Chapitre 2
Les automates finis

2.1 Introduction

- Automates finis : première modélisation de la notion de procédure effective.(Ont aussi d'autres applications).

- Dérivation de la notion d'automate fini de celle de programme exécuté sur un ordinateur : état, état initial, fonction de transition.

- Hypothèse du nombre d'états fini. Conséquence : séquences d'états finies ou cycliques.

- Problème de la représentation des données : nombre de données différentes limitées car nombre d'états initiaux possibles fini.

Représentation des données.

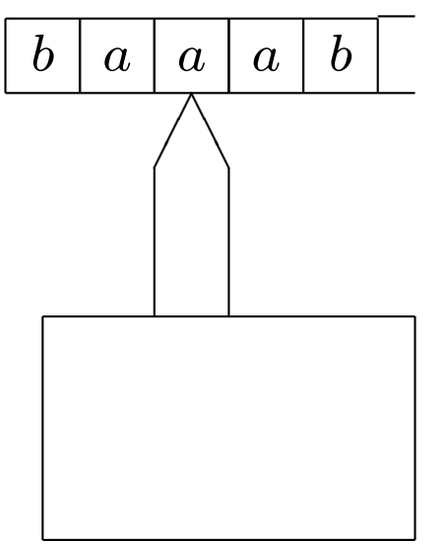
- Problème : reconnaître un langage.
- Données : mot.
- On supposera le mot fourni caractère par caractère, la machine traitant un caractère à chaque cycle et s'arrêtant à la fin du mot.

2.2 Description

- Ruban d'entrée.
- Ensemble d'états :
  - état initial,
  - états accepteurs.
- Mécanisme d'exécution.

ruban :

tête :



2.3 Formalisation

Un automate fini déterministe est défini par un quintuplet $M = (Q, \Sigma, \delta, s, F)$, où

- Q est un ensemble fini d'états,
- Σ est un alphabet,
- $\delta : Q \times \Sigma \to Q$ est la fonction de transition,
- $s \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états accepteurs.

Définition du langage accepté

- Configuration : $(q, w) \in Q \times \Sigma^*$.
- Configuration dérivable en une étape : $(q, w) \vdash_M (q', w')$.
- Configuration dérivable (en plusieurs étapes) : $(q, w) \vdash_M^* (q', w')$.
- Exécution d'un automate :

$$(s, w) \vdash (q_1, w_1) \vdash (q_2, w_2) \vdash \cdots \vdash (q_n, \varepsilon)$$

- Mot accepté :

$$(s, w) \vdash_M^* (q, \varepsilon)$$

  et $q \in F$.

- Langage accepté $L(M)$:

$$\{w \in \Sigma^* \mid (s, w) \vdash_M^* (q, \varepsilon) \text{ avec } q \in F\}.$$

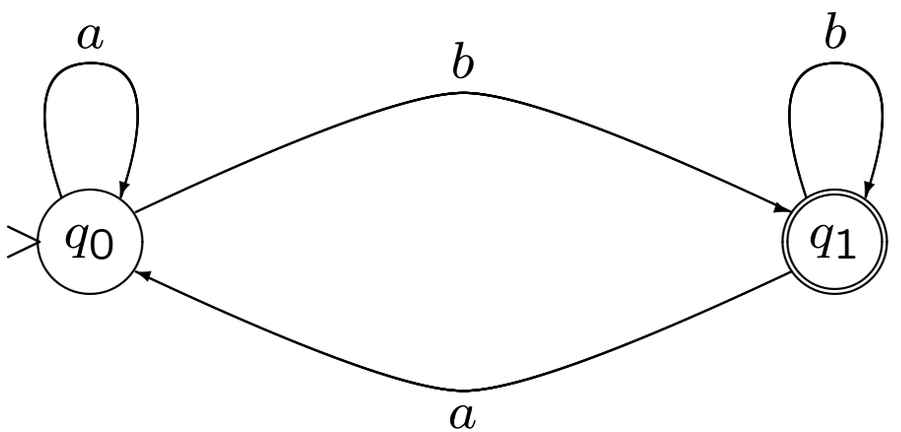
2.4 Exemples

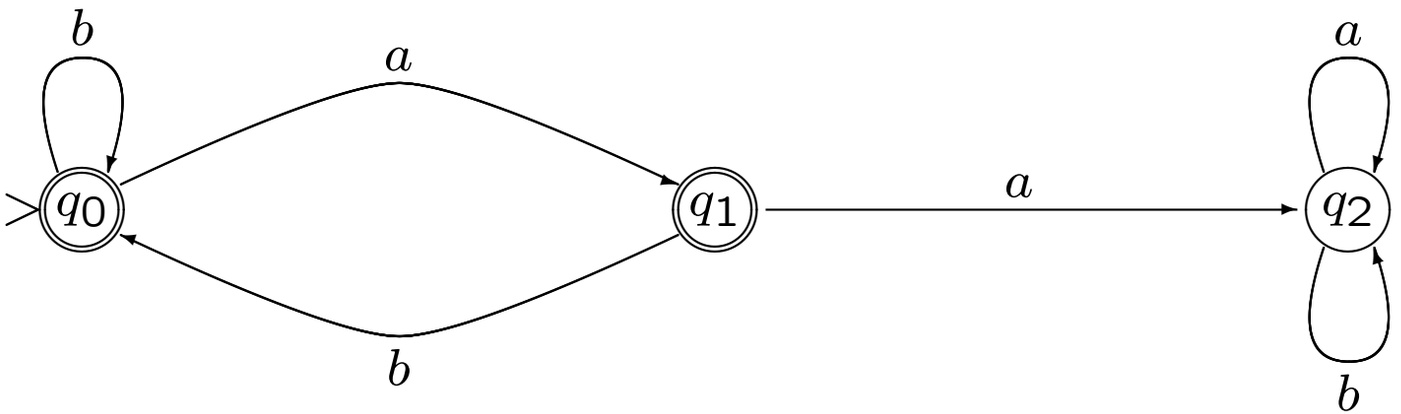
Mots se terminant par b :

δ :

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_1

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$s = q_0$
$F = \{q_1\}$





$\{w \mid w$ ne contient pas 2 a consécutifs$\}$.

2.5 Les automates finis non déterministes

Automates qui peuvent *choisir* parmi plusieurs transitions.

Motivation :

- Voir les conséquences de l'extension d'une définition donnée.
- Faciliter la description de langages par les automates finis.
- Le concept de non-déterminisme est généralement utile.

Description

Les automates finis non déterministes sont des automates finis où l'on permet :

- plusieurs transitions correspondant à la même lettre dans chaque état,
- des transitions sur le mot vide (c'est-à-dire sans avancer dans le mot d'entrée),
- des transitions sur des mots de longueur supérieure à 1 (regroupement de transitions).

Acceptent si au moins une exécution accepte.

Formalisation

Un automate fini non déterministe est défini par un quintuplet $M = (Q, \Sigma, \Delta, s, F)$, où

- Q est un ensemble d'états,
- Σ est un alphabet,
- $\Delta \subset (Q \times \Sigma^* \times Q)$ est la relation de transition,
- $s \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états accepteurs.

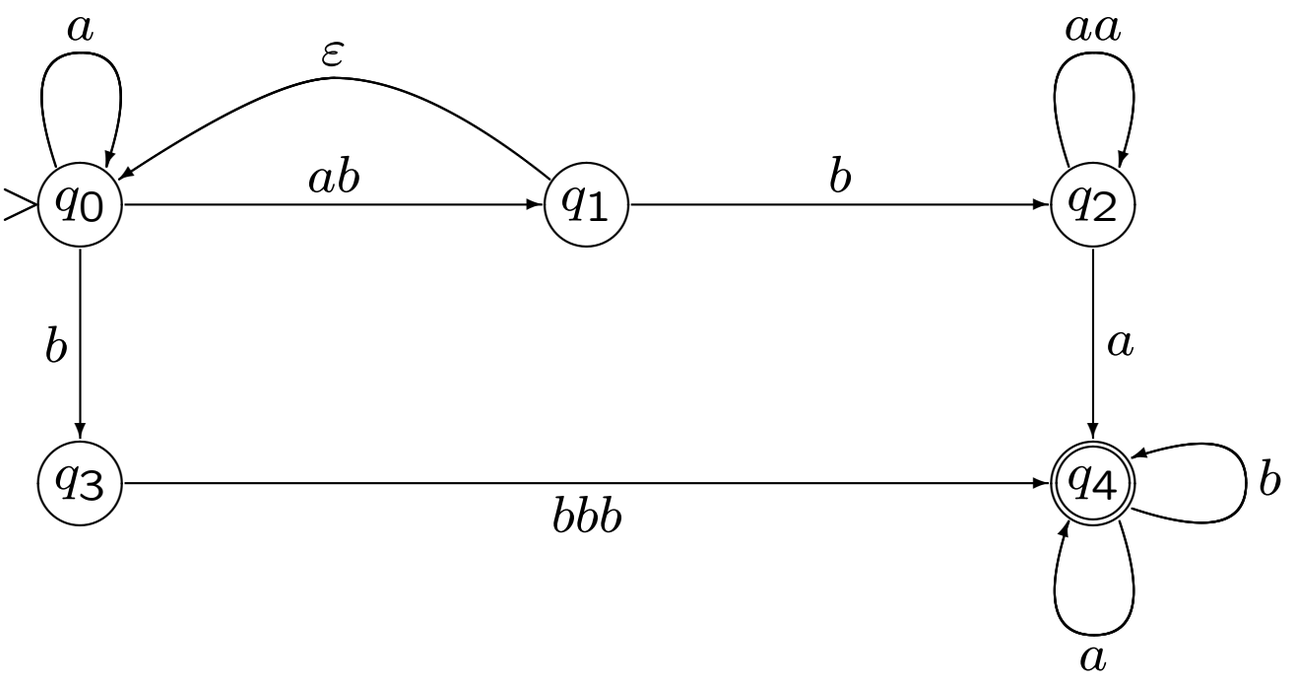
Définition du langage accepté

La configuration (q', w') est dérivable en une étape de la configuration (q, w) par la machine M ($(q, w) \vdash_M (q', w')$) si

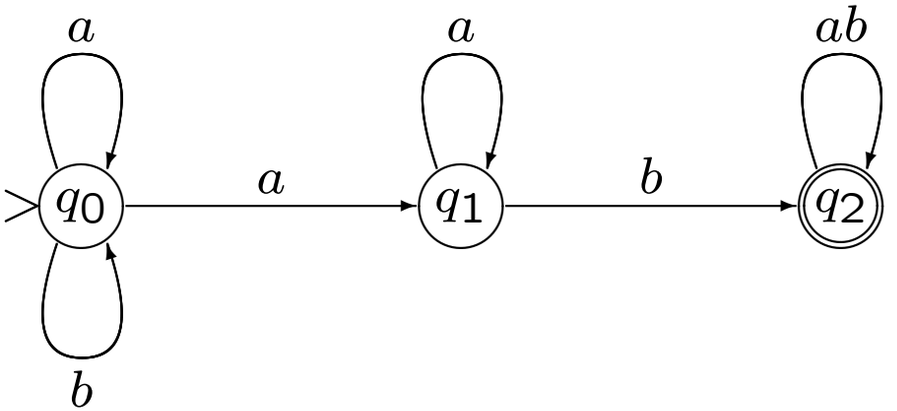
- $w = uw'$ (le mot w commence par un préfixe $u \in \Sigma^*$),
- $(q, u, q') \in \Delta$ (le triplet (q, u, q') est dans la relation de transition Δ).

Un mot est accepté s'il existe une exécution pour ce mot menant à un état accepteur.

Exemples



$$L(M) = ((a \cup ab)^* bbbb\Sigma^*) \cup ((a \cup ab)^* abb(aa)^* a\Sigma^*)$$



$$L(M) = \Sigma^* ab(ab)^*$$

Mots se terminant par au moins une répétition de ab.

2.6 Elimination du non-déterminisme

Définition

Deux automates M_1 et M_2 sont équivalents s'ils acceptent le même langage, c'est-à-dire si $L(M_1) = L(M_2)$.

Théorème

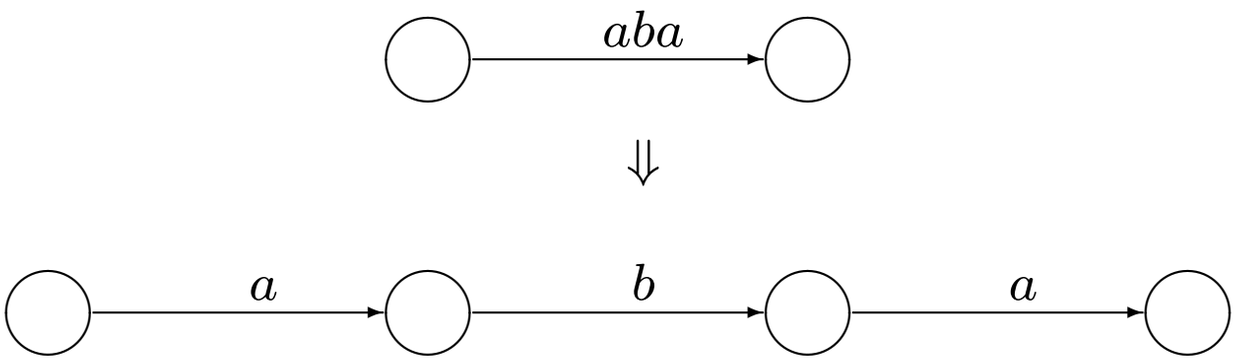
Pour tout automate non déterministe, il est possible de construire un automate déterministe équivalent.

2.6 Principe de la construction

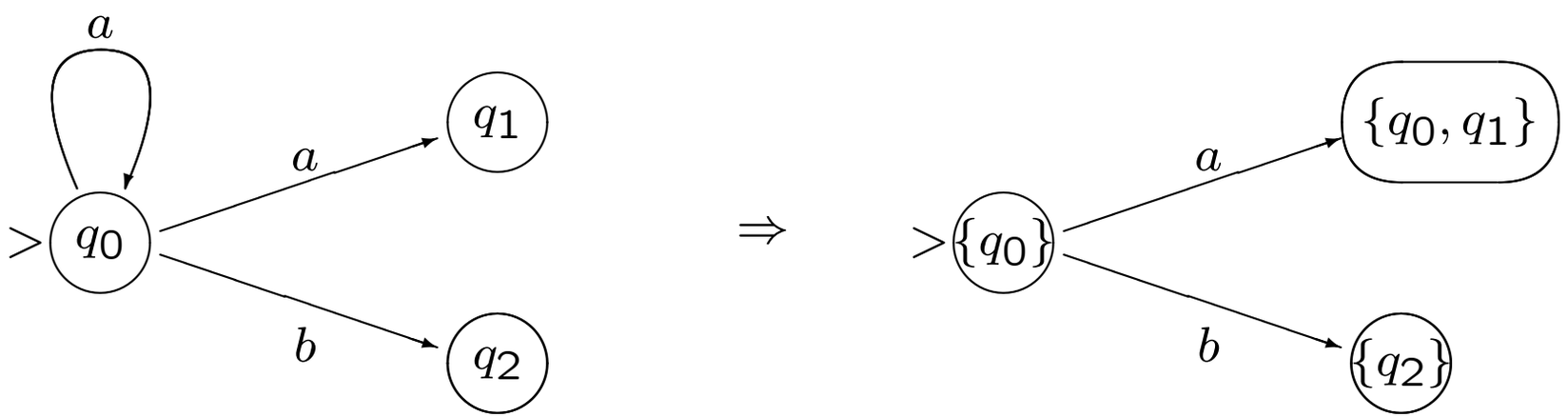
1. Eliminer les transitions de longueur supérieure à 1.

2. Eliminer les transitions compatibles.

Transitions de longueur supérieure à 1.



Transitions compatibles.



Formalisation

Automate non déterministe $M = (Q, \Sigma, \Delta, s, F)$. Construire un automate non déterministe équivalent $M' = (Q', \Sigma, \Delta', s, F)$ tel que $\forall (q, u, q') \in \Delta',\ |u| \leq 1$.

- Initialement $Q' = Q$ et $\Delta' = \Delta$.

- Pour chaque transition $(q, u, q') \in \Delta$ avec $u = \sigma_1 \sigma_2 \ldots \sigma_k$, $(k > 1)$:
  - on enlève cette transition de Δ',
  - on ajoute de nouveaux états $p_1,\ \ldots,\ p_{k-1}$ à Q',
  - on ajoute les nouvelles transitions (q, σ_1, p_1), (p_1, σ_2, p_2), $\ldots$, (p_{k-1}, σ_k, q') à Δ'

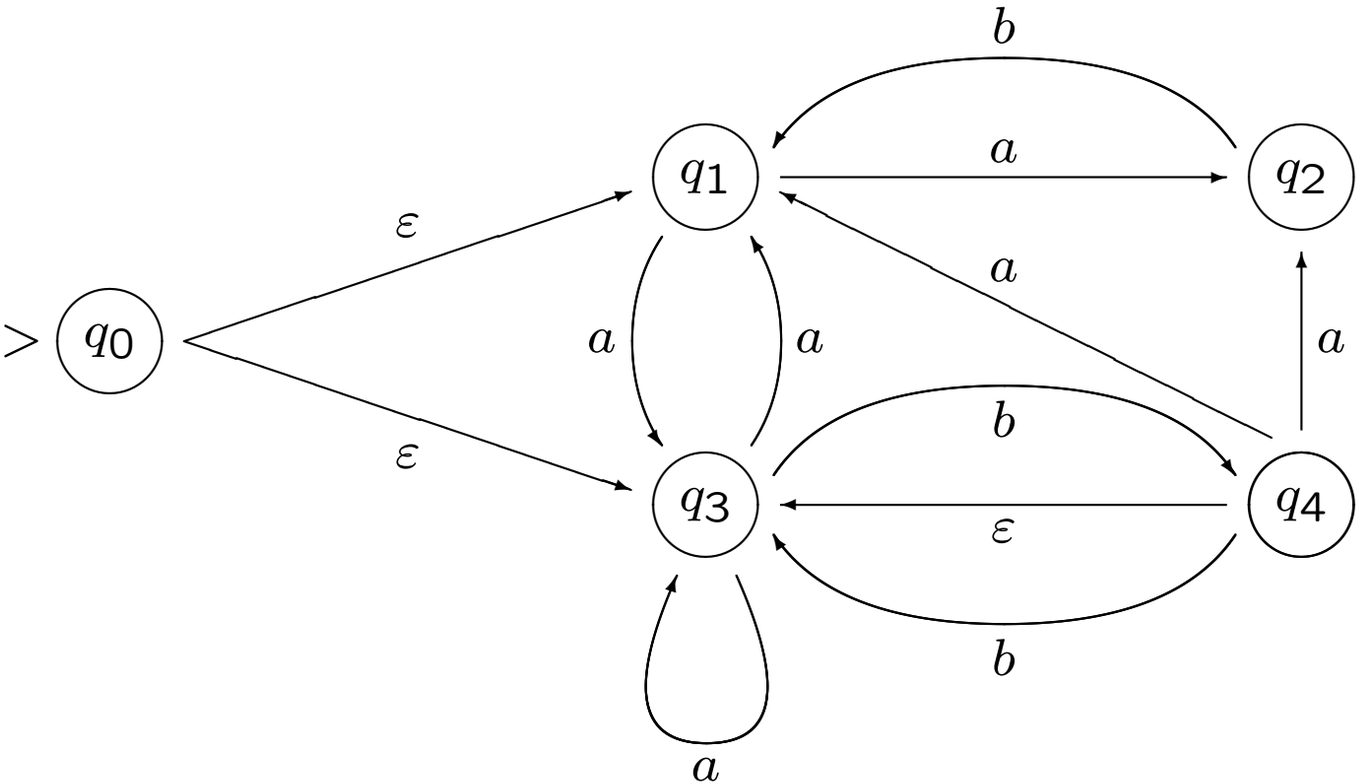
Formalisation

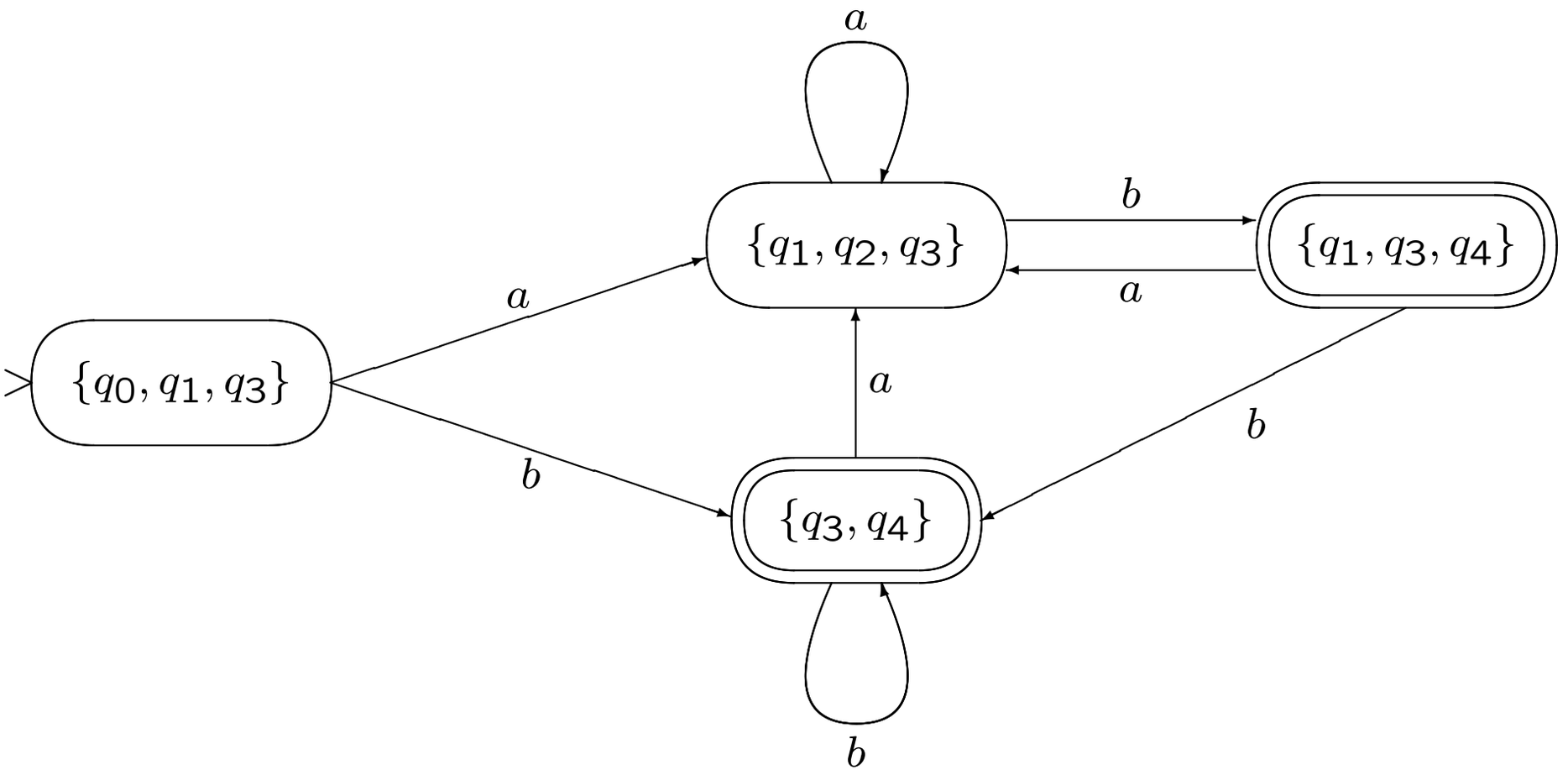
Automate non déterministe $M = (Q, \Sigma, \Delta, s, F)$ tel que $\forall (q, u, q') \in \Delta',\ |u| \leq 1$. Construire un automate déterministe équivalent $M' = (Q', \Sigma, \delta', s, F)$.

$$E(q) = \{p \in Q \mid (q, w) \vdash_M^* (p, w)\}.$$

- $Q' = 2^Q$.
- $s' = E(s)$.
- $\delta'(\mathbf{q}, a) = \bigcup\{E(p) \mid \exists q \in \mathbf{q}\ : (q, a, p) \in \Delta\}$.
- $F' = \{\mathbf{q} \in Q' \mid \mathbf{q} \cap F \neq \emptyset\}$.

Exemple




$\{q_0, q_1, q_3\}$
$\{q_1, q_2, q_3\}$
$\{q_1, q_3, q_4\}$
$\{q_3, q_4\}$
a
a
b
b
a
a
b
b

Autre construction : Uniquement états accessibles

1. Au départ Q' contient l'état initial s'.

2. Les opérations suivantes sont alors répétées jusqu'à ce qu'elles ne modifient plus l'ensemble Q'.

   (a) On choisit un état $\mathbf{q} \in Q'$ auquel l'opération (b) n'a pas encore été appliquée.

   (b) Pour chaque lettre $a \in \Sigma$ on calcule l'état $\mathbf{p}$ tel que $\mathbf{p} = \delta'(\mathbf{q}, a)$. L'état $\mathbf{p}$ est ajouté à Q'.

2.7 Automates finis et expressions régulières

Théorème
Un langage est régulier si et seulement si il est accepté par un automate fini.

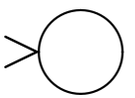
Nous démontrons :

1. Si un langage est dénoté par une expression régulière, il est accepté par un automate fini non déterministe.

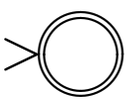
2. Si un langage est accepté par un automate fini non déterministe, il est régulier.

Des expressions aux automates

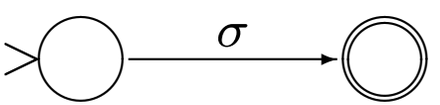
- $\emptyset$



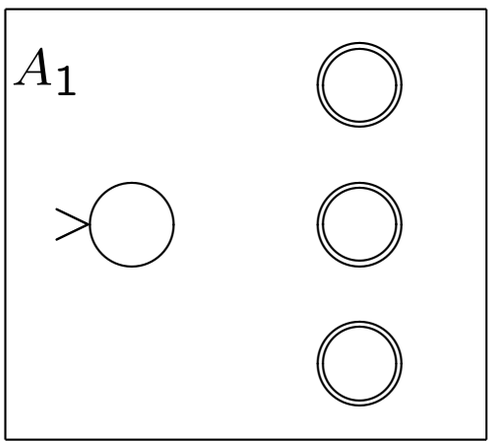
- ε

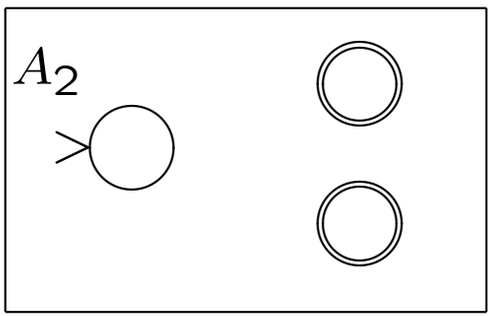


- $\sigma \in \Sigma$

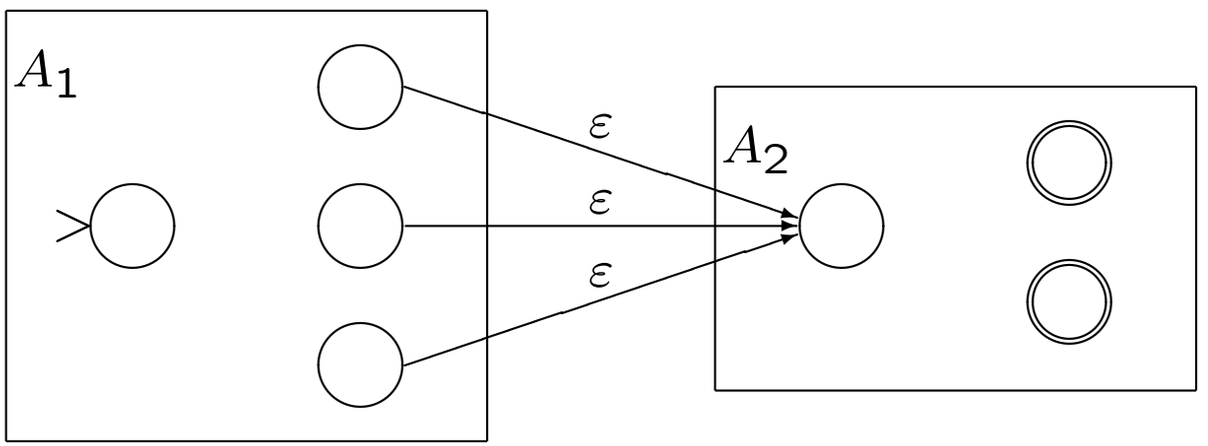


$\alpha_1 : A_1 = (Q_1, \Sigma, \Delta_1, s_1, F_1)$
$\alpha_2 : A_2 = (Q_2, \Sigma, \Delta_2, s_2, F_2)$





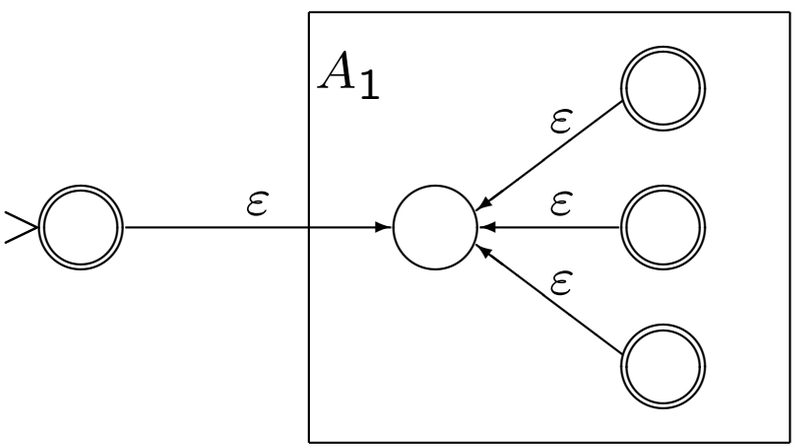
- $\alpha_1 \cdot \alpha_2$



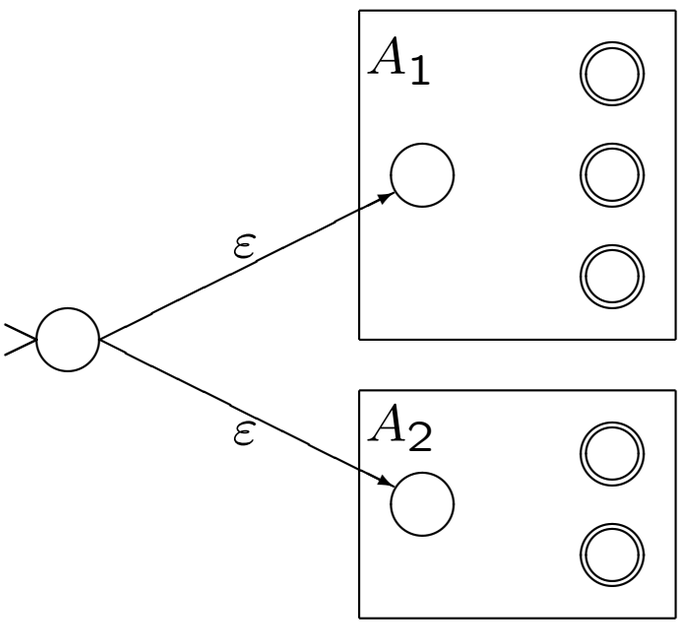
Formellement, $A = (Q, \Sigma, \Delta, s, F)$ où

- $Q = Q_1 \cup Q_2$,
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q, \varepsilon, s_2) \mid q \in F_1\}$,
- $s = s_1$,
- $F = F_2$.

- $\alpha = \alpha_1^*$



- $\alpha = \alpha_1 \cup \alpha_2$



Des automates aux langages réguliers

Idée intuitive :

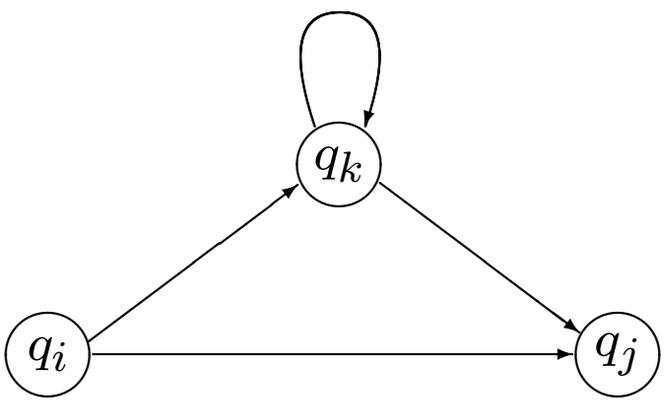
- Construire l'expression régulière correspondant à chaque chemin entre l'état initial et un état accepteur.

- Traiter les boucles à l'aide de l'opérateur $*$.

Définition
Soit un automate M et $Q = \{q_1, q_2, \ldots, q_n\}$ l'ensemble de ses états. Nous désignons par $R(i, j, k)$ l'ensemble des mots permettant de passer de l'état q_i à q_j en passant uniquement par des états $\{q_1, \ldots, q_{k-1}\}$.

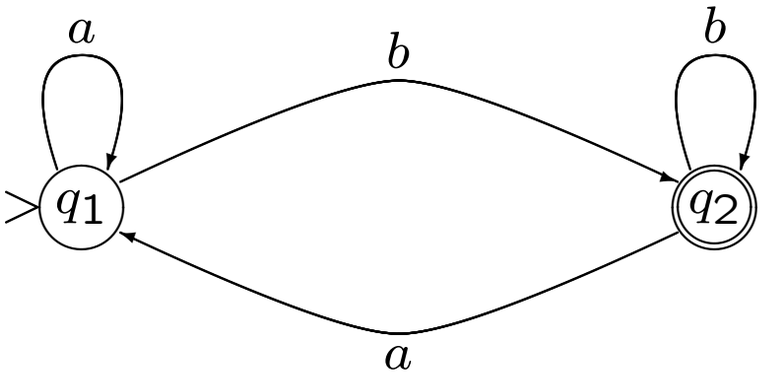
$$R(i,j,1) = \begin{cases} \{w \mid (q_i, w, q_j) \in \Delta\} & \text{si } i \neq j \\ \{\varepsilon\} \cup \{w \mid (q_i, w, q_j) \in \Delta\} & \text{si } i = j \end{cases}$$

$$\begin{aligned} R(i,j,k+1) &= R(i,j,k) \cup \\ &\quad R(i,k,k)R(k,k,k)^*R(k,j,k) \end{aligned}$$



$$L(M) = \bigcup_{q_j \in F} R(1,j,n+1).$$

Exemple



	$k=1$	$k=2$
$R(1,1,k)$	$\varepsilon \cup a$	$(\varepsilon \cup a) \cup (\varepsilon \cup a)(\varepsilon \cup a)^*(\varepsilon \cup a)$
$R(1,2,k)$	b	$b \cup (\varepsilon \cup a)(\varepsilon \cup a)^* b$
$R(2,1,k)$	a	$a \cup a(\varepsilon \cup a)^*(\varepsilon \cup a)$
$R(2,2,k)$	$\varepsilon \cup b$	$(\varepsilon \cup b) \cup a(\varepsilon \cup a)^* b$

Le langage accepté par l'automate est alors $R(1,2,3)$, soit

$$[b \cup (\varepsilon \cup a)(\varepsilon \cup a)^* b] \quad \cup \quad [b \cup (\varepsilon \cup a)(\varepsilon \cup a)^* b]\,[(\varepsilon \cup b) \cup a(\varepsilon \cup a)^* b]^*\,[(\varepsilon \cup b) \cup a(\varepsilon \cup a)^* b]$$