Théorie des langages

Automates finis

Jérôme Delobelle

jerome.delobelle@u-paris.fr

LIPADE - Université de Paris

Comment décrire un langage ?

- Description littéraire

  Ensemble des mots construits sur l'alphabet $\{a,b\}$, de longueur paire

- Enumération (écriture en extension)

$$L = \{\epsilon, aa, bb, ab, ba, aaaa, bbbb, aaab, baaa, \ldots\}$$

- Expression régulière

$$((a+b)(a+b))^*$$

- Grammaire de réecriture

  Ensemble de règles pour générer les mots du langage

- Reconnaisseur (automates)

  Machine permettant de générer tous les mots du langage

Automates finis

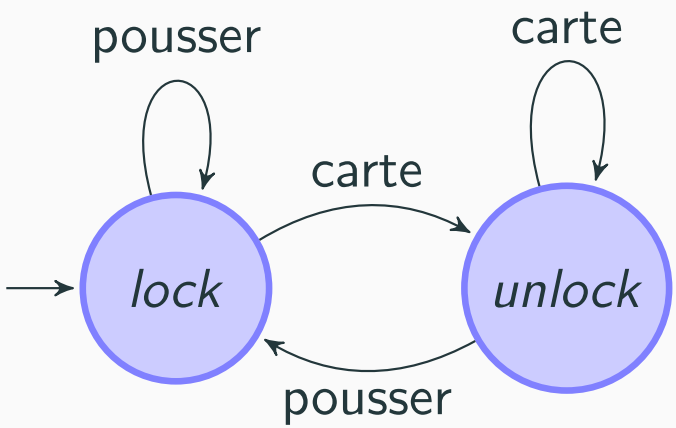
1. Introduction
2. Automate fini déterministe (AFD)
3. Automate fini non déterministe (AFI)
4. Automates complets et émondés
5. Langage reconnu par un automate
6. Lien entre AFD et AFI

Introduction

Introduction

Automate en informatique :

- Principe de suite d'actions (séquence)
- Principe d'état : change selon l'endroit où l'on se situe dans la séquence
- Principe de transition : selon la séquence et l'état, une action est faite et un nouvel état sera atteint (prise de décision)



Attention à ne pas confondre avec la notion d'automate utilisé en mécanique $\simeq$ robot (même si le fonctionnement derrière est globalement similaire)

Introduction

L'automate est dit "fini" quand il possède un nombre fini d'états distincts.

- Les automates finis ont des applications importantes :
  - Définition de certains aspects des langages naturels ou artificiels
  - Description de machines physiques (circuits électroniques, machines à calculer, distributeur d'objets, etc.)
  - Définition de protocoles de communication dans des réseaux
  - Description de systèmes de commandes (comme le système de commandes d'un ascenseur), etc.
- Les automates finis peuvent être utilisés pour calculer des fonctions, ou pour reconnaître des langages.

Introduction

L'automate est dit "fini" quand il possède un nombre fini d'états distincts.

- Les automates finis ont des applications importantes :
  - Définition de certains aspects des langages naturels ou artificiels
  - Description de machines physiques (circuits électroniques, machines à calculer, distributeur d'objets, etc.)
  - Définition de protocoles de communication dans des réseaux
  - Description de systèmes de commandes (comme le système de commandes d'un ascenseur), etc.
- Les automates finis peuvent être utilisés pour calculer des fonctions, ou pour reconnaître des langages.

En THL, les automates finis sont des machines abstraites qui savent **reconnaître l'appartenance ou non d'un mot à un langage régulier donné**.

Automate fini déterministe (AFD)

Automate fini déterministe (AFD)

Automate fini déterministe

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

- Q est un ensemble fini d'états
- Σ est un ensemble fini de symboles (un alphabet)
- $\delta \colon Q \times \Sigma \to Q$ est une fonction de transitions
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble (fini) des états finaux

Automate fini déterministe (AFD)

Automate fini déterministe

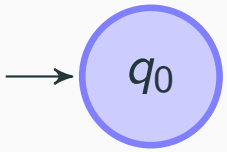
Un **automate fini déterministe (AFD)** est un quintuplet $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ où

- Q est un ensemble fini d'états
- Σ est un ensemble fini de symboles (un alphabet)
- $\delta \colon Q \times \Sigma \to Q$ est une fonction de transitions
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble (fini) des états finaux

Une paire (q, w), où $q \in Q$ est un état, et $w \in \Sigma^*$ est un mot de l'alphabet Σ est appelé une **configuration**.

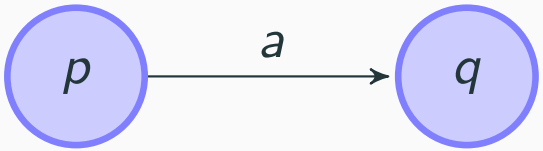
Représentation graphique

- Etat initial



- Etat final (2 notations possible)

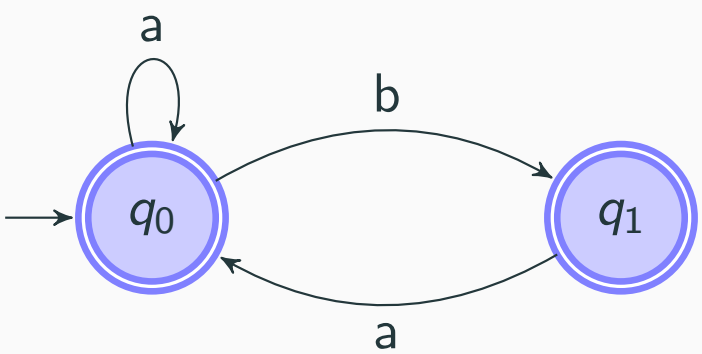
- Transition entre l'état p et q : $\delta(p, a) = q$



Exemple

Soit l'AFD $M_1 = \langle Q, \Sigma, \delta, q_0, F \rangle$ avec :

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- $\delta = \{(q_0, a) \to q_0, (q_0, b) \to q_1, (q_1, a) \to q_0\}$
- $q_0 = \{q_0\}$
- $F = \{q_0, q_1\}$



δ	a	b
$\Leftrightarrow q_0$	q_0	q_1
$\Leftarrow q_1$	q_0	

Table de transitions

Dérivation

Configuration dérivable en une étape

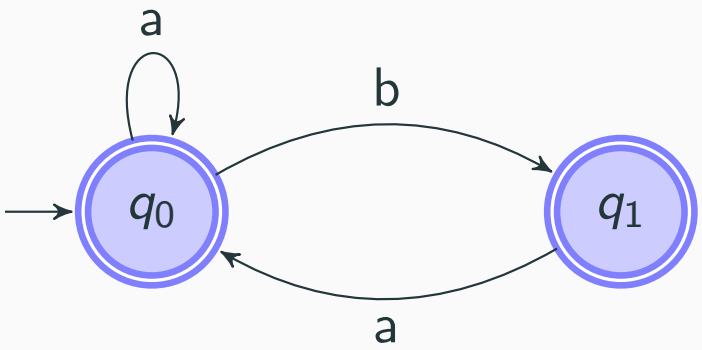
Soit M un automate, $q, q' \in Q$ deux états, $w, w' \in \Sigma^*$ deux mots et (q, w), (q', w') les deux configurations correspondantes.

On dit que la configuration (q', w') est dérivable en une étape de la configuration (q, w) par M, noté $(q, w) \mapsto (q', w')$, si

- $w = xw'$, avec $x \in \Sigma$
- M est dans l'état q
- $q' = \delta(q, x)$

On dit alors qu'on **"lit"** la lettre x.

Exemple



$(q_0, ba) \mapsto (q_1, a)$ est une dérivation en une étape.
$(q_0, aaa) \mapsto (q_0, aa)$ est une dérivation en une étape.
$(q_0, b) \mapsto (q_1, \epsilon)$ est une dérivation en une étape.

$(q_1, bab) \mapsto (q_0, ab)$ n'est pas une dérivation en une étape.
$(q_1, b) \mapsto (q_1, \epsilon)$ n'est pas une dérivation en une étape.
$(q_0, ba) \mapsto (q_0, \epsilon)$ n'est pas une dérivation en une étape.

Dérivation

Configuration dérivable

Soit M un automate, $q, q' \in Q$ deux états, $w, w' \in \Sigma^*$ deux mots et (q, w), (q', w') les deux configurations correspondantes.

On dit que la configuration (q', w') est dérivable de la configuration (q, w) par M, noté $(q, w) \overset{*}{\mapsto} (q', w')$, si $\exists k \geq 0$ et k configurations (q_i, w_i), $1 \leq i \leq k$ telles que

- $(q, w) = (q_1, w_1)$
- $(q', w') = (q_k, w_k)$
- $\forall i$, $1 \leq i \leq k$, $(q_i, w_i) \mapsto (q_{i+1}, w_{i+1})$ est une dérivation en une étape

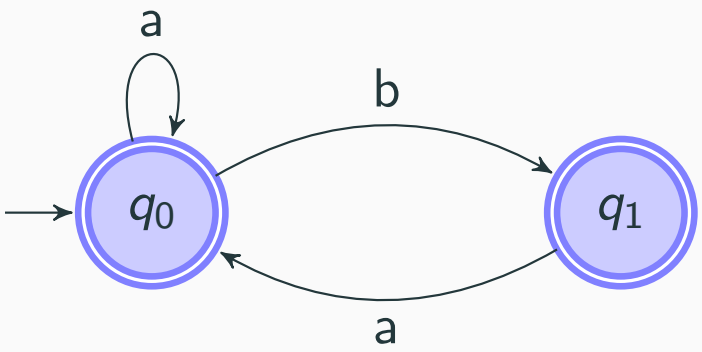
Reconnaissance d'un mot

Reconnaissance d'un mot par un automate

La reconnaissance d'un mot w par un automate M (appelée aussi **exécution** d'un automate M sur un mot w) est la suite des configurations :

$$(q_0, w) \mapsto (q_1, w_1) \mapsto (q_2, w_2) \mapsto \ldots \mapsto (q_n, \epsilon)$$

Exemple



$(q_0, ba) \mapsto (q_1, a) \mapsto (q_0, \epsilon)$ est une dérivation en deux étapes. Le mot ba est donc **reconnu** par l'automate.

Langage reconnu par un automate

Mot accepté par un automate

Un mot w est accepté par un automate si et seulement si

$$(q_0, w) \overset{*}{\mapsto} (q, \epsilon) \text{ avec } q \in F$$

Langage reconnu par un automate

Mot accepté par un automate

Un mot w est accepté par un automate si et seulement si

$$(q_0, w) \overset{*}{\mapsto} (q, \epsilon) \text{ avec } q \in F$$

Langage accepté par un automate

Le langage accepté par un automate M, noté $\mathcal{L}(M)$, est défini par

$$\mathcal{L}(M) = \{w \in \Sigma^* | (q_0, w) \overset{*}{\mapsto} (q, \epsilon) \text{ avec } q \in F\}$$

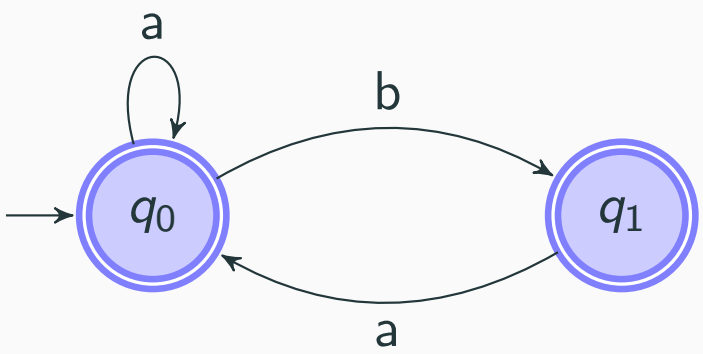
Langage reconnu par un automate

ATTENTION

Plutôt que langage accepté par un automate M, on parle souvent de langage reconnu par l'automate M. Il s'agit pourtant toujours de l'ensemble des mots acceptés par l'automate, et non pas les mots **reconnus** par l'automate.

Exemple

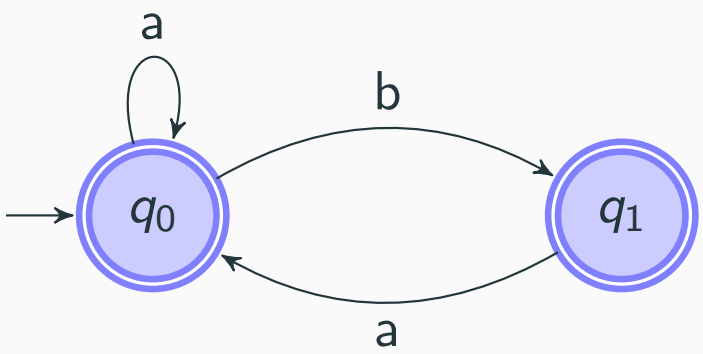
- Automate M_1



- Est-ce que M_1 **reconnait** *aaba* ? **accepte** *aaba* ?

Exemple

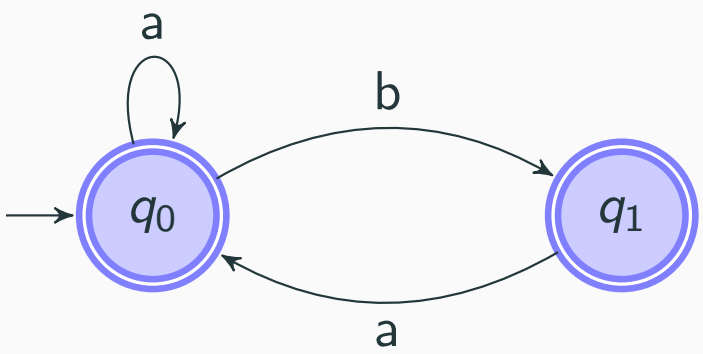
- Automate M_1



- Est-ce que M_1 **reconnait** *aaba* ? **accepte** *aaba* ?
- Est-ce que M_1 **reconnait** *abba* ? **accepte** *abba* ?

Exemple

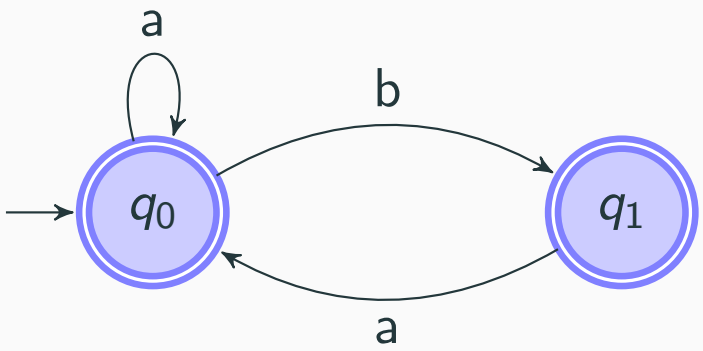
- Automate M_1



- Est-ce que M_1 **reconnait** *aaba* ? **accepte** *aaba* ?
- Est-ce que M_1 **reconnait** *abba* ? **accepte** *abba* ?
- Est-ce que M_1 **reconnait** *baab* ? **accepte** *baab* ?

Exemple

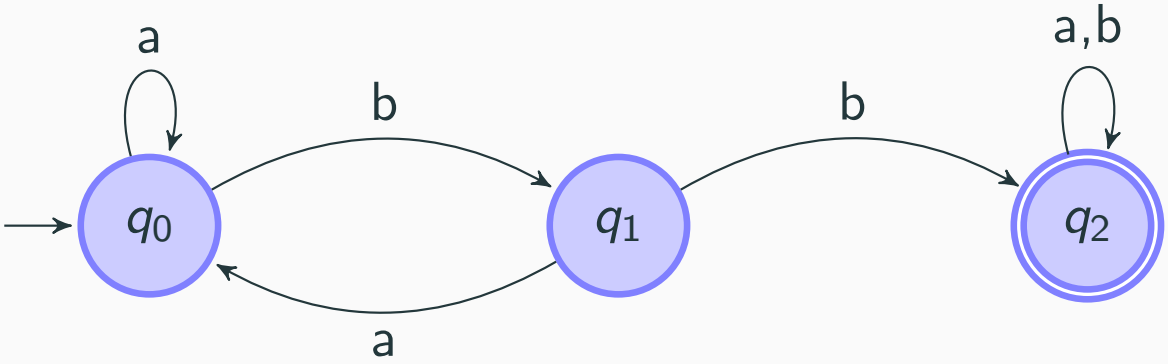
- Automate M_1



- Est-ce que M_1 **reconnait** *aaba* ? **accepte** *aaba* ?
- Est-ce que M_1 **reconnait** *abba* ? **accepte** *abba* ?
- Est-ce que M_1 **reconnait** *baab* ? **accepte** *baab* ?
- $\mathcal{L}(M_1) = \{w \in \{a, b\}^* | w$ ne contient pas deux b consécutifs$\}$

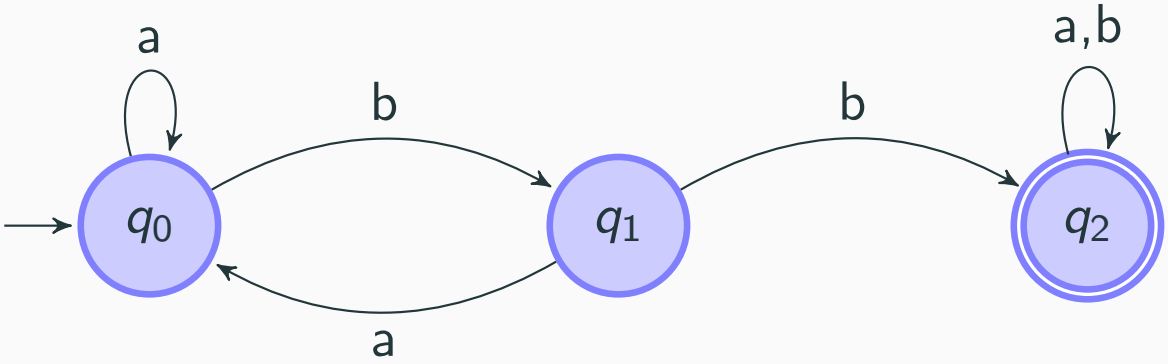
Exemple

- Automate M_2



Exemple

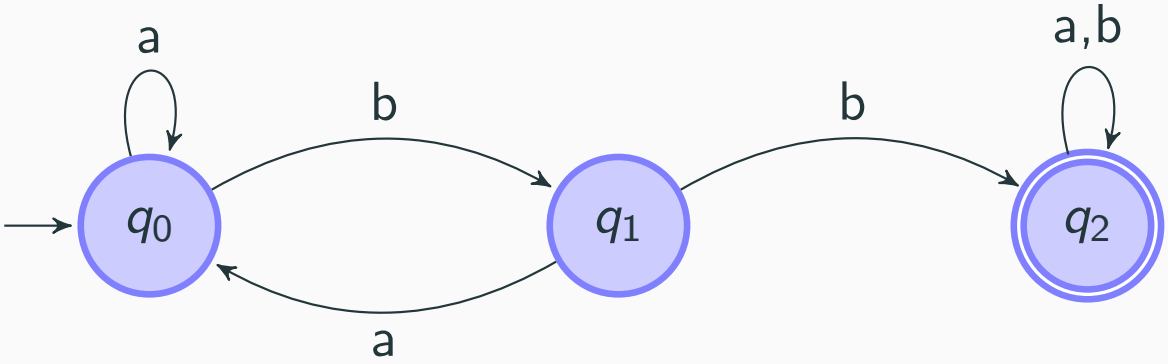
- Automate M_2



- Est-ce que M_2 **reconnait** *abba* ? **accepte** *abba* ?

Exemple

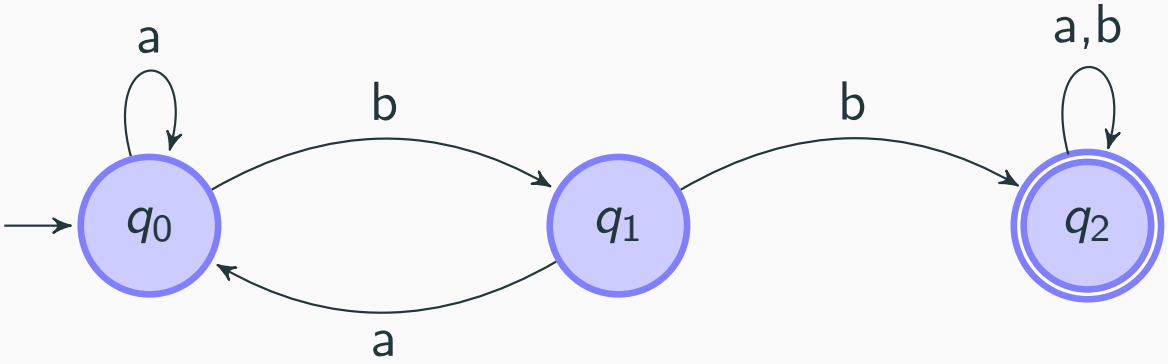
- Automate M_2



- Est-ce que M_2 **reconnait** *abba* ? **accepte** *abba* ?
- Est-ce que M_2 **reconnait** *aaba* ? **accepte** *aaba* ?

Exemple

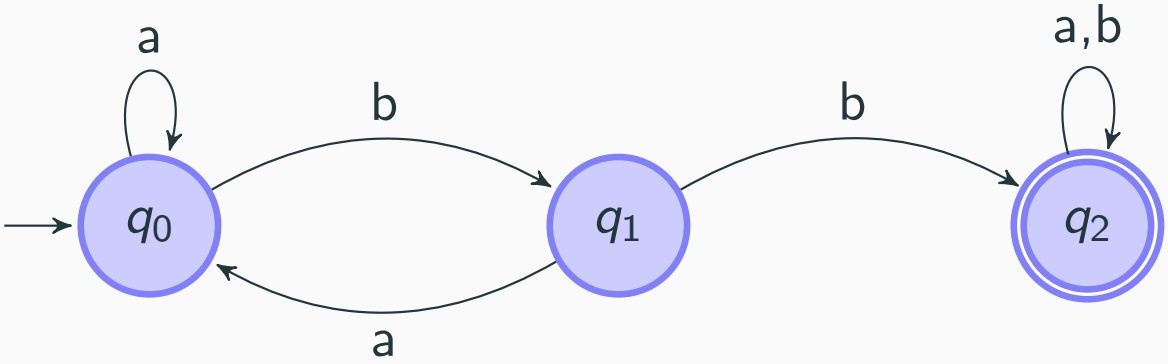
- Automate M_2



- Est-ce que M_2 **reconnait** *abba* ? **accepte** *abba* ?
- Est-ce que M_2 **reconnait** *aaba* ? **accepte** *aaba* ?
- Est-ce que M_2 **reconnait** *abaa* ? **accepte** *abaa* ?

Exemple

- Automate M_2



- Est-ce que M_2 **reconnait** *abba* ? **accepte** *abba* ?
- Est-ce que M_2 **reconnait** *aaba* ? **accepte** *aaba* ?
- Est-ce que M_2 **reconnait** *abaa* ? **accepte** *abaa* ?
- $\mathcal{L}(M_2) = \{w \in \{a, b\}^* | w$ contient deux b consécutifs$\}$

Automate fini non déterministe (AFI)

Automate fini non déterministe

Automate fini non déterministe

Un **automate fini non déterministe (AFI)** est un quintuplet $M = \langle Q, \Sigma, \Delta, S, F \rangle$ où

- Q est un ensemble fini d'*états*
- Σ est un ensemble fini de symboles (un *alphabet*)
- $\Delta \subseteq (Q \times \Sigma \times Q)$ est une *relation de transitions*
- $S \subseteq Q$ est l'ensemble (fini) des *état initiaux*
- $F \subseteq Q$ est l'ensemble (fini) des *états finaux*

Automate fini non déterministe

Automate fini non déterministe

Un **automate fini non déterministe (AFI)** est un quintuplet $M = \langle Q, \Sigma, \Delta, S, F \rangle$ où

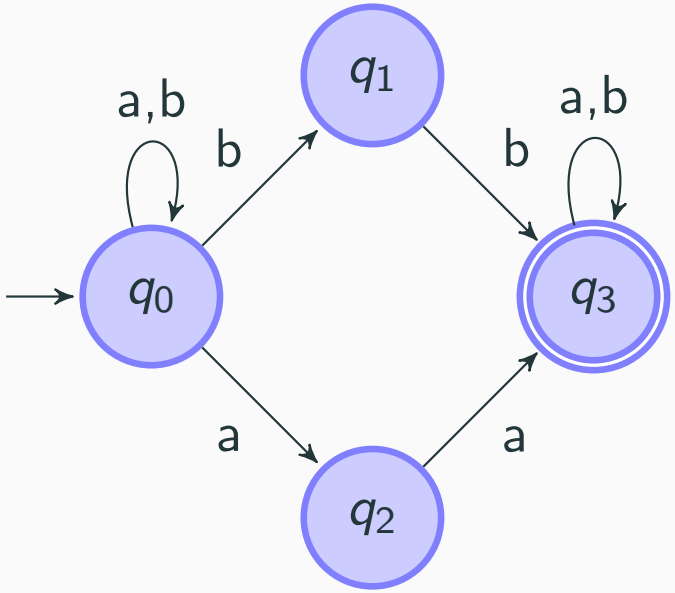
- Q est un ensemble fini d'*états*
- Σ est un ensemble fini de symboles (un *alphabet*)
- $\Delta \subseteq (Q \times \Sigma \times Q)$ est une *relation de transitions*
- $S \subseteq Q$ est l'ensemble (fini) des *état initiaux*
- $F \subseteq Q$ est l'ensemble (fini) des *états finaux*

Différences avec un automate fini déterministe :

- Plusieurs états de départ possible
- Ce n'est plus une fonction de transition, mais une **relation** de transition.

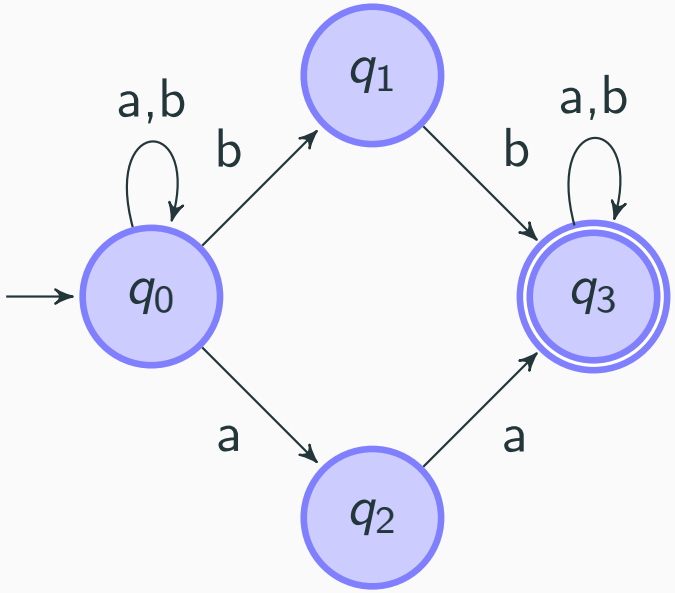
Exemple

- Automate M_3



Exemple

- Automate M_3



- $abb \in \mathcal{L}(M_3)$

Automates complets et émondés

Accessibilité d'un état

Soit un automate fini $M = \langle Q, \Sigma, \Delta, S, F \rangle$:

Etat accessible

Un état $q \in Q$ est accessible s'il existe un chemin de $q_0 \in S$ à q dans M.

Accessibilité d'un état

Soit un automate fini $M = \langle Q, \Sigma, \Delta, S, F \rangle$:

Etat accessible

Un état $q \in Q$ est accessible s'il existe un chemin de $q_0 \in S$ à q dans M.

Etat co-accessible

Un état $q \in Q$ est co-accessible s'il existe un chemin de q jusqu'à un état final dans M.

Accessibilité d'un état

Soit un automate fini $M = \langle Q, \Sigma, \Delta, S, F \rangle$:

Etat accessible

Un état $q \in Q$ est accessible s'il existe un chemin de $q_0 \in S$ à q dans M.

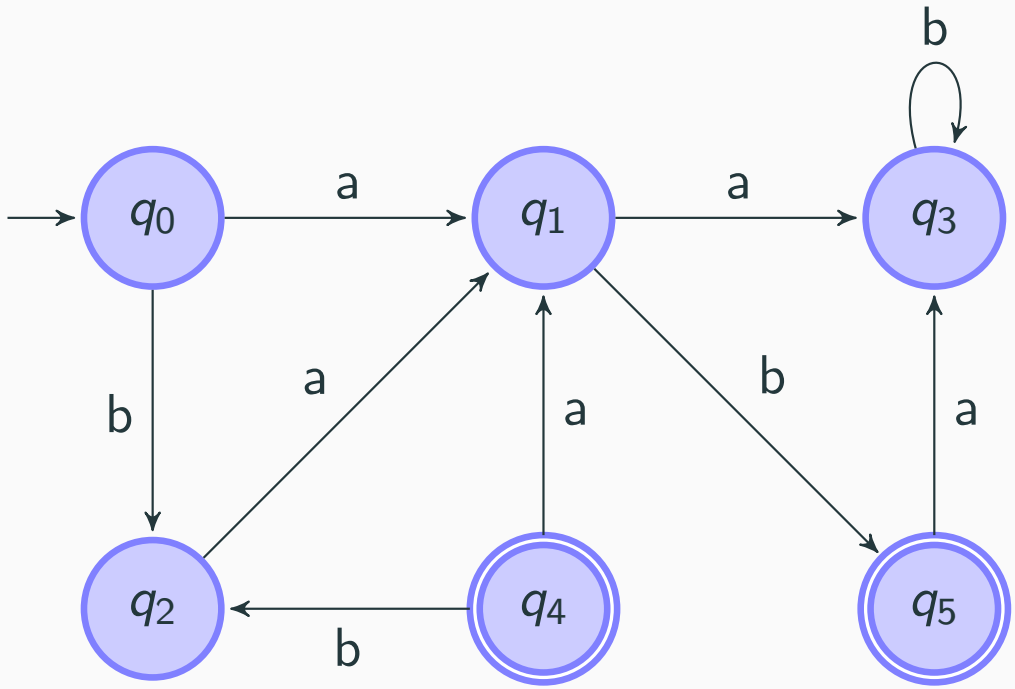
Etat co-accessible

Un état $q \in Q$ est co-accessible s'il existe un chemin de q jusqu'à un état final dans M.

Etat utile

Un état $q \in Q$ est utile s'il est accessible et co-accessible dans M.

Exemple



- Etats accessibles ? $Acc = \{q_0, q_1, q_2, q_3, q_5\}$
- Etats co-accessibles ? $Co-acc = \{q_0, q_1, q_2, q_4, q_5\}$
- Etats utiles ? $U = Acc \cap Co-acc = \{q_0, q_1, q_2, q_5\}$

Automate émondé

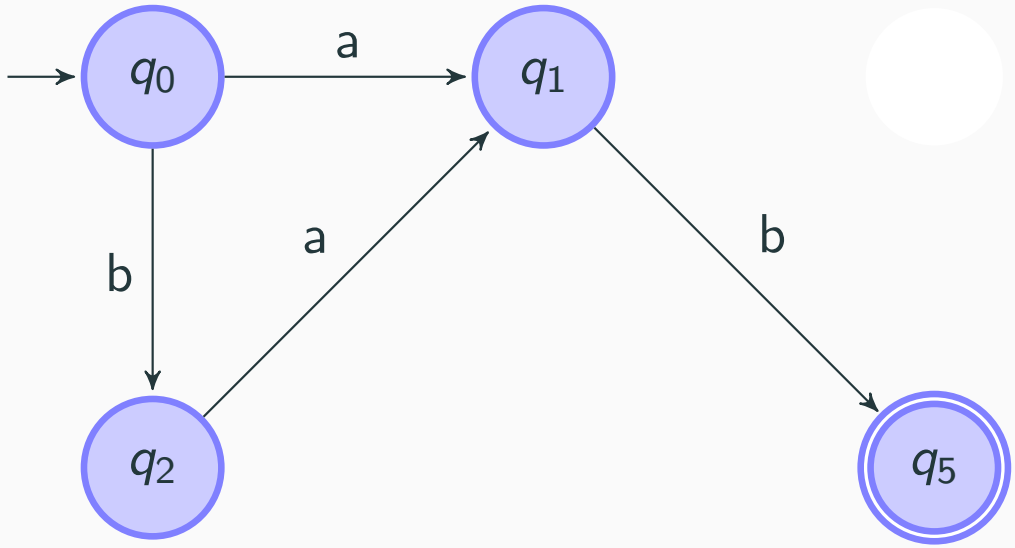
Automate émondé

Un automate M est émondé si et seulement si tous ses états sont utiles.

Automate émondé

Automate émondé

Un automate M est émondé si et seulement si tous ses états sont utiles.



Automate complet

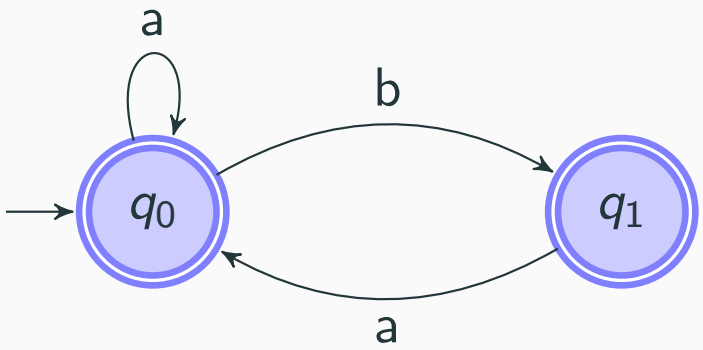
Automate complet

Un automate est complet si pour tout état $q \in Q$ il existe une transition pour chaque lettre de l'alphabet Σ.

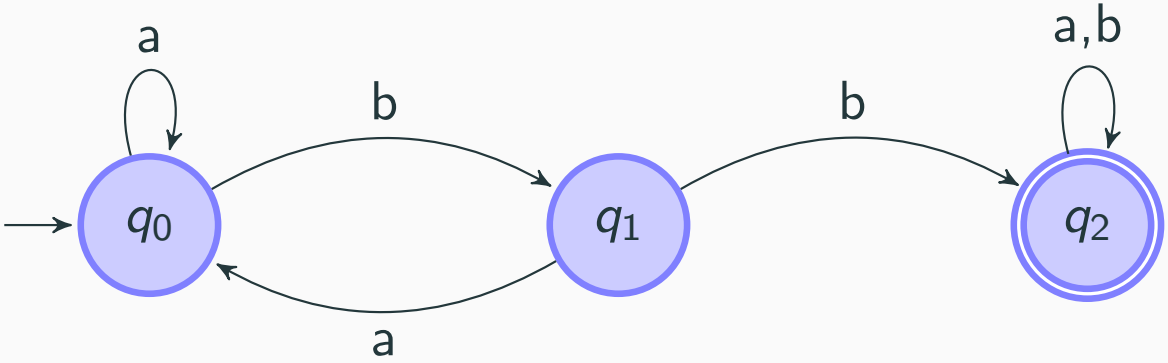
$$\forall q \in Q,\ \forall x \in \Sigma,\ \delta(q, x) \text{ est défini}$$

Exemple

- $\Sigma = \{a, b\}$
- L'automate M_1 est-il complet ?



- L'automate M_2 est-il complet ?



Automates complets

Propriété

Pour un automate complet, la reconnaissance d'un mot ne "bloque" jamais.

$$(q_0, w) \mapsto (q_1, w_1) \mapsto (q_2, w_2) \mapsto \ldots \mapsto (q_n, \epsilon)$$

On a deux possibilités

- Soit $q_n \in F$, et w est un mot accepté
- Soit $q_n \notin F$, et w n'est pas un mot accepté

Etats puits et poubelles

Etat puits

Un état puits est un état $q \in Q$ pour lequel toutes les transitions sont de la forme $\delta(q, x) = q$.

Etats puits et poubelles

Etat puits

Un état puits est un état $q \in Q$ pour lequel toutes les transitions sont de la forme $\delta(q, x) = q$.

Etat puits de M_2 ?

Etats puits et poubelles

Etat puits

Un état puits est un état $q \in Q$ pour lequel toutes les transitions sont de la forme $\delta(q, x) = q$.

Etat puits de M_2 ?

Etat poubelle

Un état poubelle est un état puits non final.

Etats puits et poubelles

Etat puits

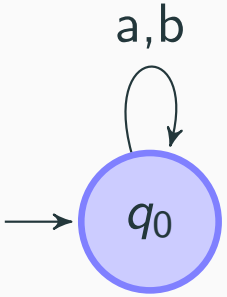
Un état puits est un état $q \in Q$ pour lequel toutes les transitions sont de la forme $\delta(q, x) = q$.

Etat puits de M_2 ?

Etat poubelle

Un état poubelle est un état puits non final.

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



Automates équivalents

Automates équivalents

Deux automates M et M' sont équivalents si et seulement si $\mathcal{L}(M) = \mathcal{L}(M')$.

Automates équivalents

Automates équivalents

Deux automates M et M' sont équivalents si et seulement si $\mathcal{L}(M) = \mathcal{L}(M')$.

Propriété

Pour tout automate fini, il existe un automate fini complet équivalent

Automates équivalents

Automates équivalents

Deux automates M et M' sont équivalents si et seulement si $\mathcal{L}(M) = \mathcal{L}(M')$.

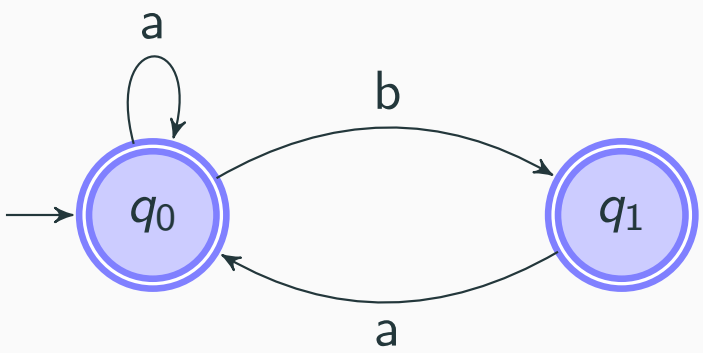
Propriété

Pour tout automate fini, il existe un automate fini complet équivalent

Si l'automate n'est pas complet, on le complète en ajoutant un état poubelle

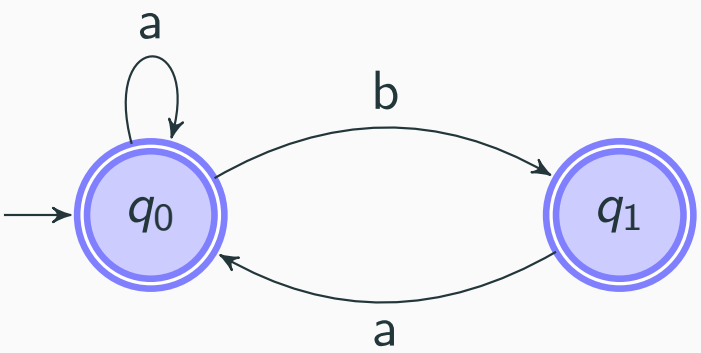
Exemple

- Automate M_1.

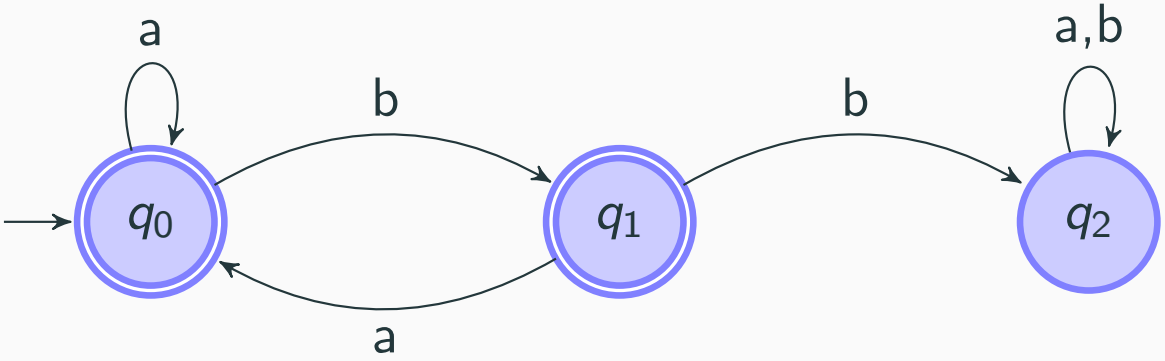


Exemple

- Automate M_1.



- Automate M_1 complété



Langage reconnu par un automate

Langage reconnu à partir d'un état par un automate

Langage reconnu à partir d'un état par un automate

Le **langage reconnu à partir d'un état** q par un automate M, noté $L(q)$ est l'ensemble des mots acceptés à partir de cet état.

$$L(q) = L_q = \{w \in \Sigma^* | (q, w) \overset{*}{\mapsto} (q', \epsilon) \text{ et } q' \in F\}$$

Langage reconnu à partir d'un état par un automate

Langage reconnu à partir d'un état par un automate

Le **langage reconnu à partir d'un état** q par un automate M, noté $L(q)$ est l'ensemble des mots acceptés à partir de cet état.

$$L(q) = L_q = \{w \in \Sigma^* | (q, w) \overset{*}{\mapsto} (q', \epsilon) \text{ et } q' \in F\}$$

Remarque : Soit q un état poubelle. $L(q) = \emptyset$.

Langage reconnu par un automate

Langage reconnu par un automate

Le langage reconnu par un automate M est défini par

$$\mathcal{L}(M) = L(q_0)$$

Système d'équations définissant un langage

Equation définissant un langage généré à partir d'un état

Le langage reconnu à partir d'un état q par un automate M est défini par une équation de la forme :

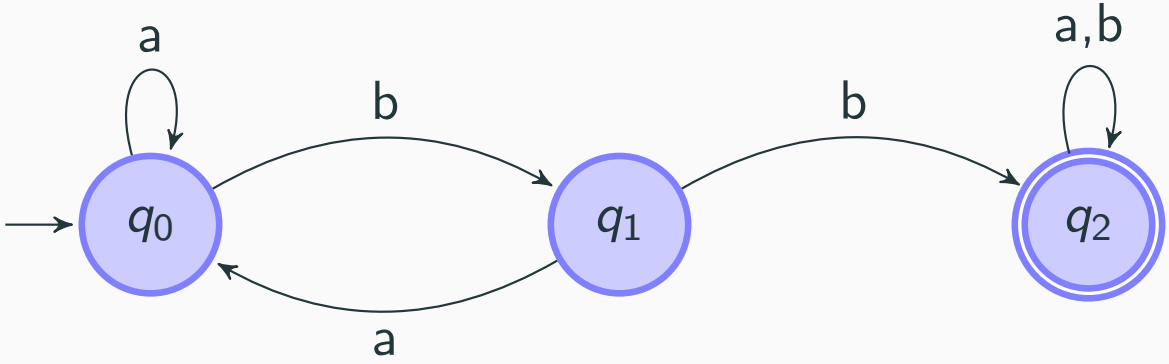
$$L(q) = (\bigcup_{x \in \Sigma} x.L(\delta(q,x))) \cup d(L(q))$$

$$\text{où } d(A) = \begin{cases} \emptyset & \text{si A n'est pas un état final} \\ \{\epsilon\} & \text{si A est un état final} \end{cases}$$

On pourra également noter :

$$L_q = (\sum_{x \in \Sigma} x.L(\delta(q,x))) + d(L_q)$$

Exemple



$$\begin{cases} L(q_0) &= aL(q_0) \cup bL(q_1) \cup \emptyset \\ L(q_1) &= aL(q_0) \cup bL(q_2) \cup \emptyset \\ L(q_2) &= aL(q_2) \cup bL(q_2) \cup \{\epsilon\} \end{cases}$$

OU

$$\begin{cases} L_0 &= aL_0 + bL_1 \\ L_1 &= aL_0 + bL_2 \\ L_2 &= aL_2 + bL_2 + \epsilon \end{cases}$$

Lien entre AFD et AFI

Equivalence entre AFD et AFI

Théorème

La famille des langages acceptés par un automate fini déterministe (AFD) est identique à la famille des langages acceptés par un automate fini non déterministe (AFI).

Equivalence entre AFD et AFI

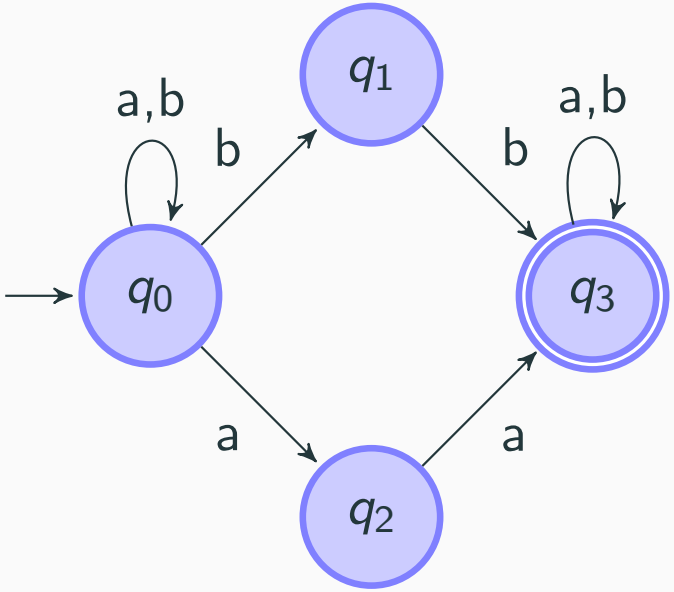
Théorème

La famille des langages acceptés par un automate fini déterministe (AFD) est identique à la famille des langages acceptés par un automate fini non déterministe (AFI).

Propriété

Pour tout automate fini non déterministe, il existe un automate fini déterministe équivalent.

Exemple : automate M_3



Déterminisons M_3

$$\begin{cases} L_0 &= aL_0 + bL_0 + aL_2 + bL_1 \\ L_1 &= bL_3 \\ L_2 &= aL_3 \\ L_3 &= aL_3 + bL_3 + \epsilon \end{cases}$$

Exemple : automate M_3

Déterminisons M_3

$$\begin{cases} L_0 &= aL_0 + bL_0 + aL_2 + bL_1 \\ L_1 &= bL_3 \\ L_2 &= aL_3 \\ L_3 &= aL_3 + bL_3 + \epsilon \end{cases}$$

On sait que $\mathcal{L}(M_3) = L_0$. On a donc

$$\begin{aligned} L_0 &= a(L_0 + L_2) + b(L_0 + L_1) \\ L_0 + L_2 &= a(L_0 + L_2 + L_3) + b(L_0 + L_1) \\ L_0 + L_1 &= a(L_0 + L_2) + b(L_0 + L_1 + L_3) \\ L_0 + L_2 + L_3 &= a(L_0 + L_2 + L_3) + b(L_0 + L_1 + L_3) + \epsilon \\ L_0 + L_1 + L_3 &= a(L_0 + L_2 + L_3) + b(L_0 + L_1 + L_3) + \epsilon \\ L_0 + L_1 + L_3 &= L_0 + L_2 + L_3 \end{aligned}$$

Exemple : automate M_3

$$\begin{cases} L_0 & = & a(L_0 + L_2) + b(L_0 + L_1) \\ L_0 + L_2 & = & a(L_0 + L_2 + L_3) + b(L_0 + L_1) \\ L_0 + L_1 & = & a(L_0 + L_2) + b(L_0 + L_2 + L_3) \\ L_0 + L_2 + L_3 & = & a(L_0 + L_2 + L_3) + b(L_0 + L_2 + L_3) + \epsilon \end{cases}$$

On obtient l'automate suivant :