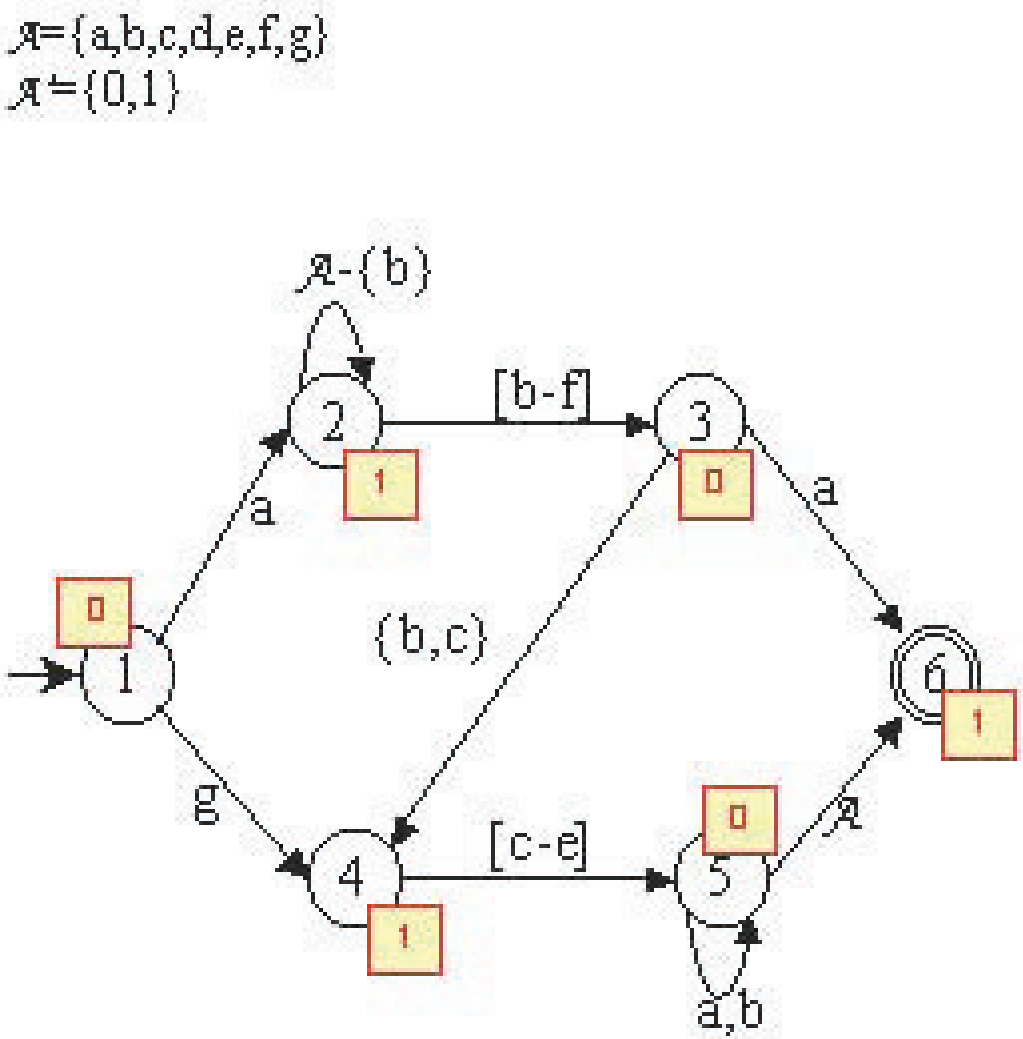
Définition d'une Machine de Moore

Définition 4.8 - Machine de Moore

Une **machine de Moore** est composée de la même manière qu'un automate fini. Elle est défini par $(A, A', Q, I, F, \mu, \mu')$ tel que :

- A un alphabet d'entrée, ensemble fini, non vide de symboles
- A' un alphabet de sortie, ensemble fini, non vide de symboles
- Q l'ensemble des états possibles (fini et non vide) ;
- I $I \in Q$ l'état initial ;
- F $F \subseteq Q$ l'ensemble des états finaux ou états d'acceptation ;
- μ la fonction de transition telle que : $\mu : A \cup \{\varepsilon\} \times Q \rightarrow Q$. Pour la configuration courante, $\mu(a_j, q_i) \mapsto q_k$ signifie qu'on passe dans l'état q_k et que le ruban est déplacé d'une case vers la gauche (seulement si $a_j = \varepsilon$) ;
- μ' une fonction de sortie telle que : $\mu' : Q \rightarrow A'$. $\mu'(q_i) \mapsto a_j$ indique que le caractère a_j est émis en entrant dans l'état q_i.

Exemple de machine de Moore



Définition d'une Machine de Mealy

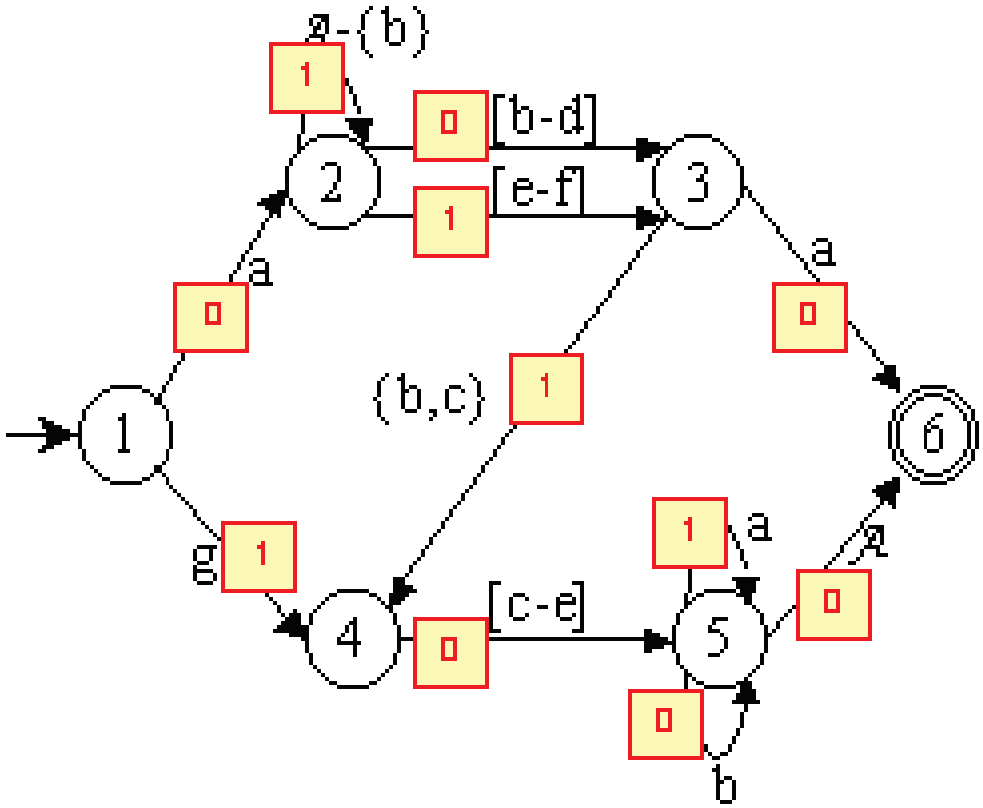
Définition 4.9 - Machine de Mealy

Une **machine de Mealy** est composée de la même manière qu'un automate fini. Elle est défini par (A, A', Q, I, F, μ) tel que :

- A un alphabet d'entrée, ensemble fini, non vide de symboles
- A' un alphabet de sortie, ensemble fini, non vide de symboles
- Q l'ensemble des états possibles pour la machine (fini et non vide) ;
- I $I \in Q$ est l'ensemble des états initiaux ou états de départ ;
- F $F \subseteq Q$ l'ensemble des états finaux ou états d'acceptation ;
- μ la fonction de transition μ telle que : $A \cup \{\varepsilon\} \times Q \rightarrow Q \times A'$. Pour la configuration courante d'état et de symbole, $\mu(a_j, q_i) \mapsto (q_k, a'_1)$ signifie qu'on passe dans l'état q_k, que le symbole a'_l est généré (imprimé) et que le ruban est déplacé d'une case vers la gauche (seulement si $a_j = \varepsilon$).

Exemple de machine de Mealy

$\mathcal{A}=\{a,b,c,d,e,f,g\}$
$\mathcal{A}=\{0,1\}$



Équivalence entre machines de Mealy et de Moore

Définition 4.10 - Équivalence entre Machine de Mealy et Machine de Moore

Soit une machine de Moore, notée Mo, qui imprime x au départ et une machine de Mealy, notée Me. Les deux machines sont équivalentes, Mo $\equiv$ Me, si pour tout mot d'entrée, Me imprime w et Mo imprime xw.

Théorème 4.1 - Théorème Moore $\rightarrow$ Mealy

Pour toute machine de Moore, il existe une machine de Mealy tel que les deux machines soient équivalentes.

Théorème 4.2 - Théorème Mealy $\rightarrow$ Moore

Pour toute machine de Mealy, il existe une machine de Moore tel que les deux machines soient équivalentes.

Exemple de machine de Mealy

Donner la machine de Mealy permettant de calculer le complément binaire. Par exemple, "101" donne "010", "001010" donne "110101"...

Langage entre états - Langage droit et gauche

Définition 4.11 - Langage entre états

Un **langage entre deux états** quelconques q_0 et q_1 de $T = (A, Q, I, F, \mu)$ est défini par :

$$L_T(q_0, q_1) = \{w | w \in A^*, \forall x \in A^*, (wx, q_0) \xrightarrow{*/T} (x, q_1)\}.$$

Définition 4.12 - Langage droit et Langage gauche

Le **langage gauche** d'un état $q \in Q$ d'un AFN $T = (A, Q, I, F, \mu)$ est donné par la fonction $LG_T : Q \to A^*$ où

$$LG_T(q) = \cup_{i \in I} L_T(i, q) = \{w | w \in A^*, \forall x \in A^*, \exists i \in I, (wx, i) \xrightarrow{*/T} (x, q)\}$$

Le **langage droit** d'un état $q \in Q$ d'un AFN $T = (A, Q, I, F, \mu)$ est donné par la fonction $LD_T : Q \to A^*$ où

$$LD_T(q) = \cup_{f \in F} L_T(q, f) = \{w | w \in A^*, \exists f \in F, (w, q) \xrightarrow{*/T} (\varepsilon, f)\}$$

Langage d'un AFN

Définition 4.13 - Langage d'un AFN / Langage reconnaissable / Rec(A^*)

Un langage **reconnu** (**accepté**, **défini**) par un AFN $T = (A, Q, I, F, \mu)$ est donné par la fonction L : $AFN \rightarrow A^*$ où $L(T) = \cup_{f \in F} LG_T(f) = \cup_{i \in I} LD_T(i) = \cup_{i \in I, f \in F} l_T(i, f)$. $L(T)$ est donc l'ensemble des chaînes acceptées par T, c'est-à-dire :

$$L(T) = \{w | w \in A^*, \exists f \in F, i \in I : (w, i) \overset{*/T}{\rightarrow} (\varepsilon, f)\}$$

Un langage L sur un alphabet A est dit **langage reconnaissable**, noté $L \in Rec(A^*)$, s'il existe un automate fini T tel que $L = L(T)$.

Définition 4.14 - Automates équivalents

Deux automates sont équivalents s'ils acceptent le même langage, c'est-à-dire : $T1 \equiv T2$ si et seulement si $L(T1) = L(T2)$.

Exemple de langage reconnaissable

Montrer que le langage des entiers signés est reconnaissable.

Définition - État stérile/État co-accessible

Définition 4.15 - État stérile

Un **état stérile** ou **état mort** e est un état pour lequel il n'est pas possible d'atteindre un état final, c'est-à-dire si :
$LD(e) = \emptyset$. En théorie des graphes, ils sont aussi appelés *nœuds puits*.
Un état qui n'est pas stérile (ie $LD(e) \neq \emptyset$) est dit **état co-accessible**.

État accessible / état inaccessible

Définition 4.16 - État accessible

Un état e est dit **accessible depuis un état** f pour un AFN $T = (A, Q, I, F, \mu)$ si $\exists wx \in A^*$ telle que $(wx, f) \xrightarrow{*/T} (x, e)$.
Un état e est dit **accessible** pour un AFN $T = (A, Q, I, F, \mu)$ si et seulement si :
$\exists wx \in A^*$ et $i \in I$ tels que $(wx, i) \xrightarrow{*/T} (x, e)$ (il existe au moins un chemin de i à e, c'est-à-dire $LG(e) \neq \emptyset$).
Dans le cas contraire, l'état e est dit **inaccessible** (il n'existe pas de chemin de $i \in I$ à e, c'est-à-dire $LG(e) = \emptyset$).

État isolé / état co-accessible

Définition 4.17 - État isolé

Un état e est dit **isolé** s'il ne participe à aucune transition.

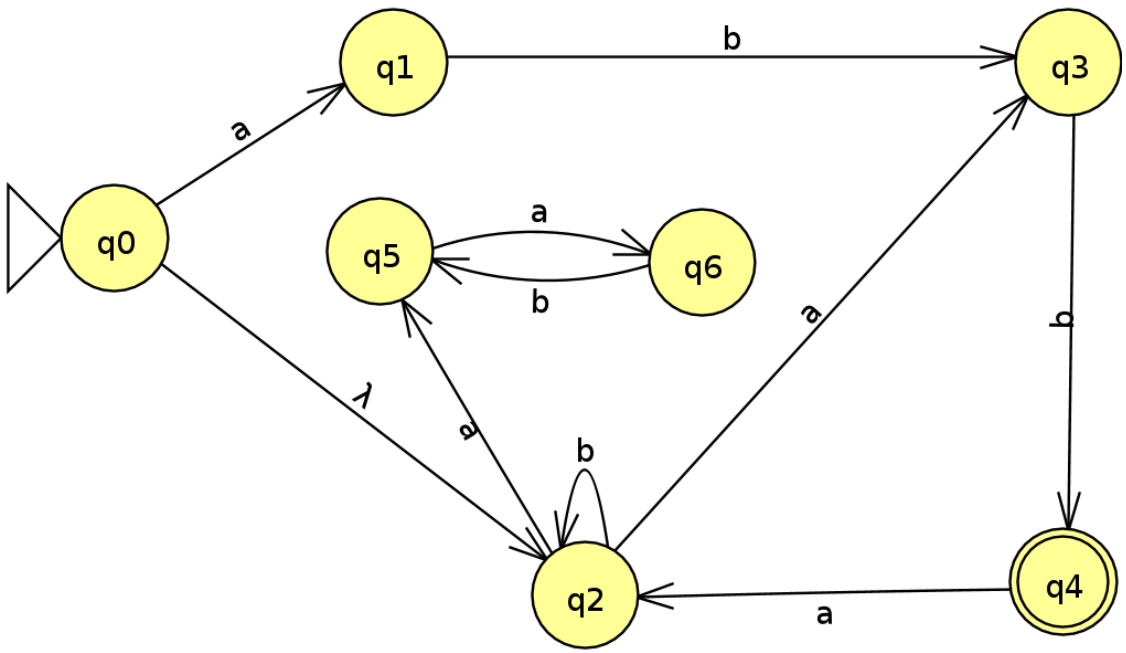
Définition 4.18 - État utile

Soit un AFN $T = (A, Q, I, F, \mu)$ et un état $e \in Q$:

- e est **utile**$_I$ s'il est accessible ;
- e est **utile**$_F$ s'il est co-accessible ;
- e est **utile** si :
  $utile_I(e) \wedge utile_F(e) \equiv Accessible(e) \wedge co - accessible(e)$. Dans le cas contraire, l'état est dit **inutile**

Propriétés des états

Donner les propriétés des états pour l'automate suivant défini sur l'alphabet $\{a, b\}$ ($\lambda = \epsilon$).



Dimension d'un automate

Définition 4.19 - Dimension d'un AFN/Automate minimal

La **dimension** d'un AFN $T = (A, Q, I, F, \mu)$, notée $|T|$ est définie par : $|T| = |Q|$. Autrement dit, la dimension d'un automate est le nombre d'états de cet automate.

Un **automate de dimension minimale** est un automate T tel que $\forall T', T' \equiv T$ alors $|T'| \geq |T|$.

Propriétés liées aux caractéristiques des états

Définition 4.20 - AFN complet

Un **AFN complet** $T = (A, Q, I, F, \mu)$ est un AFN vérifiant :
$Complet(T) \equiv (\mu(a, q) \neq 0, \forall q \in Q, \forall a \in A)$ avec 0 l'état d'erreur.

Définition 4.21- AFN utile/émondé

Un **AFN utile** $T = (A, Q, I, F, \mu)$ est un AFN vérifiant :

- $Utile_I$ est défini par : $Utile_I(T) \equiv utile_I(q), \forall q \in Q$;
- $Utile_F$ est défini par : $Utile_F(T) \equiv utile_F(q), \forall q \in Q$;

Un **AFN émondé** ou **utile** est un AFN dont tous les états sont utiles.
Donc : $Emondé(T) \equiv Utile_I(T) \wedge Utile_F(T) \equiv utile(q), \forall q \in Q$.

Propriétés liées aux caractéristiques des états

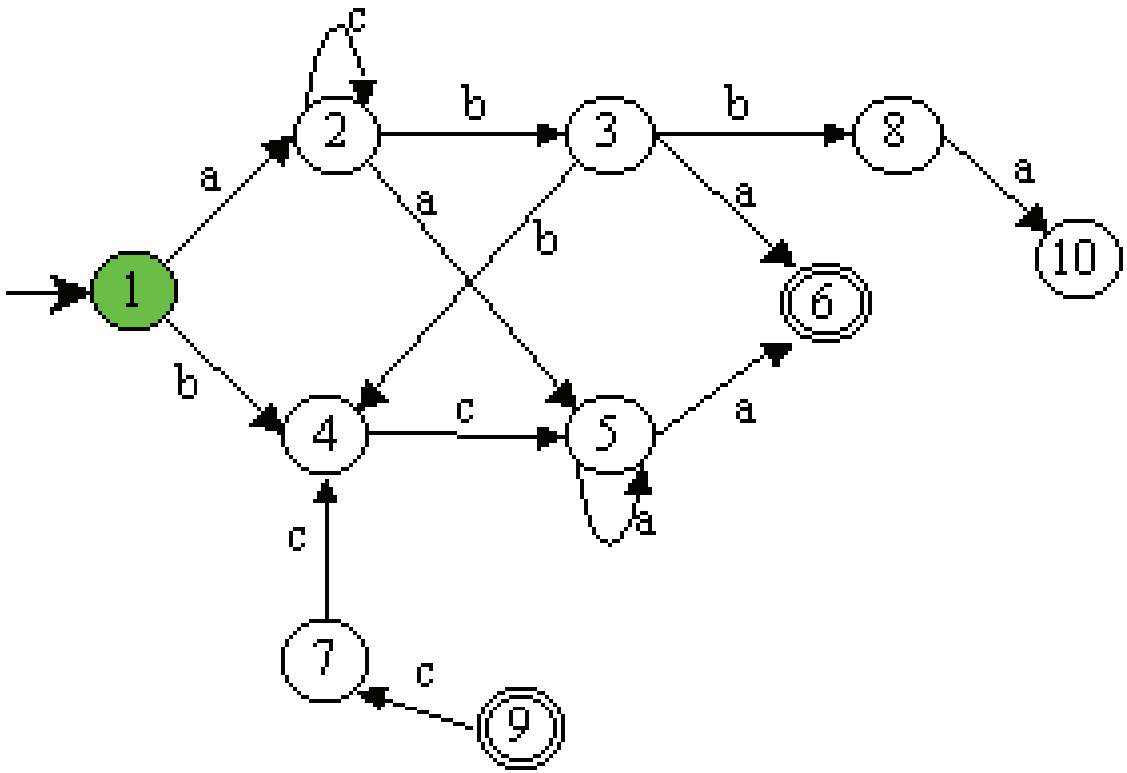
Théorème 4.3 - Théorème de l'émondage

Pour tout langage reconnaissable L sur un alphabet A (ie $L \in Rec(A^*)$) il existe un automate émondé qui le reconnaît.

Algorithme de l'émondage

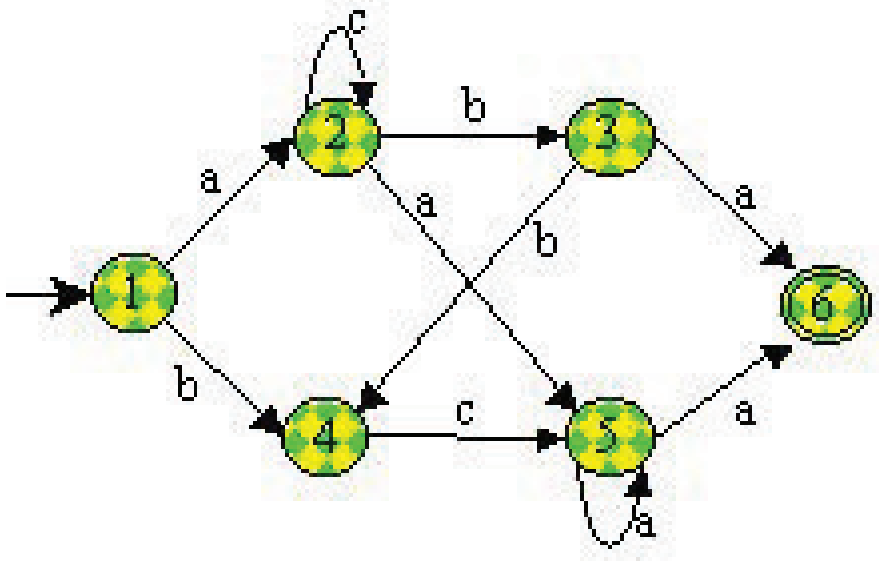
1. Détermination de l'ensemble w des états accessibles
   1. Soit w_i l'ensemble des états accessibles à l'instant i.
      Au départ, les seuls états accessibles sont les états initiaux, donc $w_0 = I$
   2. Tout état q est accessible s'il existe une transition $\mu(a, p) = q$ avec p lui-même accessible. Donc, $w_{i+1} = w_i \cup \{q | q \in Q, \exists p \in w_i$ et $\exists a \in A$ tels que $\mu(a, p) = q\}$
   3. C'est terminé quand $w_{i+1} = w_i$ donc $w = w_i$
2. Détermination de l'ensemble z des états co-accessibles
   1. Soit z_i l'ensemble des états co-accessibles à l'instant i. Au départ, seuls les états finaux sont co-accessibles, donc $z_0 = F$
   2. Tout état q est co-accessible s'il existe une transition $\mu(a, q) = p$ avec p lui-même co-accessible. Donc, $z_{i+1} = z_i \cup \{q | q \in Q, \exists p \in z_i$ et $\exists a \in A$ tels que $\mu(a, q) = p\}$
   3. C'est terminé quand $z_{i+1} = z_i$ donc $z = z_i$
3. Construction de l'automate émondé T' :
   $T' = (A, Q' = w \cap z, I' = I \cap w \cap z, F' = F \cap w \cap z, \mu')$ avec
   $\mu' = \{(q, a, q') | (q, a, q') \in \mu, q, q' \in Q'\}$

Exemple d'émondage d'un automate fini



Étape W_0

Exemple d'émondage d'un automate fini



Émondage avec $W \cap Z$

Propriétés sur la structure de l'automate

Définition 4.22 - Automate unitaire/standard/normalisé

Un AFN $T = (A, Q, I, F, \mu)$ est dit **unitaire** s'il ne possède qu'un seul état initial, c'est-à-dire que $I = \{i\}$ avec $i \in Q$.
Un AFN $T = (A, Q, I, F, \mu)$ est dit **standard** s'il est unitaire ($I = \{i\}$) et si $\{\mu(x, q) = i | x \in A, q \in Q\} = \emptyset$.
Autrement dit, aucune transition n'arrive sur le seul état initial.
Un AFN $T = (A, Q, I, F, \mu)$ est dit **normalisé** s'il est standard et si $F = \{f\}$ et $\{\mu(x, f) = q | x \in A, q \in Q\} = \emptyset$. Autrement dit, s'il est standard, ne possède qu'un seul état final et qu'aucune transition n'a pour origine cet état final.

Théorème de l'automate standard

Théorème 4.4 - Théorème l'automate standard

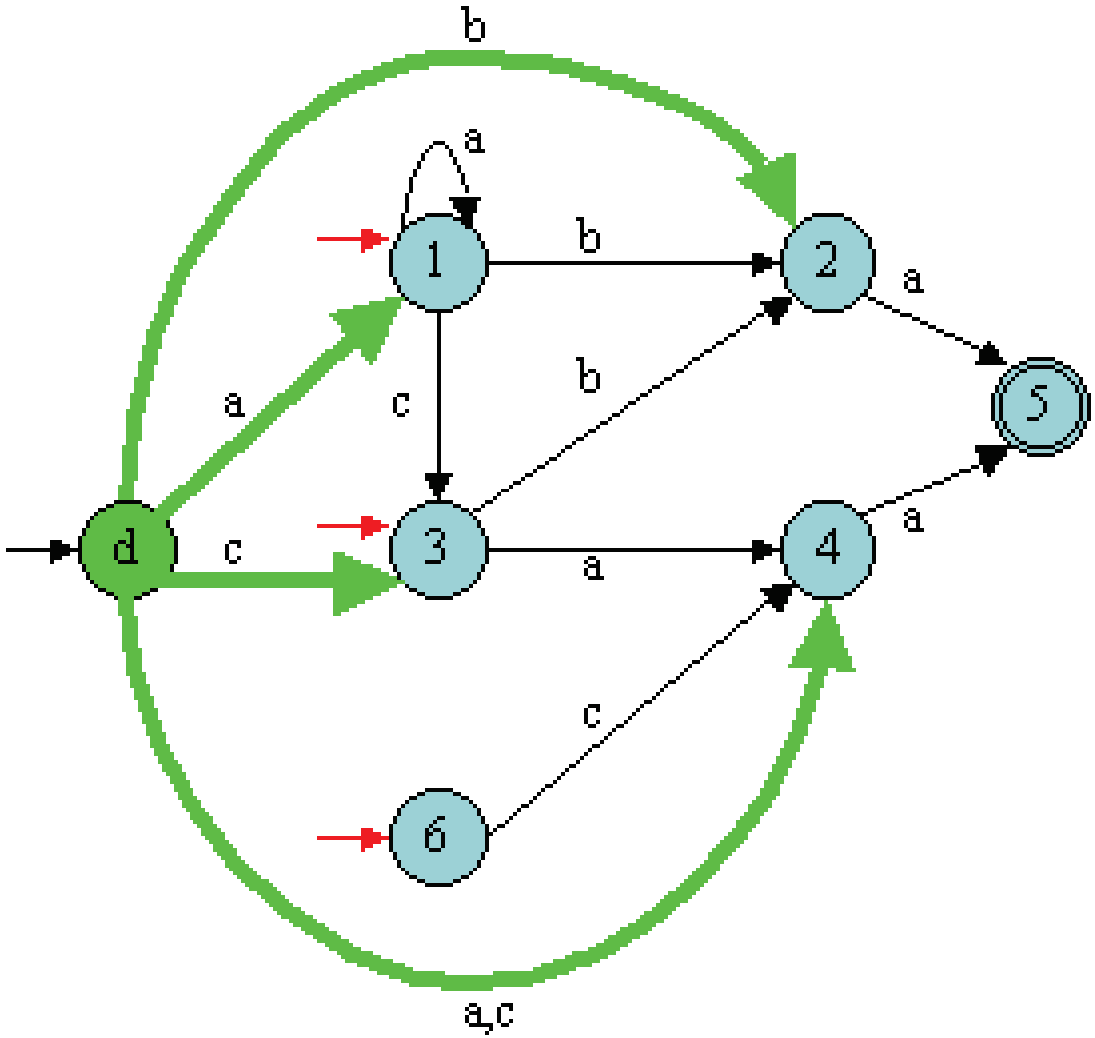
Pour tout langage reconnaissable L sur un alphabet A (ie $L \in Rec(A^*)$) il existe un automate standard qui le reconnaît.

Algorithme de la standardisation

Soit T' l'automate standard équivalent à T, $T = (A, Q \cup \{d\}, \{d\}, F', \mu')$

- $d \notin Q$
- $F' = F \cup \{d\}$ si $I \cap F \neq \emptyset$
  $F' = F$ sinon
- $\mu' = \mu \cup \{(d, a, q) | (i, a, q) \in \mu, i \in I\}$

Exemple de standardisation d'un automate fini



Théorème de l'automate normalisé

Théorème 4.5 - Théorème l'automate normalisé

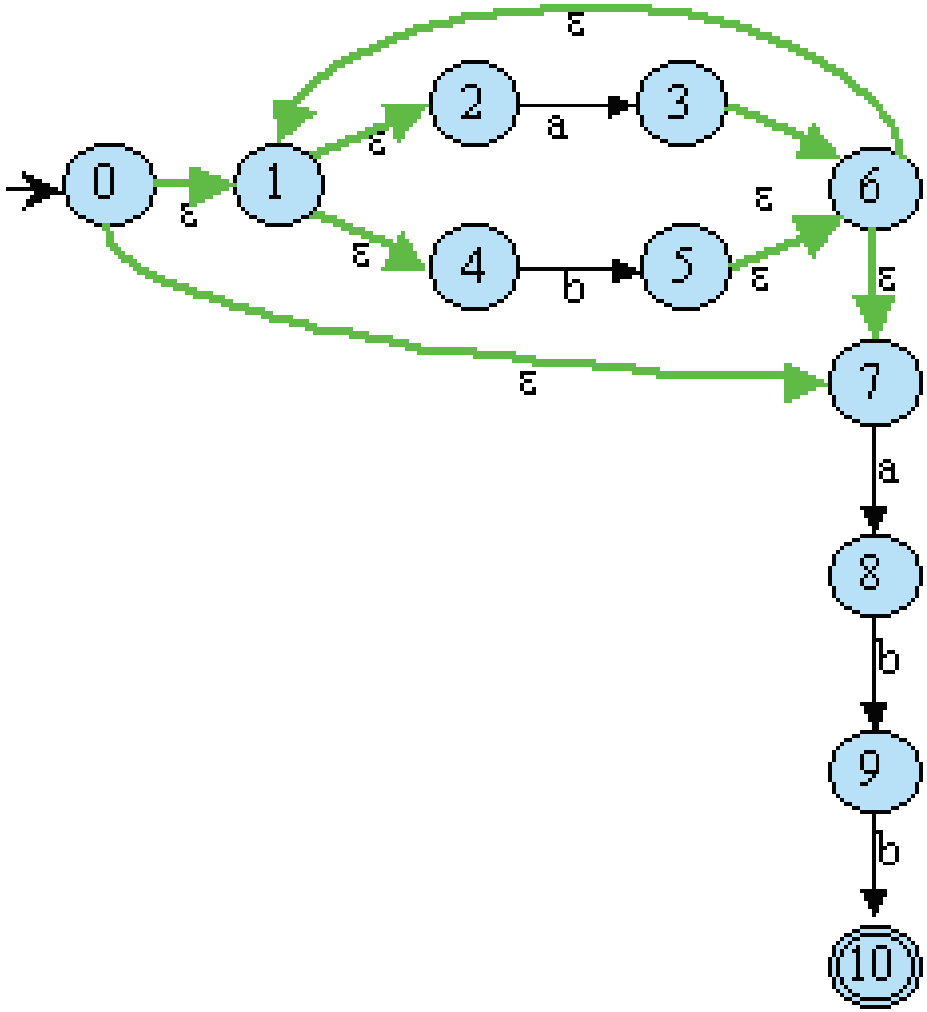
Pour tout langage reconnaissable L sur un alphabet A ne contenant pas le mot vide (ie $L \in Rec(A^+)$) il existe un automate normalisé qui le reconnaît.

Cas des automates avec des ε-transitions

Définition 4.23 - ε-transition

Une ε-transition est une transition qui n'utilise aucune entrée. C'est une transition *spontanée*, c'est-à-dire que l'automate *décide* simplement de changer d'état sans lire de symbole.

Exemple d'automate avec ε-transitions



Cas des automates avec des ε-transitions

Définition 4.24 - AFN ε-libre

Un AFN $T = (A, Q, I, F, \mu)$ est dit ε-libre s'il ne possède pas de ε-transition, c'est-à-dire si : $\mu(\varepsilon, q) = 0$, $\forall q \in Q$.

Automates déterministes (AFD)

Définition 4.25 - Automate fini déterministe (AFD)

Un AFN $T = (A, Q, I, F, \mu)$ est dit **déterministe** (AFD) si :

- il est ε-libre
- il est unitaire
- Si $\forall q \in Q$, $\forall a \in A$: $|\mu(a, q)| \leq 1$.
  Autrement dit, si $\mu(a, q) = q_1$ et $\mu(a, q) = q_2$ alors $q_1 = q_2$.

Automate minimal

Définition 4.26 - États distinguables/États k-indistinguables

Soit un AFD $T = (A, Q, I, F, \mu)$, on dit que la chaîne d'entrée $w \in A^*$ distingue l'état e de l'état f $(e \neq f)$ dans T si :

- partant de l'état e et le faisant fonctionner avec w, on termine dans un état d'acceptation ;
- partant de l'état f et le faisant fonctionner avec w, on termine dans un état de non-acceptation (ou vice versa).

Les états e et f sont alors dits **distinguables**. Autrement dit, w distingue e et f si $\exists q_1, q_2 \in Q$, $(w, e) \overset{*/T}{\rightarrow} (\varepsilon, q_1)$ et $(w, f) \overset{*/T}{\rightarrow} (\varepsilon, q_2)$ avec exactement un des états q_1 ou q_2 dans F mais pas l'autre.

Deux états e et f d'un AFD T sont **k-indistinguables**, noté $e \overset{k}{\equiv} f$, si $\forall w \in A^*$, $|w| \leq k$, w ne distingue pas e et f.

Deux états e et f d'un AFD T sont **indistinguables**, si $\forall k \geq 0$, $e \overset{k}{\equiv} f$

Automates réduit/automate minimal

Définition 4.27 - Automate fini déterministe réduit

Un AFD $T = (A, Q, I, F, \mu)$ est dit **réduit** si tous les états de Q sont accessibles et si tous les d'états de Q pris deux à deux sont distinguables.

Définition 4.28 - Automate minimal

Un AFD $T = (A, Q, I, F, \mu)$ est dit **minimal** s'il est réduit et de dimension minimale : $\forall T' \in AFD,\ L(T) = L(T') \Longrightarrow |T| \leq |T'|$.
Un AFD $T = (A, Q, I, F, \mu)$ est dit **minimal complet** s'il est réduit et complet et de dimension minimale dans l'ensemble des AFD complets : $\forall T' \in AFD, Complet(T'), L(T) = L(T') \Longrightarrow |T| \leq |T'|$.

Théorème des facteurs

Théorème 4.6 - Théorème des facteurs

Pour tout langage reconnaissable L sur un alphabet A (ie $L \in Rec(A^*)$),
- $Pref(L)$ (le langage des préfixes des mots L),
- $Suff(L)$ (le langage des suffixes des mots de L) et
- $Fact(L)$ (le langage des facteurs dans L)

sont aussi reconnaissables.

Démonstration : en partant d'un automate émondé, le langage des préfixes est obtenu en considérant tous les états comme des états finaux, le langage des suffixes en considérant que tous les états sont initiaux, le langages des facteurs en considérant que tous les états sont initiaux et finaux.

Théorème de l'ε-fermeture

Théorème 4.7 - l'ε-fermeture

A tout automate $T = (A, Q, I, F, \mu)$ comportant des ε-transitions il existe un automate $T' = (A, Q, I', F', \mu')$ équivalent, sans ε-transition (ε-libre), avec

- $I' = \varepsilon\text{-}fermeture_T(I)$
- $F' = \{q \mid \exists f \in F, f \in \varepsilon\text{-}fermeture_T(q)\}$
- $\mu' = \varepsilon\text{-}fermeture_\mu$ privé des ε-transitions

ε-fermeture

Définition 4.29 - ε-fermeture d'un état

L'**ε-fermeture** d'un état $e \in Q$ d'un AFN $T = (A, Q, I, F, \mu)$, notée **ε-fermeture$_T$(e)**, est l'ensemble des états de T accessibles depuis l'état e par 0, 1 ou plusieurs ε-transitions.

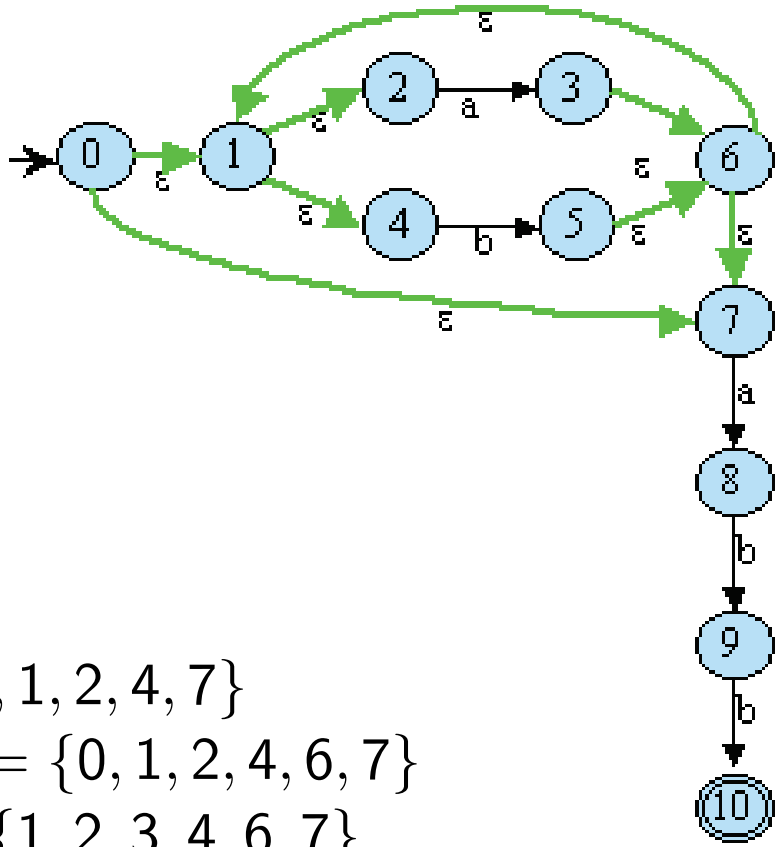
Définition 4.30 - ε-fermeture d'un ensemble d'états

L'**ε-fermeture** d'un ensemble d'états $E \subset Q$ d'un AFN $T = (A, Q, I, F, \mu)$, notée **ε-fermeture$_T$(E)**, est l'ensemble des états de T accessibles depuis chacun des états $e \in E$ par des ε-transitions.

Définition 4.31 - ε-fermeture d'une fonction de transition

L'**ε-fermeture** de la fonction de transition μ d'un AFN $T = (A, Q, I, F, \mu)$, noté **ε-fermeture$_\mu$**, est la fonction $(a, q) \mapsto \varepsilon\text{-}fermeture_\mu(\mu(a, q))$

Exemple de calcul de ε-fermeture



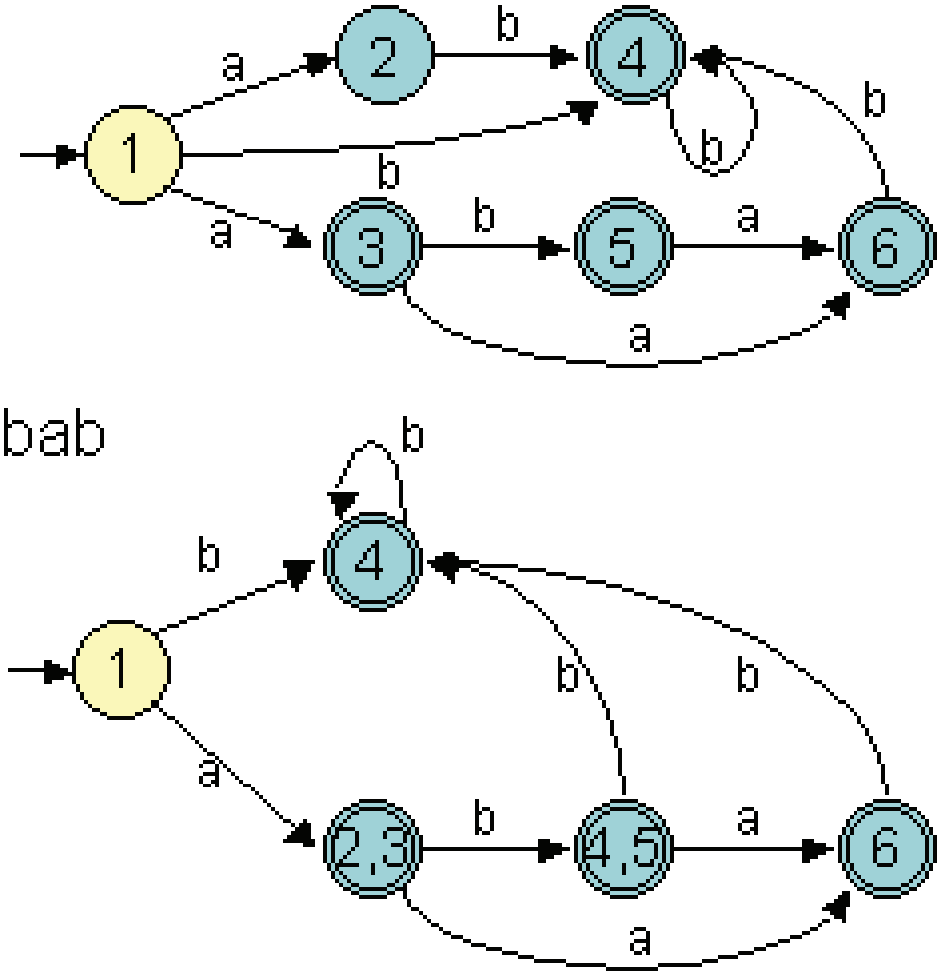
ε-*fermeture*$(0) = \{0, 1, 2, 4, 7\}$
ε-*fermeture*$(\{0, 3\}) = \{0, 1, 2, 4, 6, 7\}$
ε-*fermeture*$(a, 2) = \{1, 2, 3, 4, 6, 7\}$

Théorème d'équivalence entre AFN et AFD

Théorème 4.8 - Théorème d'équivalence entre AFN et AFD

Soit $T = (A, Q, I, F, \mu)$ un AFN qui reconnaît le langage $L = L(T)$ alors il existe un AFD $T = (A, Q', I', F', \mu')$ tel que $L' = L(T') = L$ (l'automate est équivalent).

Exemple d'AFN et d'AFD équivalents



M_T

Définition 4.31 - M_T

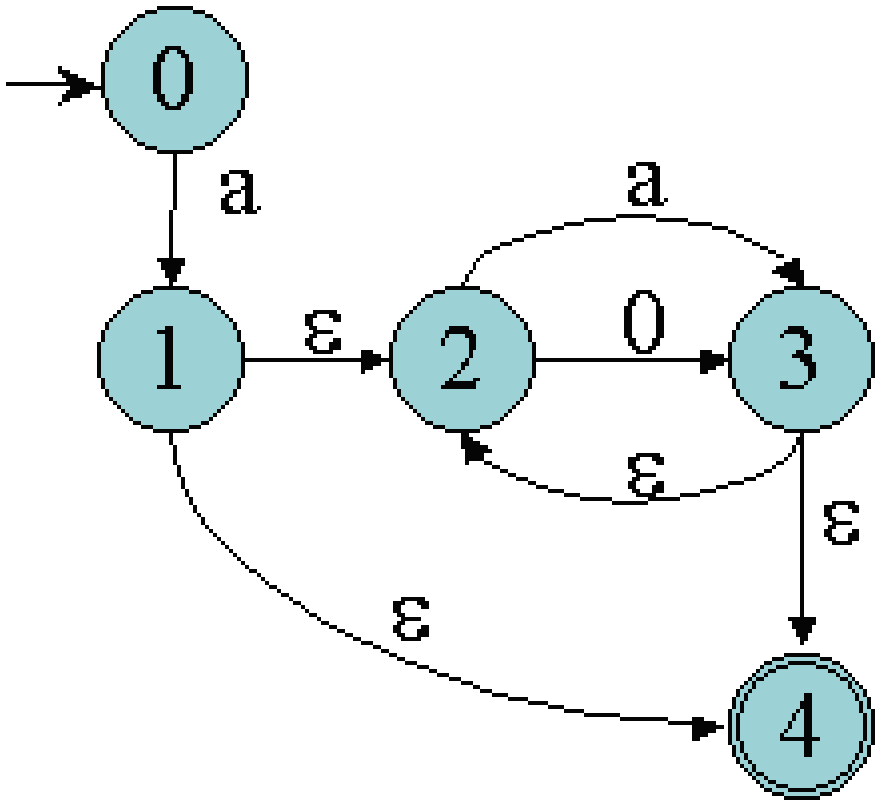
$M_T(a, E)$ est l'ensemble des états E' de T vers lesquels il existe une transition sur le symbole d'entrée a à partir des états $e \in E$.
Donc : $M_T(a, E) = \{q \mid \exists p \in E, \mu(a, p) = q\}$

Algorithme de construction d'un AFD à partir d'un AFN

Construction d'un AFD $T' = (A', Q', I', F', \mu')$ à partir d'un AFN quelconque $T = (A, Q, I, F, \mu)$

- $A' = A$;
- Q' est un ensemble d'ensembles d'états de T ($\emptyset$ au départ)
- $I' = \varepsilon$-fermeture(T, I) ; Ajouter I' dans Q' ;
- Pour chaque "état" E (ensemble d'états de T) non-marqué dans Q' :
  - Le marquer
  - Déterminer pour chaque symbole a de A
    - $E' = M(a, E)$
    - $E'' = \varepsilon$-fermeture(T, E')
    - Si E'' n'existe pas dans Q' alors :
      L'ajouter dans Q'
      S'il contient un état final de T alors l'ajouter dans F'
    - Créer une transition $\mu'(a, E)$

Exemple de construction d'un AFD à partir d'un AFN (1)



Exemple de construction d'un AFD à partir d'un AFN (2)

Phase 1

μ	a	0
$A = \{0\} = I'$		

Exemple de construction d'un AFD à partir d'un AFN (3)

Phase 2

μ	a	0
$A = \{0\} = I'$ $B = \{1, 2, 4\} \in F$ $C = \emptyset$	$\{1, 2, 4\}$	$\emptyset$

Exemple de construction d'un AFD à partir d'un AFN (4)

Phase 3

μ	a	0
$A = \{0\} = I'$	$\{1,2,4\}$	$\emptyset$
$B = \{1,2,4\} \in F$	$\{3,2,4\}$	$\{3,2,4\}$
$C = \emptyset$		
$D = \{2,3,4\} \in F$		

Exemple de construction d'un AFD à partir d'un AFN (5)

Phase 4

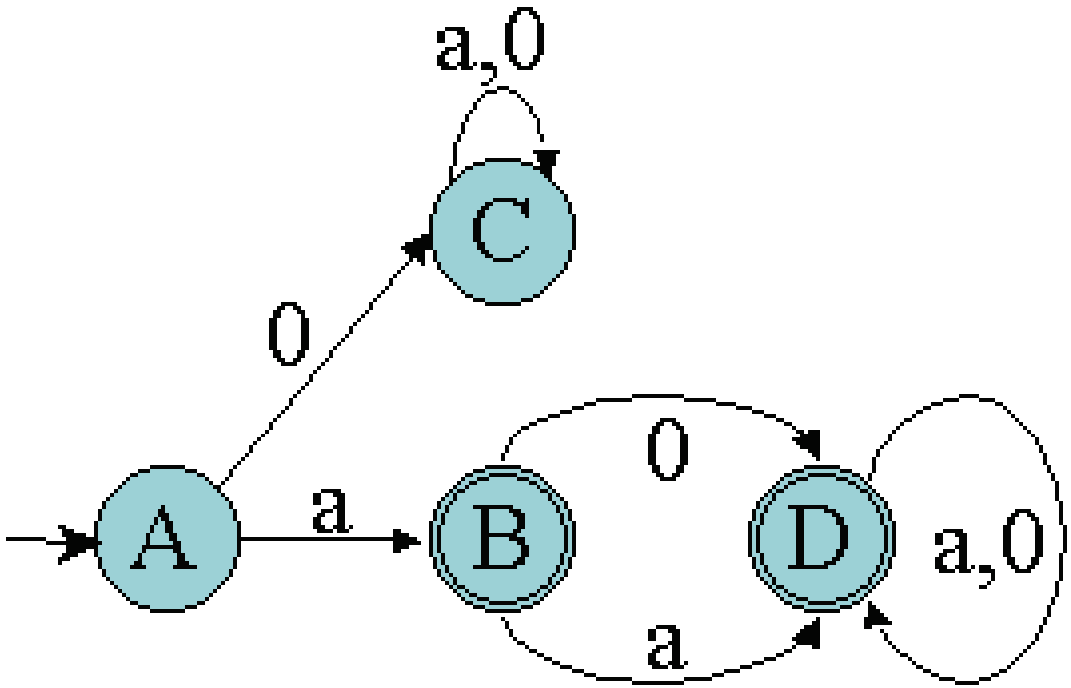
μ	a	0
$A = \{0\} = I'$	$\{1, 2, 4\}$	$\emptyset$
$B = \{1, 2, 4\} \in F$	$\{3, 2, 4\}$	$\{3, 2, 4\}$
$C = \emptyset$	$\emptyset$	$\emptyset$
$D = \{2, 3, 4\} \in F$		

Exemple de construction d'un AFD à partir d'un AFN (6)

Phase 5

μ	a	0
$A = \{0\} = I'$	$\{1, 2, 4\}$	$\emptyset$
$B = \{1, 2, 4\} \in F$	$\{3, 2, 4\}$	$\{3, 2, 4\}$
$C = \emptyset$	$\emptyset$	$\emptyset$
$D = \{2, 3, 4\} \in F$	$\{3, 2, 4\}$	$\{3, 2, 4\}$

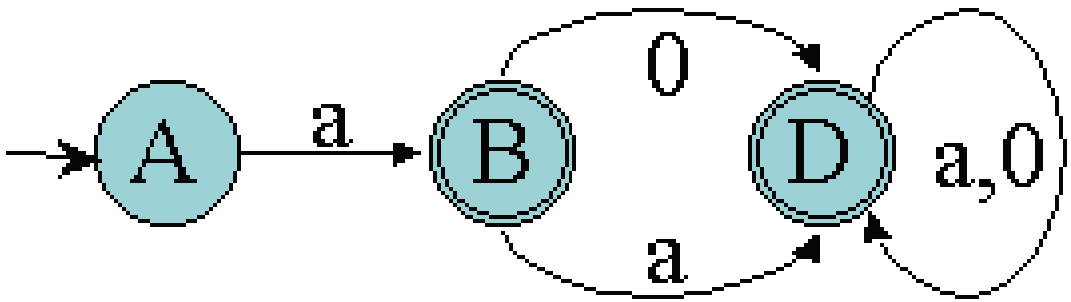
Exemple de construction d'un AFD à partir d'un AFN (7)



Exemple de construction d'un AFD à partir d'un AFN (9)

μ	a	0
$A = \{0\} = I'$	$\{1, 2, 4\}$	$\emptyset$
$B = \{1, 2, 4\} \in F$	$\{3, 2, 4\}$	$\{3, 2, 4\}$
$D = \{2, 3, 4\} \in F$	$\{3, 2, 4\}$	$\{3, 2, 4\}$

Exemple de construction d'un AFD à partir d'un AFN (10)

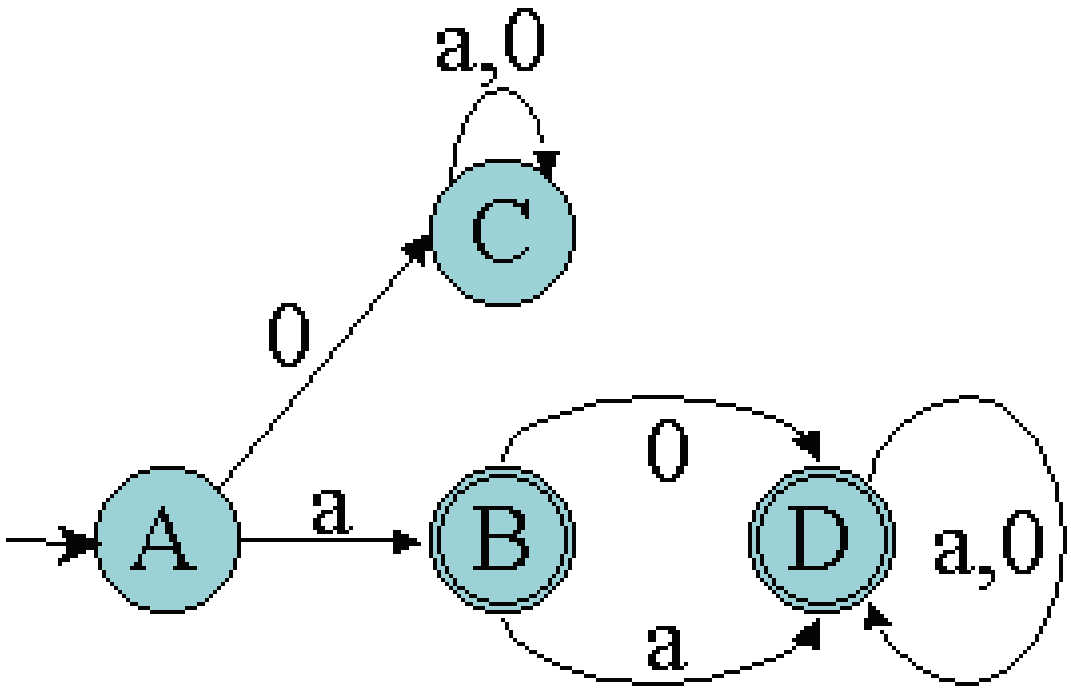


Algorithme de Moore de minimisation d'un AFD

Construction d'un AFN minimal $T' = (A, Q', I', F', \mu')$ à partir d'un AFD $T = (A, Q, I, F, \mu)$

- Partitionner Q en deux groupes : F et $Q - F$. $Q' = F \cup Q - F$ ($\equiv_0$)
- Tant que Q' est modifié, faire
  - Calculer $\equiv_{i+1}$ à partir de $\equiv_i$
  - Q' partition de Q selon $\equiv_{i+1}$
- Fin tant que
- $I' = q \in Q'$ tel que $I \in q$
- $F' = \{q \in Q' | \exists f \in F \text{ et } f \in q\}$
- $T' =$ transitions sur les classes d'équivalences
- Émondage de l'automate obtenu.

Exemple de minimisation à partir d'un AFD (1)



Exemple de minimisation à partir d'un AFD (2)

Partionnement initial : F, Q-F

	A	B	C	D
ϵ	1	2	1	2
a				
0				
bilan1				

Exemple de minimisation à partir d'un AFD (2)

Calcul des $M(a, E)$ **avec** $a \in A$ **et** $E \in \{F, Q - F\}$

	A	B	C	D
ϵ	1	2	1	2
a	2	2	1	2
0	1	2	1	2
bilan1				

Exemple de minimisation à partir d'un AFD (3)

Bilan1

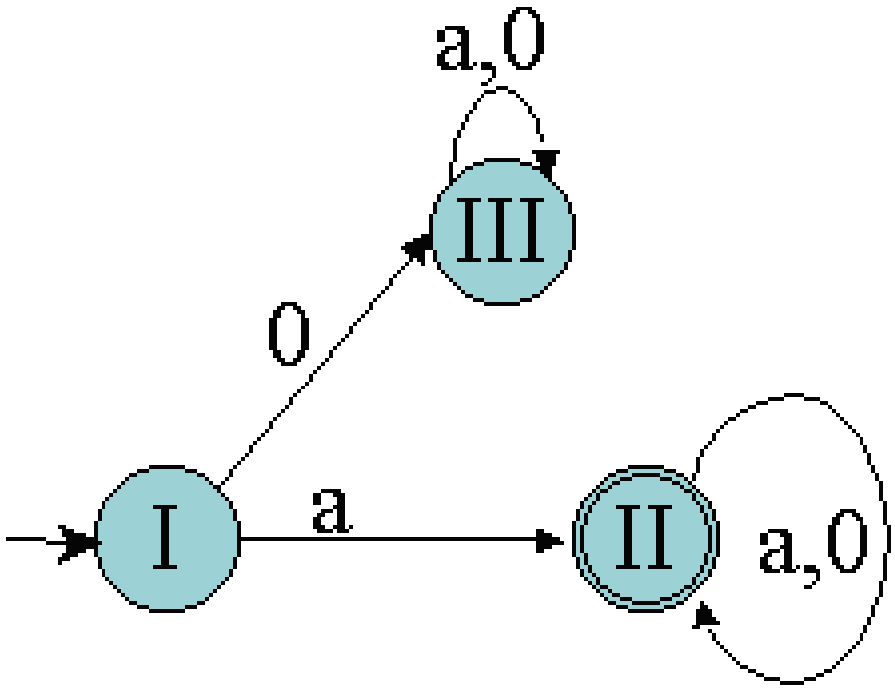
	A	B	C	D
ϵ	1	2	1	2
a	2	2	1	2
0	1	2	1	2
bilan1	1	2	3	2

Exemple de minimisation à partir d'un AFD (4)

Réitération

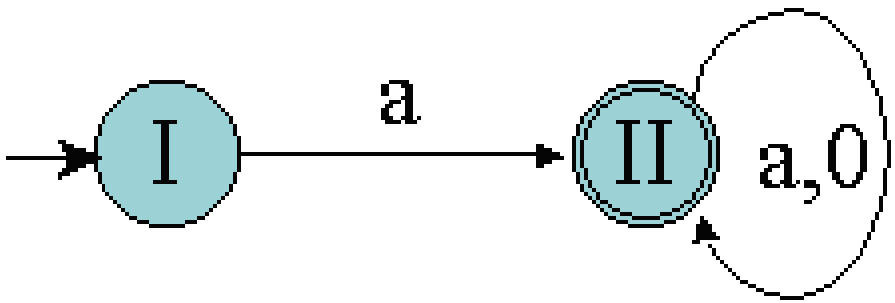
	A	B	C	D
ϵ	1	2	1	2
a	2	2	1	2
0	1	2	1	2
bilan1	1	2	3	2
a	2	2	3	2
0	3	2	3	2
bilan2	1	2	3	2

Exemple de minimisation à partir d'un AFD (5)



Exemple de minimisation à partir d'un AFD (6)

Suppression des états stériles et non-accessibles



Pré-traitement pour la minimisation d'un AFD

Objectif : réduire le coût de la minimisation

- Appliquer l'algorithme d'émondage avant la minimisation ;
- Supprimer les états qui ont les mêmes transitions pour n'en garder qu'une seule.

Minimalité d'un AFD / Équivalence entre AFD

Un AFD est-il minimal ?

Appliquer l'algorithme de minimisation de Moore.

Si les deux automates sont identiques (à la numérotation des états près) alors l'automate d'origine est minimal.

Deux AFD sont-ils équivalents ?

Il n'existe qu'un unique AFD minimal pour un langage donné.

Minimiser les 2 AFD, l'automate minimal obtenu doit être le même.

Théorème du produit cartésien

Théorème 4.9 - Théorème du produit cartésien

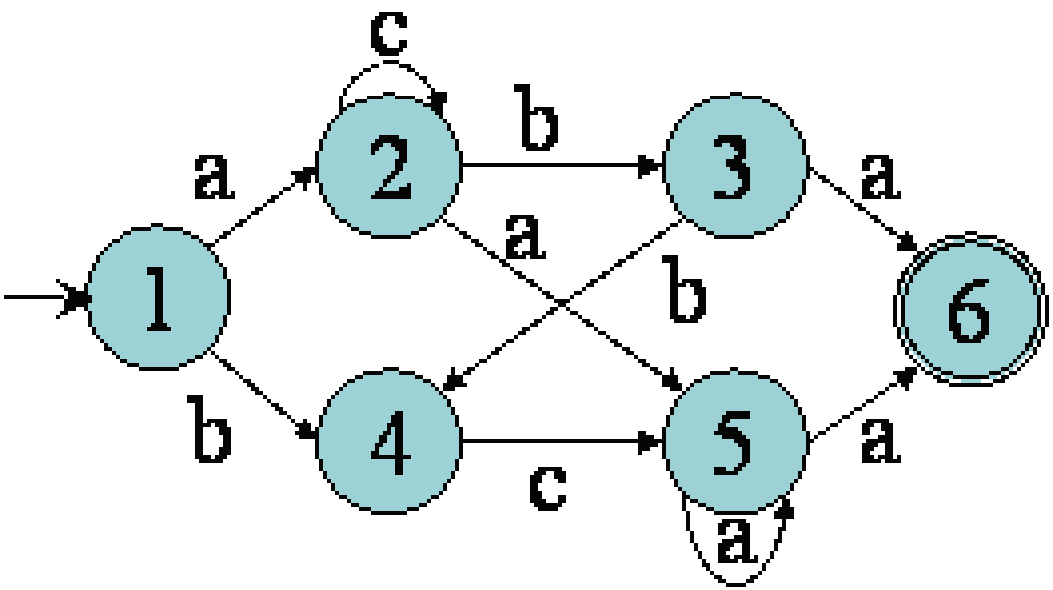
Soit L_1 et L_2 deux langages reconnaissables, alors le produit cartésien de L_1 et L_2 noté $L_1 \times L_2$ est reconnaissable.
Autrement dit, si $L_1 \in Rec(A^*)$ et $L_2 \in Rec(A^*)$ alors $L_1 \times L_2 \in Rec(A^*)$.

L_1 et L_2 reconnaissables : $\exists\ T_1 = (A_1, Q_1, I_1, F_1, \mu_1)$ et $T_2 = (A_2, Q_2, I_2, F_2, \mu_2)$ de manière à ce que $Q_1 \cap Q_2 = \emptyset$
$L_1 \times L_2$ est reconnaissable par $T = (A, Q, I, F, \mu)$:

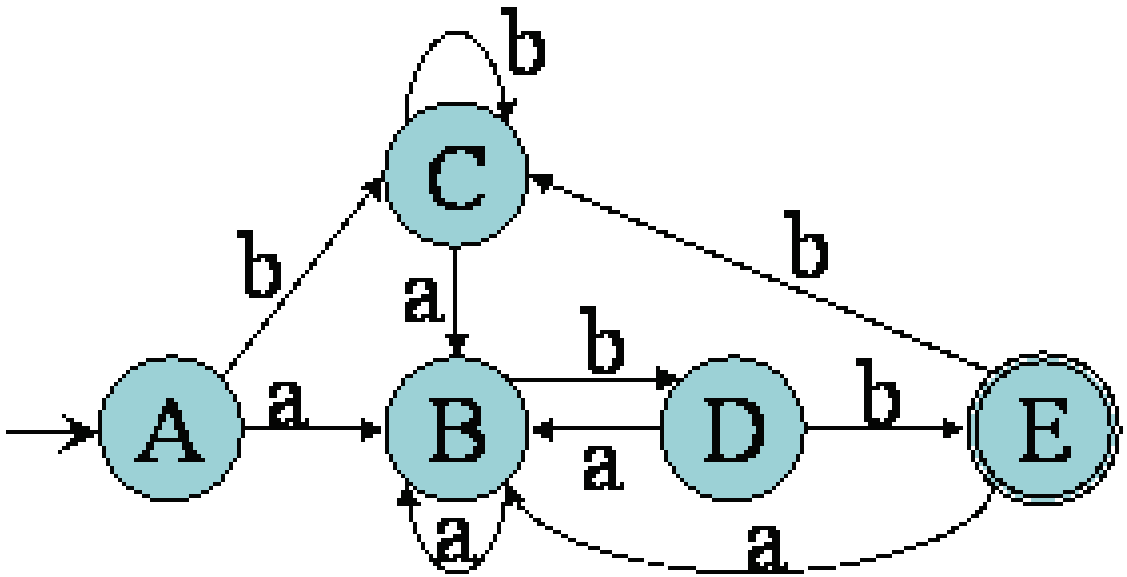
- $A = A_1 \cup A_2$
- $Q = Q_1 \cup Q_2$
- $I = I_1$
- $F = F_2$ si $\forall i \in I_2,\ i \notin F_2$
  $F = F_1 \cup F_2$ sinon
- $\mu = \mu_1 \cup \mu_2 \cup \mu'$ avec
  $\mu' = \{(q1, a, q') \mid q1 \in F_1, (i, a, q') \in \mu_2, i \in I_2\}$

On peut supprimer les états I_2 s'ils ne sont pas utiles (si non accessibles)

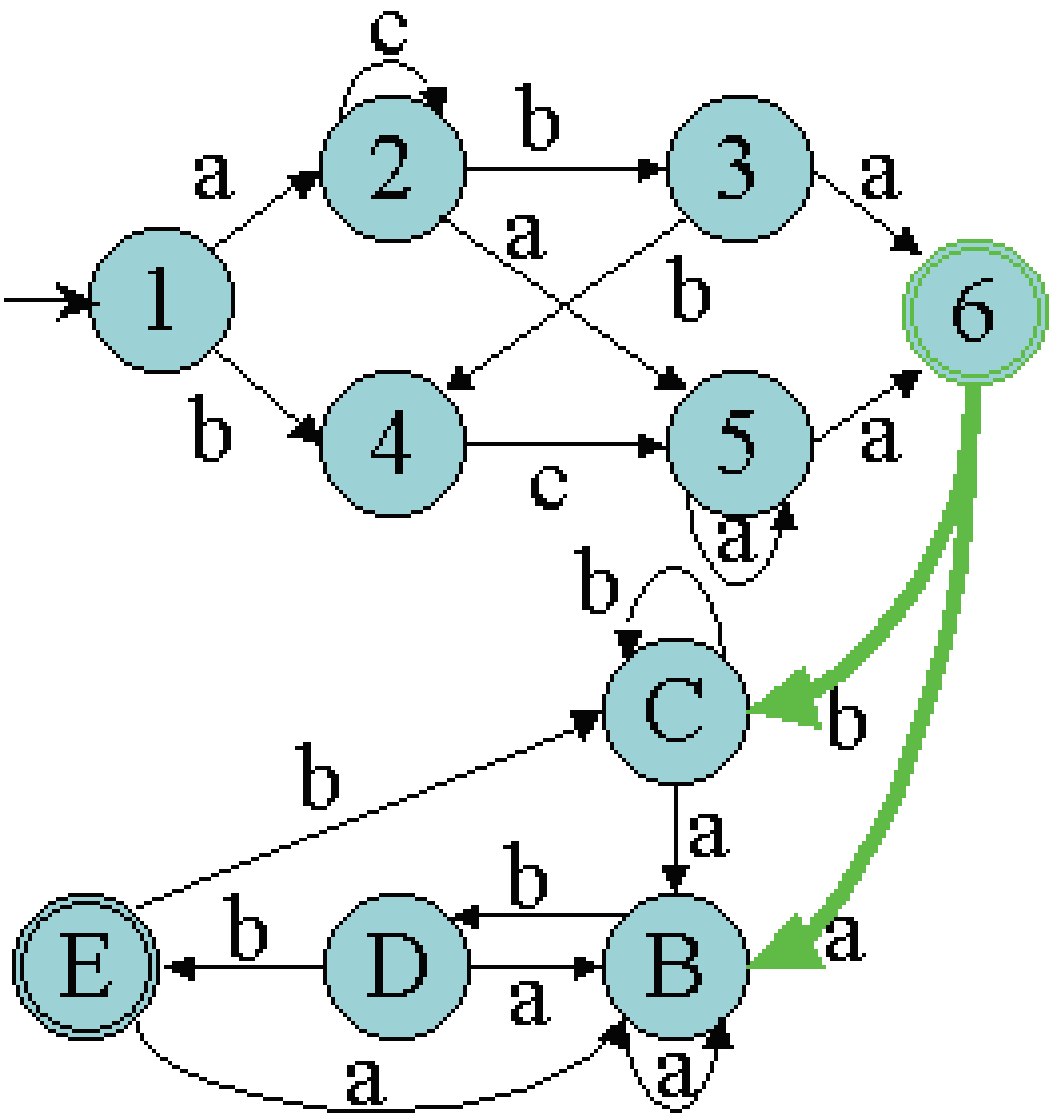
Produit cartésien : exemple



Produit cartésien : exemple



Produit cartésien : exemple



Théorème de la mise à l'étoile

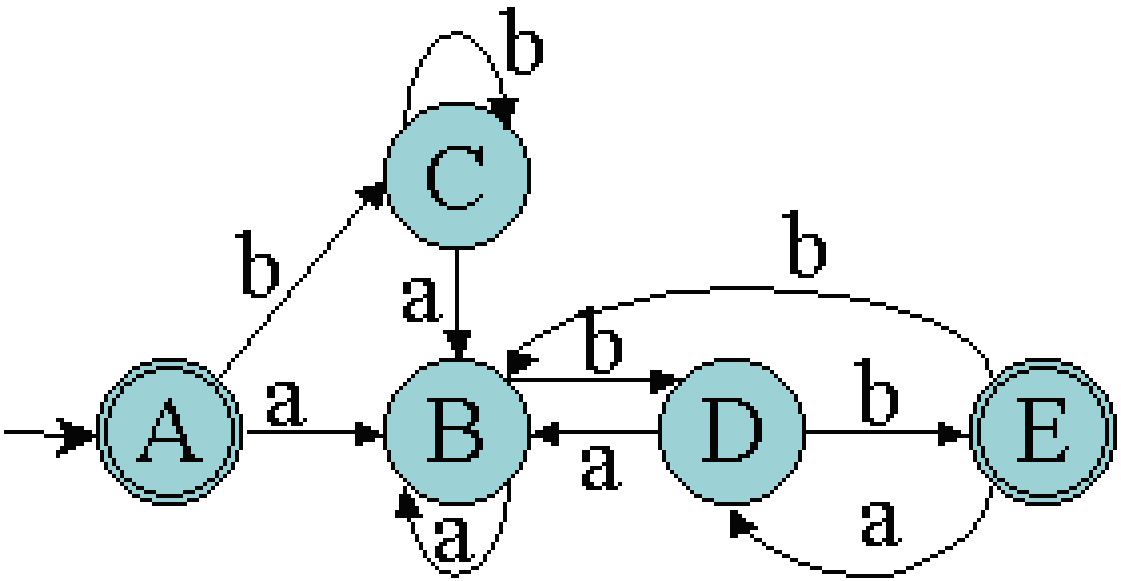
Théorème 4.10 - Théorème de la mise à l'étoile

Soit L un langage reconnaissable, alors la mise à l'étoile de L, noté L^* est reconnaissable.
Autrement dit, si $L \in Rec(A^*)$ alors $L^* \in Rec(A^*)$.

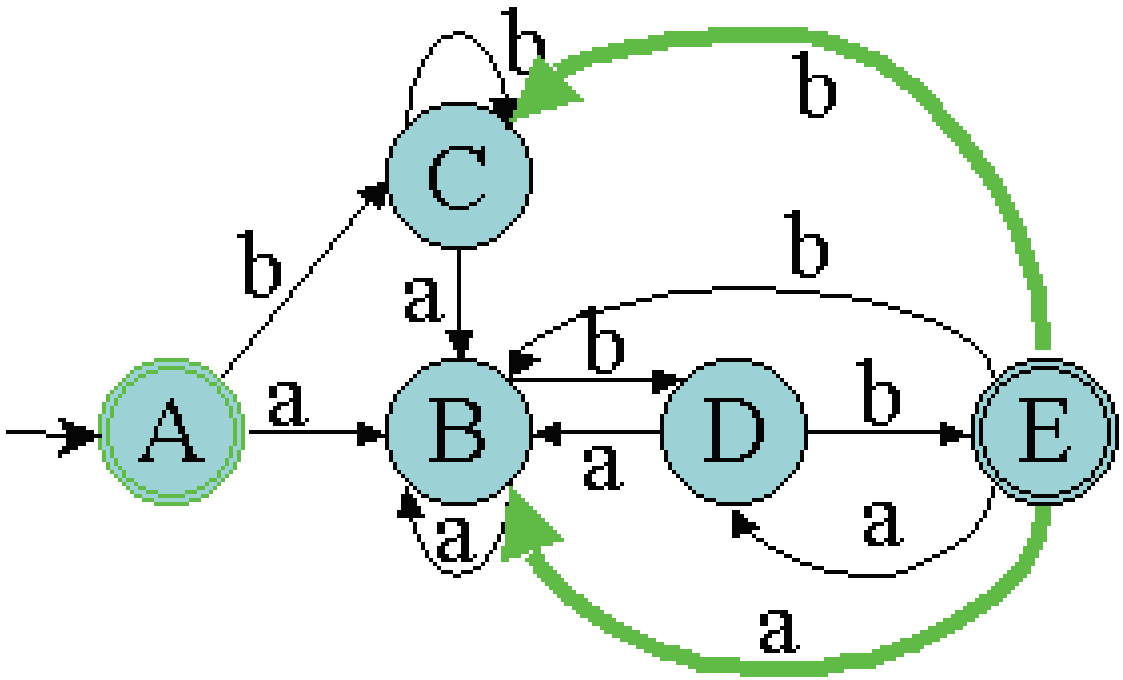
L reconnaissable : $\exists\ T = (A, Q, I, F, \mu)$ un automate fini standard
L^* est reconnaissable par $T' = (A, Q', I', F', \mu')$:

- $Q' = Q$
- $I' = I = \{i\}$
- $F' = F \cup I$
- $\mu' = \mu \cup \{(q, a, q') \mid q \in F, (i, a, q') \in \mu\}$

Mise à l'étoile : exemple



Mise à l'étoile : exemple



Théorème de l'union

Théorème 4.11 - Théorème de l'union

Soit L_1 et L_2 deux langages reconnaissables, alors l'union de L_1 et L_2 noté $L_1 \cup L_2$ est reconnaissable.
Autrement dit, si $L_1 \in Rec(A^*)$ et $L_2 \in Rec(A^*)$ alors $L_1 \cup L_2 \in Rec(A^*)$.

L_1 et L_2 reconnaissables : $\exists\ T_1 = (A_1, Q_1, I_1, F_1, \mu_1)$ et $T_2 = (A_2, Q_2, I_2, F_2, \mu_2)$ de manière à ce que $Q_1 \cap Q_2 = \emptyset$
$L_1 \cup L_2$ est reconnaissable par $T' = (A, Q, I, F, \mu)$:

- $A = A_1 \cup A_2$
- $Q = Q_1 \cup Q_2$
- $F = I_1 \cup I_2$
- $F = F_1 \cup F_2$
- $\mu = \mu_1 \cup \mu_2$

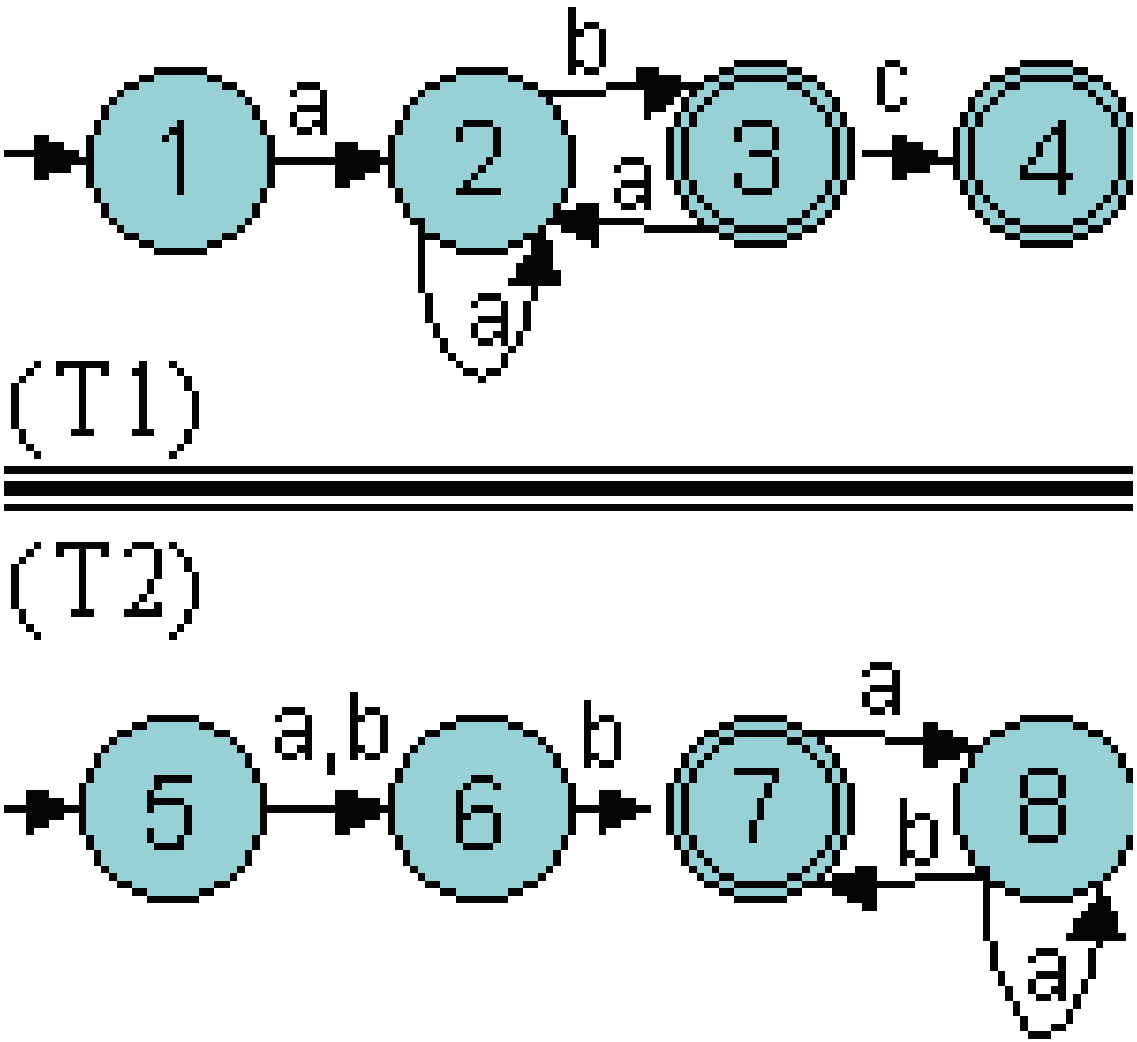
Algorithme de l'union pour obtenir un AFD

Cas de 2 AFD émondés : $T_1 = (A_1, Q_1, I_1, F_1, \mu_1)$ et $T_2 = (A_2, Q_2, I_2, F_2, \mu_2)$
Construction d'un AFD $T = (A, Q, I, F, \mu)$:

- $Q \subseteq Q_1 \times Q_2$
- $I = \{(i_1, i_2)\}$
- Pour chaque état créé (p_1, p_2), et chaque symbole a de $A_1 \cup A_2$, faire :
  Créer la transition :
  - $((p_1, p_2), a, (q_1, q_2))$ si $(p_1, a, q_1) \in \mu_1$ et $(p_2, a, q_2) \in \mu_2$
  - $((p_1, p_2), a, (q_1, \emptyset))$ si $(p_1, a, q_1) \in \mu_1$ et $(p_2, a, q_2) \notin \mu_2$
  - $((p_1, p_2), a, (\emptyset, q_2))$ si $(p_1, a, q_1) \notin \mu_1$ et $(q_2, a, q_2) \in \mu_2$
- Fin Faire

AFD à minimiser si besoin !

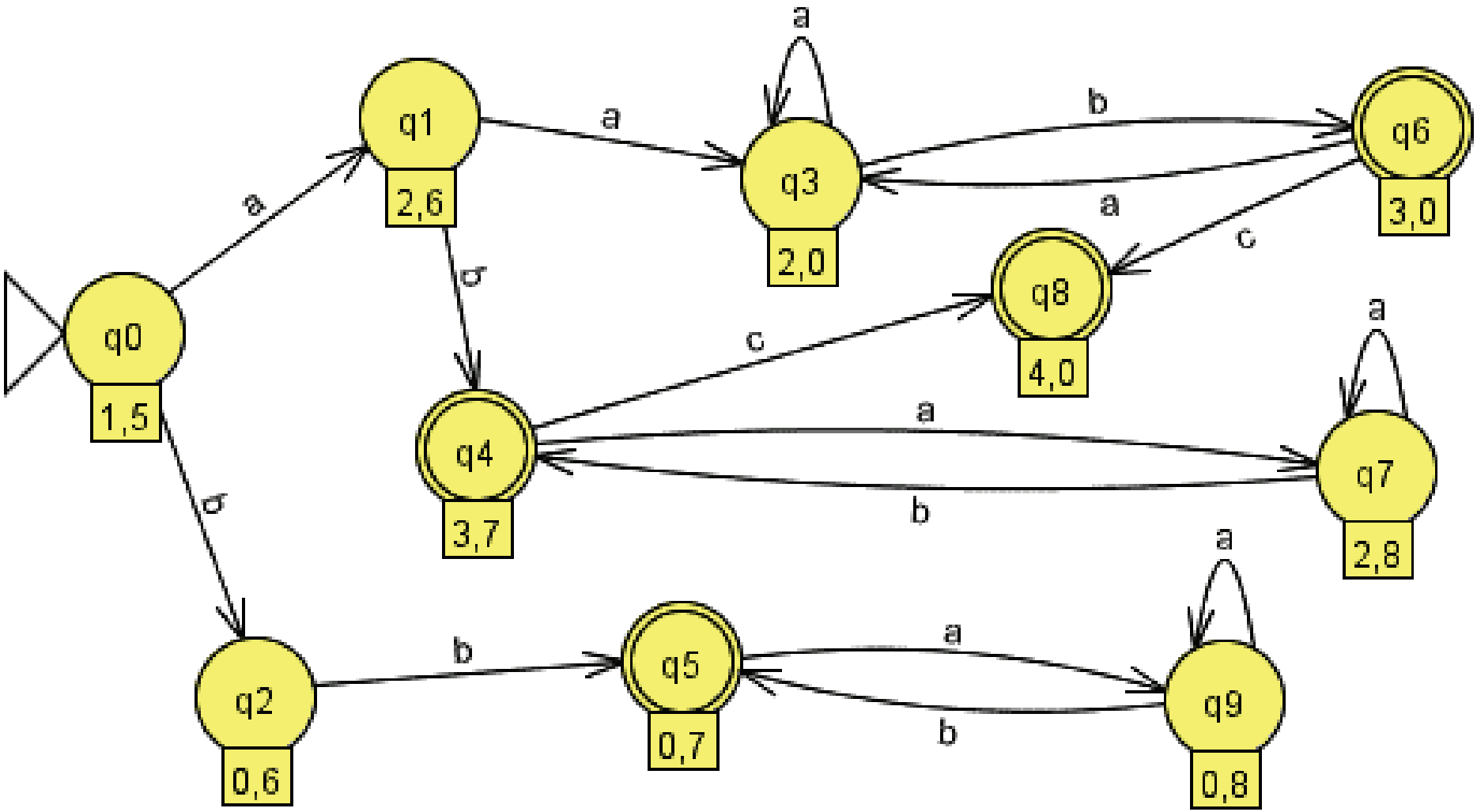
Union : exemple - les 2 AFD



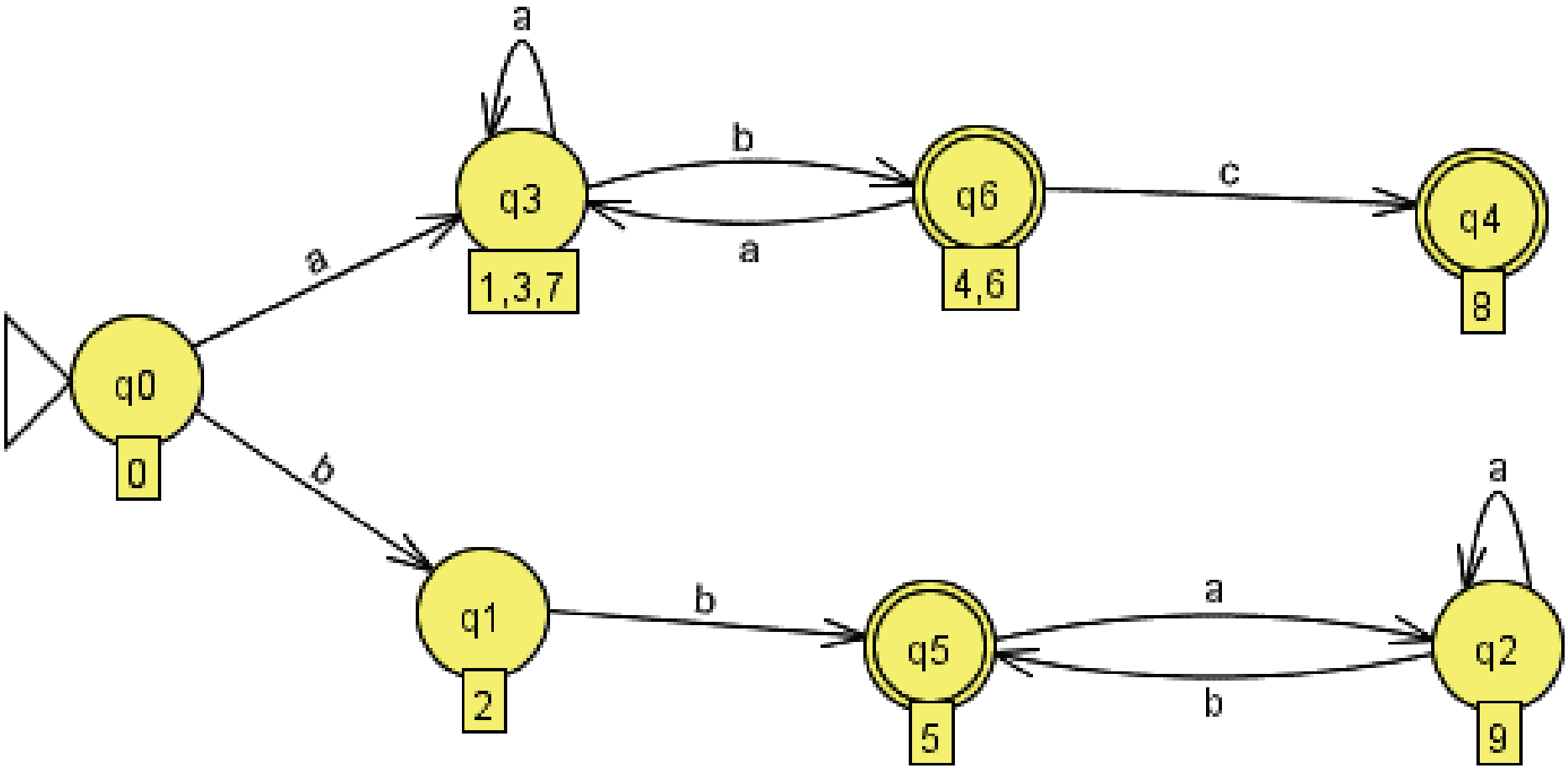
Union : exemple - l'AFD résultat (non minimal)

μ	a	b	c
$\{1,5\} \in I$	$\{2,6\}$	$\{0,6\}$	$\{0,0\}$
$\{2,6\}$	$\{2,0\}$	$\{3,7\}$	$\{0,0\}$
$\{0,6\}$	$\{0,0\}$	$\{0,7\}$	$\{0,0\}$
$\{2,0\}$	$\{2,0\}$	$\{3,0\}$	$\{0,0\}$
$\{3,7\} \in F$	$\{2,8\}$	$\{0,0\}$	$\{4,0\}$
$\{0,7\} \in F$	$\{0,8\}$	$\{0,0\}$	$\{0,0\}$
$\{3,0\} \in F$	$\{2,0\}$	$\{0,0\}$	$\{4,0\}$
$\{2,8\}$	$\{2,8\}$	$\{3,7\}$	$\{0,0\}$
$\{4,0\} \in F$	$\{0,0\}$	$\{0,0\}$	$\{0,0\}$
$\{0,8\}$	$\{0,8\}$	$\{0,7\}$	$\{0,0\}$

Union : exemple - l'AFD résultat (non minimal)



Union : exemple - l'AFD après minimalisation



Théorème du langage fini

Lemme 4.1 - Lemme du langage réduit à un mot/un symbol

Tout langage réduit à un mot/un symbole est reconnaissable.

Théorème 4.12 - Théorème du langage fini

Tout langage fini est reconnaissable.

Théorème du complémentaire

Théorème 4.13 - Théorème du complémentaire

Soit L un langage reconnaissable sur un alphabet A, alors le complémentaire de L sur A^*, $Compl(L)$ est reconnaissable

Si L est reconnaissable, alors il est reconnaissable avec un automate fini déterministe complet $T = (A, Q, I, F, \mu)$ $Compl(L)$ est reconnaissable par l'automate $T' = (A, Q, I, Q - F, \mu)$

Conclusion

Théorème 4.13

Tout langage rationnel est reconnaissable par un automate fini

Démonstration : les langages rationnels sont obtenus à partir des langages finis par union, produit et fermeture

Question : les langages reconnaissables sont-ils rationnels ?

Oui. Voir chapitre suivante