Chapitre 5 : Les Automates

Option Informatique – MP

Plan

Les automates finis

Les automates déterministes

Introduction

On a défini des langages mais on aimerait avoir un moyen simple de déterminer si un mot appartient à langage donné (au moins dans le cas où le langage est un langage rationnel).
Nous allons voir la notion d'automates finis qui permet de faire cela. Le principe général est le suivant :

Introduction

On a défini des langages mais on aimerait avoir un moyen simple de déterminer si un mot appartient à langage donné (au moins dans le cas où le langage est un langage rationnel).
Nous allons voir la notion d'automates finis qui permet de faire cela. Le principe général est le suivant :

- Un automate possède un ou plusieurs états

Introduction

On a défini des langages mais on aimerait avoir un moyen simple de déterminer si un mot appartient à langage donné (au moins dans le cas où le langage est un langage rationnel).
Nous allons voir la notion d'automates finis qui permet de faire cela. Le principe général est le suivant :

- Un automate possède un ou plusieurs états
- Chaque lettre de l'alphabet engendre une transition qui permet d'aller d'un état donné à un autre état (ou des autres états voir plus loin)

Introduction

On a défini des langages mais on aimerait avoir un moyen simple de déterminer si un mot appartient à langage donné (au moins dans le cas où le langage est un langage rationnel).
Nous allons voir la notion d'automates finis qui permet de faire cela. Le principe général est le suivant :

- Un automate possède un ou plusieurs états
- Chaque lettre de l'alphabet engendre une transition qui permet d'aller d'un état donné à un autre état (ou des autres états voir plus loin)
- Il y a des états initiaux et des états finaux.

Automates finis déterministes

Définition (Automate fini déterministe)

Un automate (fini) déterministe ou AFD sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, q_0, F, \delta)$ où

Automates finis déterministes

Définition (Automate fini déterministe)

Un automate (fini) déterministe ou AFD sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, q_0, F, \delta)$ où

- L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate

Automates finis déterministes

Définition (Automate fini déterministe)

Un automate (fini) déterministe ou AFD sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, q_0, F, \delta)$ où

- L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate
- L'élément q_0 est un élément de Q. Il s'appelle l'état initial

Automates finis déterministes

Définition (Automate fini déterministe)

Un automate (fini) déterministe ou AFD sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, q_0, F, \delta)$ où

- L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate
- L'élément q_0 est un élément de Q. Il s'appelle l'état initial
- L'ensemble F est une partie de Q. Ses éléments s'appellent les états finaux (ou acceptants)

Automates finis déterministes

Définition (Automate fini déterministe)

Un automate (fini) déterministe ou AFD sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, q_0, F, \delta)$ où

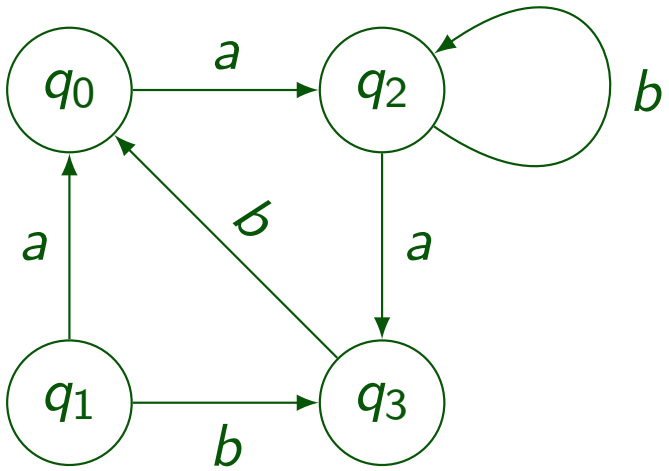
- L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate
- L'élément q_0 est un élément de Q. Il s'appelle l'état initial
- L'ensemble F est une partie de Q. Ses éléments s'appellent les états finaux (ou acceptants)
- L'application δ est définie sur une partie de $Q \times \mathscr{A}$ et à valeurs dans Q. C'est la fonction de transition.

Automates finis déterministes

On veut définir un automate sur $\mathscr{A} = \{a, b\}$. On se donne $Q = \{q_0, q_1, q_2, q_3\}$ avec q_0 l'état initial, $F = \{q_2\}$ et δ donnée par

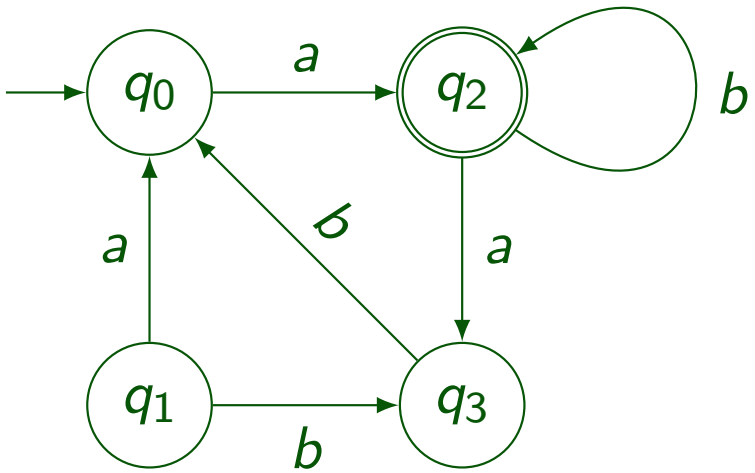
	q_0	q_1	q_2	q_3
a	q_2	q_0	q_3	
b		q_3	q_2	q_0

Afin de mieux visualiser l'automate on représente les états et les transitions ainsi :



Automates finis déterministes

On indique l'état initial par une flèche entrante
On désigne les états acceptants par un double cercle. On trouve aussi une flèche sortante vers rien.



Exemple

On peut par exemple penser au monnayeur d'une machine :

Exemple

On peut par exemple penser au monnayeur d'une machine :

- Les états désignent alors le montant inséré.

Exemple

On peut par exemple penser au monnayeur d'une machine :

- Les états désignent alors le montant inséré.
- L'état initial est l'état qui représente la somme nulle.

Exemple

On peut par exemple penser au monnayeur d'une machine :

- Les états désignent alors le montant inséré.
- L'état initial est l'état qui représente la somme nulle.
- Les transitions sont calculées en ajoutant à chaque étape l'argent ajouté à la somme déjà dans la machine.

Exemple

On peut par exemple penser au monnayeur d'une machine :

- Les états désignent alors le montant inséré.
- L'état initial est l'état qui représente la somme nulle.
- Les transitions sont calculées en ajoutant à chaque étape l'argent ajouté à la somme déjà dans la machine.
- Les états finaux sont ceux où la somme est supérieure à la somme voulue par la machine pour délivrer le bien.

Terminologie

Soit (Q, q_0, F, δ) un automate fini déterministe :

Terminologie

Soit (Q, q_0, F, δ) un automate fini déterministe :

- Un blocage de l'automate est un couple $(q, x) \in Q \times \mathscr{A}$ tel que $\delta(q, x)$ n'est pas défini.

Terminologie

Soit (Q, q_0, F, δ) un automate fini déterministe :

- Un blocage de l'automate est un couple $(q, x) \in Q \times \mathscr{A}$ tel que $\delta(q, x)$ n'est pas défini.
- L'automate sans blocages est dit complet.

Terminologie

Soit (Q, q_0, F, δ) un automate fini déterministe :

- Un blocage de l'automate est un couple $(q, x) \in Q \times \mathscr{A}$ tel que $\delta(q, x)$ n'est pas défini.
- L'automate sans blocages est dit complet.
- L'automate est dit standard si pour tout $(q, x) \in Q \times \mathscr{A}$, $\delta(q, x) \neq q_0$. C'est-à-dire que l'automate ne « retombe » pas dans l'état initial.

Terminologie

Soit (Q, q_0, F, δ) un automate fini déterministe :

- Un blocage de l'automate est un couple $(q, x) \in Q \times \mathscr{A}$ tel que $\delta(q, x)$ n'est pas défini.
- L'automate sans blocages est dit complet.
- L'automate est dit standard si pour tout $(q, x) \in Q \times \mathscr{A}$, $\delta(q, x) \neq q_0$. C'est-à-dire que l'automate ne « retombe » pas dans l'état initial. Nous verrons plus loin l'intérêt des automates standards.

Calcul dans un automate

Nous allons maintenant « calculer » dans l'automate. Cela consiste à lire un mot de $\mathscr{A}^\star$ en parcourant en conséquence les différents états de l'automates.
Pour cela on va étendre la fonction de transition à des mots de $\mathscr{A}^\star$ en faisant évoluer l'état de l'automate à chaque lettre du mot.

Calcul dans un automate

Définition

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe sur $\mathscr{A}$ on définit une application de transition notée $\delta^\star : Q \times \mathscr{A}^\star \to Q$ par :

- $\forall q \in Q, \delta^\star(q, \varepsilon) = q$

Calcul dans un automate

Définition

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe sur $\mathscr{A}$ on définit une application de transition notée $\delta^\star : Q \times \mathscr{A}^\star \to Q$ par :

- $\forall q \in Q, \delta^\star(q, \varepsilon) = q$
- $\forall a \in \mathscr{A}, \forall w \in \mathscr{A}^\star, \delta(q, aw) = \delta^\star(\delta(q, a), w)$

Remarques

1. Il arrive que l'on note aussi simplement δ pour $\delta^\star$.

Remarques

1. Il arrive que l'on note aussi simplement δ pour $\delta^\star$.
2. Si l'automate est complet, la fonction $\delta^\star$ est définie pour tout $q \in Q$ et tout $w \in \mathscr{A}^\star$.

Remarques

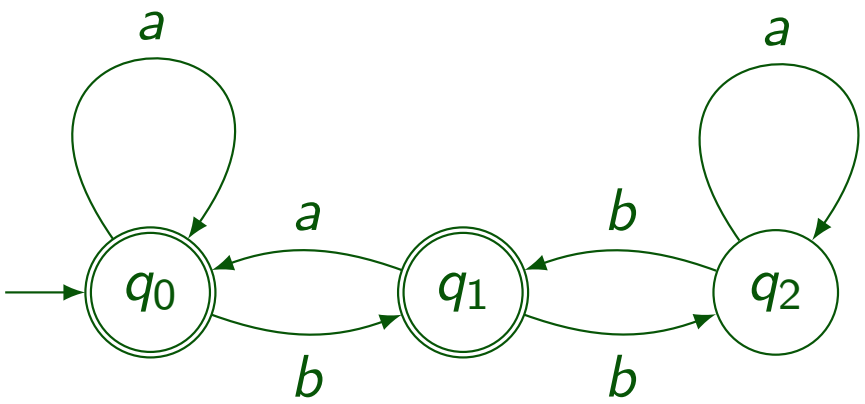
1. Il arrive que l'on note aussi simplement δ pour $\delta^\star$.
2. Si l'automate est complet, la fonction $\delta^\star$ est définie pour tout $q \in Q$ et tout $w \in \mathscr{A}^\star$.
3. Pour $w \in \mathscr{A}^\star$, on appelle donc calcul dans l'automate sur le mot $w = w_0 w_1 \cdots w_{n-1}$ issu de l'état q un « chemin » dans l'automate :

$$q = \alpha_0 \xrightarrow{w_0} \alpha_1 \xrightarrow{w_1} \cdots \xrightarrow{w_{n-1}} \alpha_n$$

où pour tout $i \in [\![0 ; n-1]\!]$, $\delta(\alpha_i, w_i) = w_{i+1}$. On a alors $\delta^\star(q, w) = \alpha_n$.

Exemple

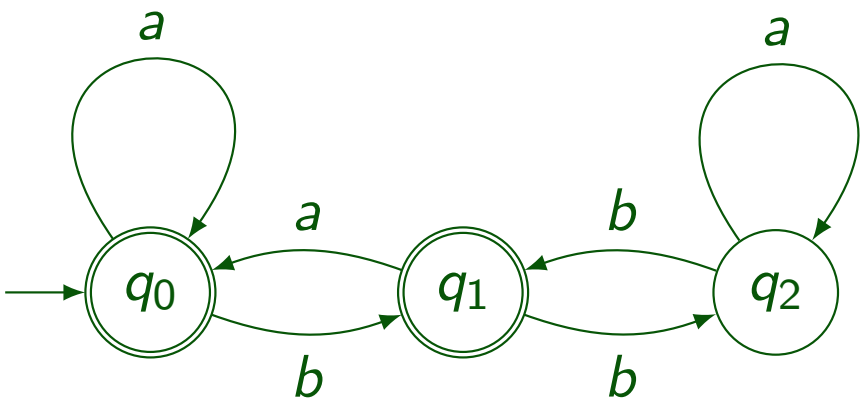
Considérons l'automate complet suivant



- $\delta^\star(q_0, abba) = q_2$
- $\delta^\star(q_1, aabb) =$

Exemple

Considérons l'automate complet suivant



- $\delta^\star(q_0, abba) = q_2$
- $\delta^\star(q_1, aabb) = q_2$

Langage reconnu par un automate

Définition (Langage reconnu par un automate)

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe.

Langage reconnu par un automate

Définition (Langage reconnu par un automate)

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe.

1. Un mot w sur $\mathscr{A}$ est dit reconnu par l'automate si $\delta^\star(q_0, w) \in F$.

Langage reconnu par un automate

Définition (Langage reconnu par un automate)

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe.

1. Un mot w sur $\mathscr{A}$ est dit reconnu par l'automate si $\delta^\star(q_0, w) \in F$.
2. L'ensemble des mots reconnus par l'automate A se note $L(A)$. C'est le langage reconnu par l'automate.

Remarques

- Le but d'un automate est de reconnaitre des mots.

Remarques

- Le but d'un automate est de reconnaitre des mots.
- On dit aussi que les mots reconnus sont les mots acceptés.

Remarques

- Le but d'un automate est de reconnaitre des mots.
- On dit aussi que les mots reconnus sont les mots acceptés.
- Un mot $w \in \mathscr{A}^\star$ **n'est pas reconnu** par A si

Remarques

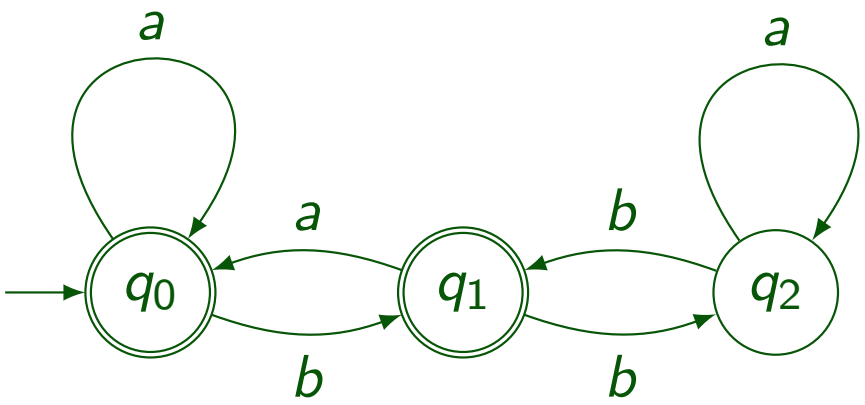
- Le but d'un automate est de reconnaitre des mots.
- On dit aussi que les mots reconnus sont les mots acceptés.
- Un mot $w \in \mathscr{A}^\star$ **n'est pas reconnu** par A si
  - La lecture du mot par l'automate engendre un blocage, c'est-à-dire qu'il existe un préfixe u tel que $w = uav$ et $(\delta^*(q_0, u), a)$ est un blocage de A.

Remarques

- Le but d'un automate est de reconnaitre des mots.
- On dit aussi que les mots reconnus sont les mots acceptés.
- Un mot $w \in \mathscr{A}^\star$ **n'est pas reconnu** par A si
  - La lecture du mot par l'automate engendre un blocage, c'est-à-dire qu'il existe un préfixe u tel que $w = uav$ et $(\delta^*(q_0, u), a)$ est un blocage de A.
  - La lecture du mot peut se faire mais $\delta^\star(q_0, w) \notin F$.

Exemple

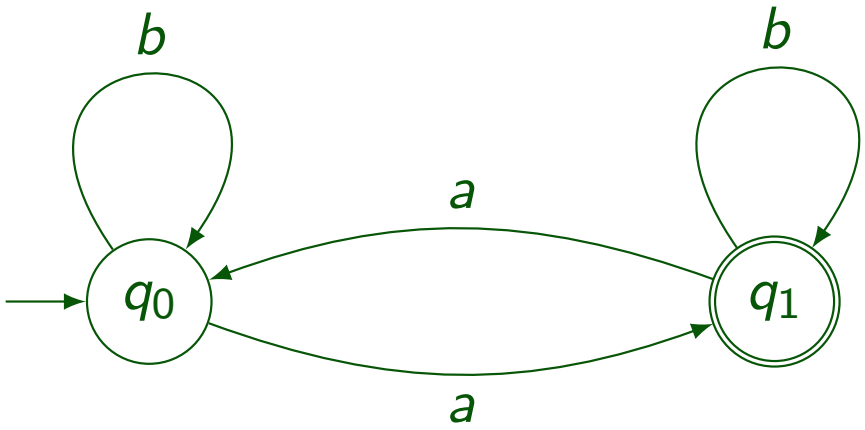
Si on reprend l'automate précédent



- On voit que a, b, aab sont reconnus
- Le mot $abba$ n'est pas reconnu.

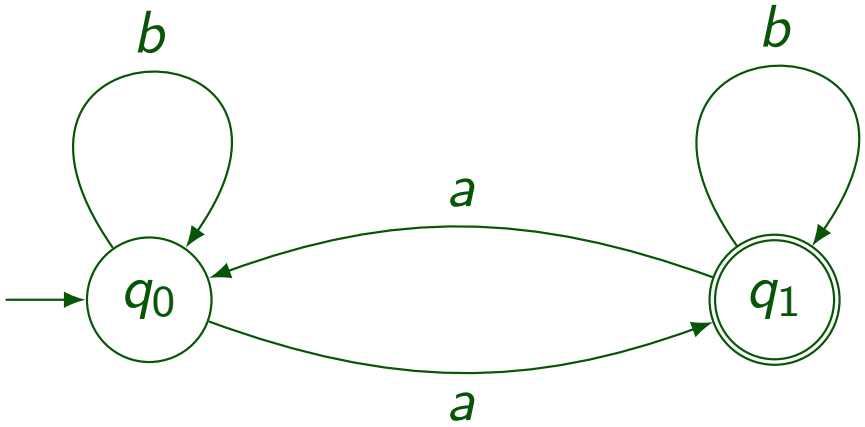
Exercice 1

Soit A l'automate

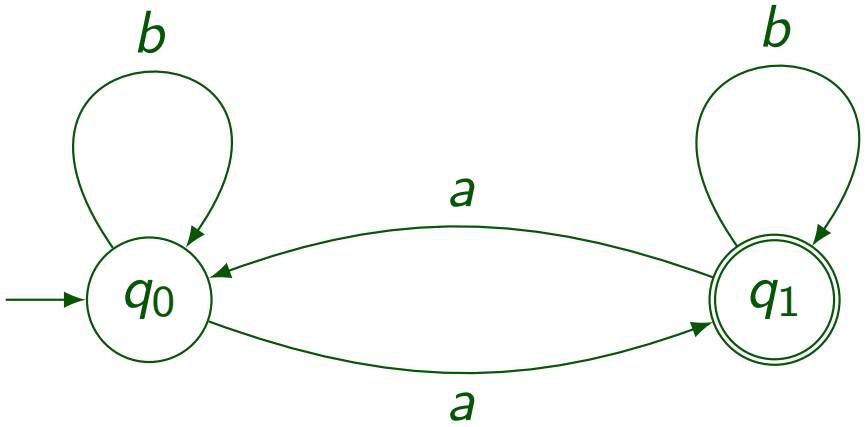


Déterminer des mots reconnus par A et des mots qui ne sont pas reconnus.
Quel est le langage reconnu par l'automate ?

Exercice 1

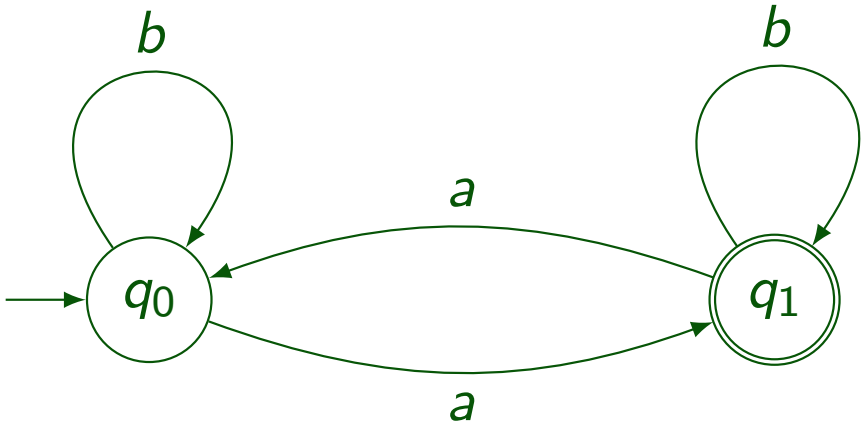


Exercice 1



- Le mot a est reconnu alors que aa ne l'est pas

Exercice 1



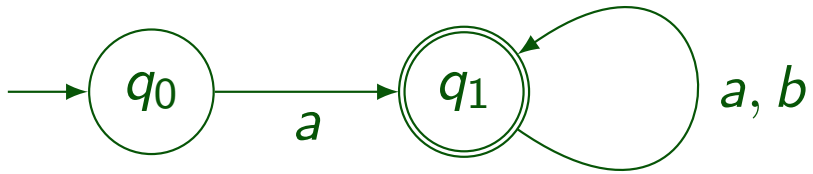
- Le mot a est reconnu alors que aa ne l'est pas
- L'automate reconnait les mots qui comportent un nombre impair de a. De fait, l'état q_0 correspond à « on a lu un nombre pair de a » et q_1 correspond à « on a lu un nombre pair de a ».

Exercice 2

Déterminer un automate dont le langage reconnu est $a\mathscr{A}^\star$.

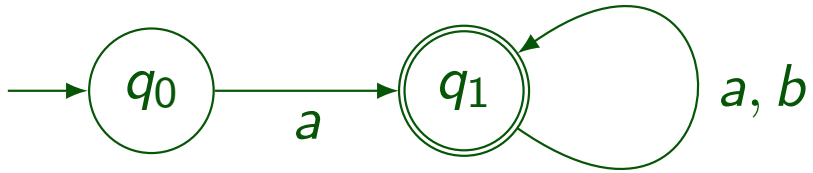
Exercice 2

Déterminer un automate dont le langage reconnu est $a\mathscr{A}^\star$.

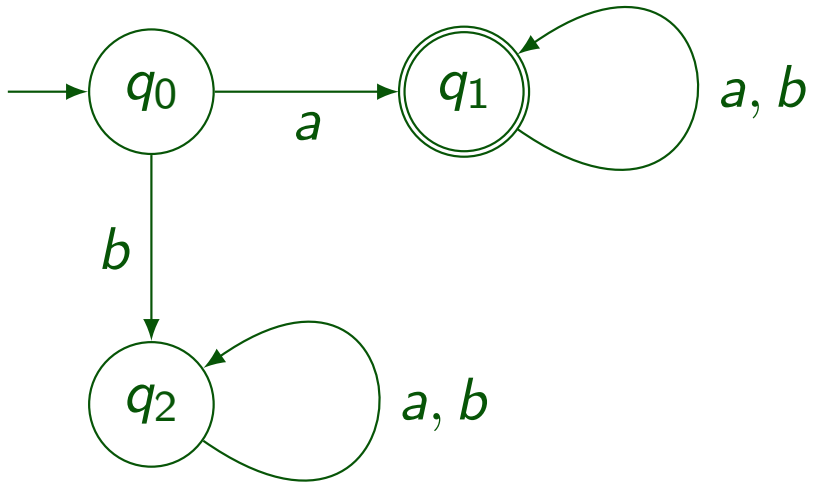


Exercice 2

Déterminer un automate dont le langage reconnu est $a\mathscr{A}^\star$.



ou



Complétion d'un automate

Théorème

Tout langage reconnu par un automate est reconnu par un automate complet

Complétion d'un automate

Théorème

Tout langage reconnu par un automate est reconnu par un automate complet

Démonstration : Soit L un langage et $A = (Q, q_0, \delta, F)$ un automate qui le reconnait. On peut construire un automate complet A' tel que $L(A') = L(A) = L$. Il suffit d'ajouter à Q un nouvel état p que l'on appelle un état puit. On pose $A' = (Q \cup \{p\}, q_0, \delta', F)$ où δ' est définie par :

Complétion d'un automate

Théorème

Tout langage reconnu par un automate est reconnu par un automate complet

Démonstration : Soit L un langage et $A = (Q, q_0, \delta, F)$ un automate qui le reconnait. On peut construire un automate complet A' tel que $L(A') = L(A) = L$. Il suffit d'ajouter à Q un nouvel état p que l'on appelle un état puit. On pose $A' = (Q \cup \{p\}, q_0, \delta', F)$ où δ' est définie par :

- Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) n'est pas un blocage $\delta'(q, x) = \delta(q, x)$.

Complétion d'un automate

Théorème

Tout langage reconnu par un automate est reconnu par un automate complet

Démonstration : Soit L un langage et $A = (Q, q_0, \delta, F)$ un automate qui le reconnait. On peut construire un automate complet A' tel que $L(A') = L(A) = L$. Il suffit d'ajouter à Q un nouvel état p que l'on appelle un état puit. On pose $A' = (Q \cup \{p\}, q_0, \delta', F)$ où δ' est définie par :

- Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) n'est pas un blocage $\delta'(q, x) = \delta(q, x)$.
- Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) est un blocage $\delta'(q, x) = p$.

Complétion d'un automate

Théorème

Tout langage reconnu par un automate est reconnu par un automate complet

Démonstration : Soit L un langage et $A = (Q, q_0, \delta, F)$ un automate qui le reconnait. On peut construire un automate complet A' tel que $L(A') = L(A) = L$. Il suffit d'ajouter à Q un nouvel état p que l'on appelle un état puit. On pose $A' = (Q \cup \{p\}, q_0, \delta', F)$ où δ' est définie par :

- Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) n'est pas un blocage $\delta'(q, x) = \delta(q, x)$.
- Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) est un blocage $\delta'(q, x) = p$.
- Pour $x \in \mathscr{A}$, on pose $\delta'(p, x) = p$.

Complétion d'un automate

Prouvons alors proprement que $L(A) = L(A')$.

- Soit $w = w_0 w_1 \dots w_n \in L(A)$. Il existe un calcul dans A

$$q_0 \xrightarrow{w_0} \cdots \xrightarrow{w_n} q$$

où $q \in F$. Ce calcul est aussi possible dans A' donc $w \in L(A')$. Finalement $\boxed{L(A) \subset L(A')}$.

- Soit $w \notin L(A)$. Il y a deux cas :
  - Le calcul dans A est impossible à cause d'un blocage. Dans ce cas, $(\delta')^\star(q_0, w) = p \notin F$.
  - Le calcul est possible dans A mais $\delta(q_0, w) \notin F$. Dans ce cas, le calcul est identique dans A' et donc on a encore $(\delta')^\star(q_0, w) \notin F$.

  Dans les deux cas $w \notin L(A')$. Finalement, par contraposée, $\boxed{L(A') \subset L(A)}$.

On a bien construit A' complet qui reconnait L.

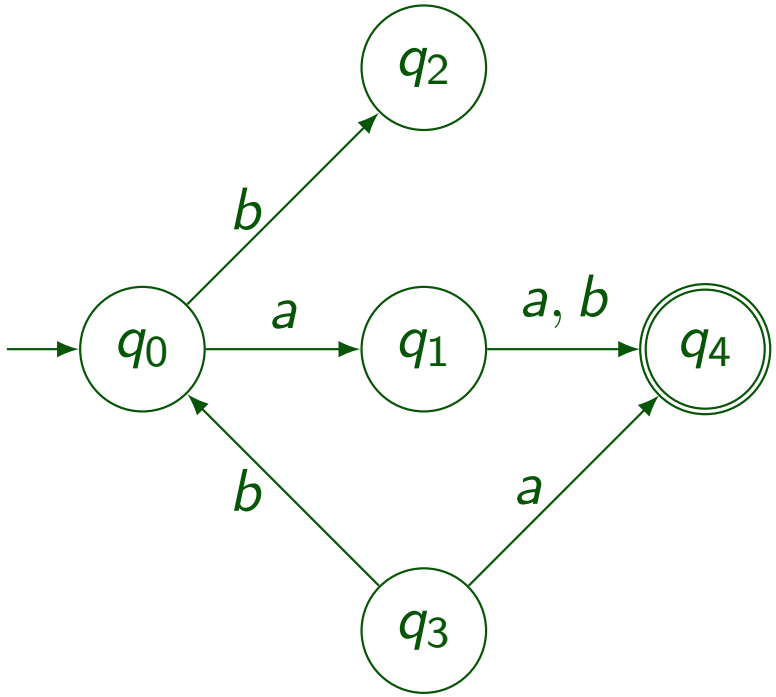
Emondage

Etats accessibles et co-accessibles

Quand on considère un automate A qui reconnait un langage L, on ne s'intéresse qu'aux états que l'on va « atteindre » à partir de l'état initial lors d'un calcul.

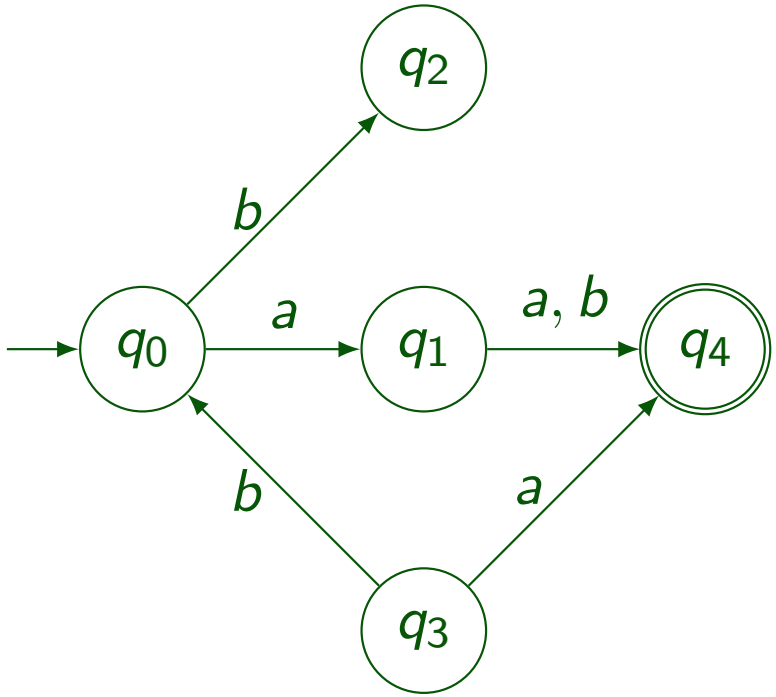
Etats accessibles et co-accessibles

Par exemple dans l'automate



Etats accessibles et co-accessibles

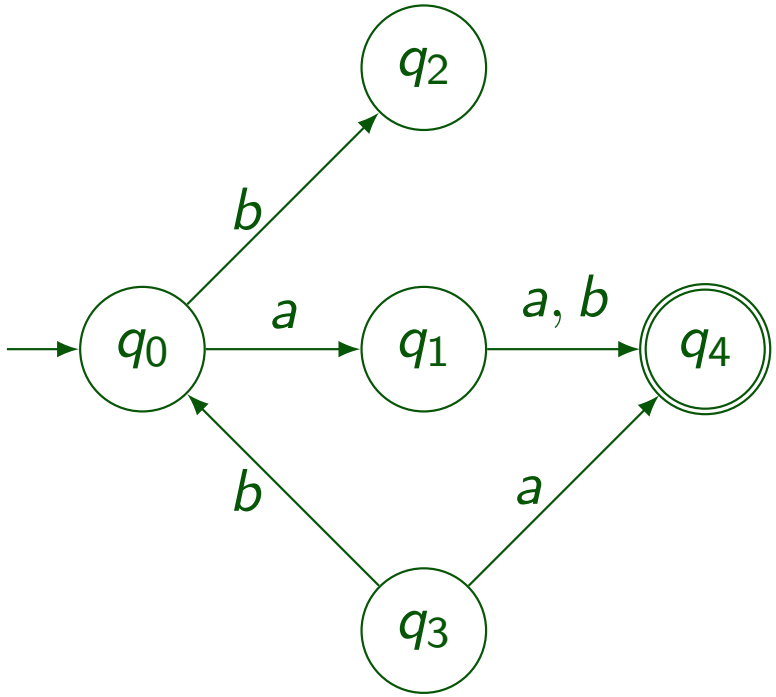
Par exemple dans l'automate



- On ne peut pas atteindre l'état q_3 en partant de q_0

Etats accessibles et co-accessibles

Par exemple dans l'automate



- On ne peut pas atteindre l'état q_3 en partant de q_0
- Si on est à q_2 on ne peut plus rejoindre l'état final q_4.

Etats accessibles et co-accessibles

Définition

Soit $A = (Q, q_0, F, \delta)$ un automate fini deterministe.

Etats accessibles et co-accessibles

Définition

Soit $A = (Q, q_0, F, \delta)$ un automate fini deterministe.

- Un état $q \in Q$ est dit **accessible** s'il existe un mot $w \in \mathscr{A}^\star$ tel que $\delta(q_0, w) = q$. Autrement dit, si on peut atteindre le sommet q en partant du sommet initial.

Etats accessibles et co-accessibles

Définition

Soit $A = (Q, q_0, F, \delta)$ un automate fini deterministe.

- Un état $q \in Q$ est dit **accessible** s'il existe un mot $w \in \mathscr{A}^\star$ tel que $\delta(q_0, w) = q$. Autrement dit, si on peut atteindre le sommet q en partant du sommet initial.
- Un état $q \in Q$ est dit **co-accesible** s'il existe un mot $w \in \mathscr{A}^\star$ tel que $\delta(q, w) \in F$. Autrement dit, si on peut atteindre un sommet acceptant en partant du sommet q.

L'automate est dit émondé si tous ses états sont accessibles et co-accessibles.

Automate émondé

À tout automate $A = (Q, q_0, F, \delta)$ reconnaissant un langage non vide (c'est-à-dire que q_0 est co-accessible) on lui associe l'automate émondé $A' = (Q', q_0, F \cap Q', \delta')$ où :

Automate émondé

À tout automate $A = (Q, q_0, F, \delta)$ reconnaissant un langage non vide (c'est-à-dire que q_0 est co-accessible) on lui associe l'automate émondé $A' = (Q', q_0, F \cap Q', \delta')$ où :

- Q' est l'ensemble des états accessibles et co-accesibles de A

Automate émondé

À tout automate $A = (Q, q_0, F, \delta)$ reconnaissant un langage non vide (c'est-à-dire que q_0 est co-accessible) on lui associe l'automate émondé $A' = (Q', q_0, F \cap Q', \delta')$ où :

- Q' est l'ensemble des états accessibles et co-accesibles de A
- δ' est telle que pour tout $q \in Q'$ et $a \in \mathscr{A}$,

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{si } \delta(q,a) \text{ existe et appartient à } Q' \\ \text{est un blocage} & \text{sinon.} \end{cases}$$

Automate émondé

Théorème

L'automate A et l'automate A' reconnaissent le même langage.

Automates finis non-déterministes

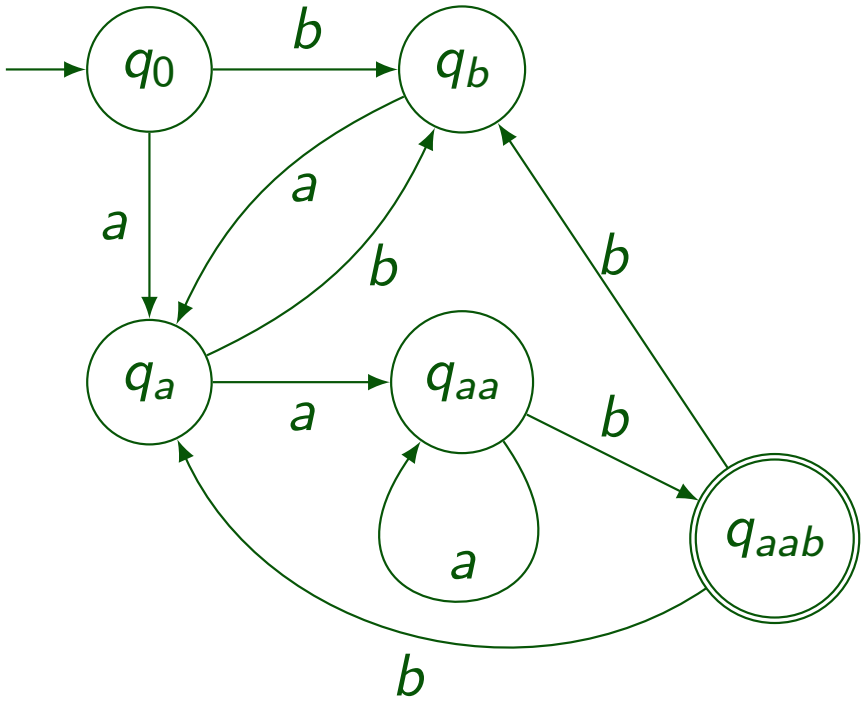
Introduction

Dans certains cas, il va être plus facile de construire des automates d'un type différent. Commençons par un exemple.

On veut construire un automate reconnaissant le langage $L = \mathscr{A}^\star aab$ des mots qui se terminent par aab.

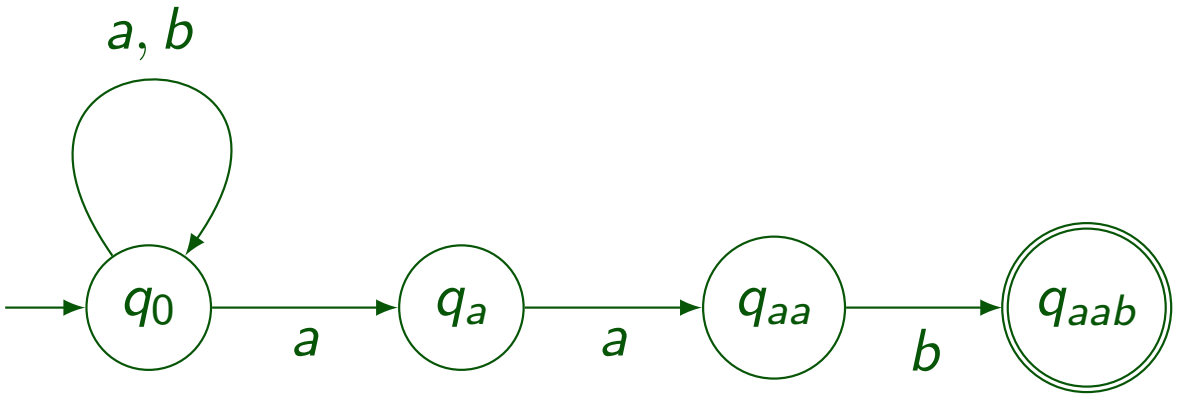
On peut le faire avec un automate déterministe :

Introduction



Introduction

On peut aussi le faire avec ce type d'automate



La différence est que là, en étant à l'état q_0 on peut aller « en lisant » la lettre a en q_0 ou en q_a. C'est en cela que cela va s'appeler un automate **non**-déterministe.

Un calcul d'un mot de $\mathscr{A}^\star aab$ « peut » aboutir à l'état acceptant.

Introduction

Dans les automates que nous avons vus, il y avait une **unique** transition à un état fixé et une lettre fixée. Nous allons regarder maintenant des automates non-déterministes où, d'un état donné et avec une lettre donnée on peut passer à plusieurs états.

Définition

Définition (Automate fini non déterministe)

Un automate (fini) non déterministe ou AFND sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, I, F, \delta)$ où

- L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate.
- L'ensemble I est une partie de Q. Ce sont les états initiaux
- L'ensemble F est une partie de Q. C'est l'ensemble des états finaux (ou acceptants)
- L'application δ est définie sur $Q \times \mathscr{A}$ et à valeurs dans $\mathscr{P}(Q)$. C'est la fonction de transition.

Remarques

- Il y a deux différences (qui ne sont qu'une seule) entre un automate déterministe et un automate non déterministe. La première est que dans ce dernier il peux y avoir plusieurs états initiaux. La deuxième est que, partant d'un état donné, une même lettre nous permet d'atteindre zéro, un ou plusieurs états donnés par une partie de Q.

Remarques

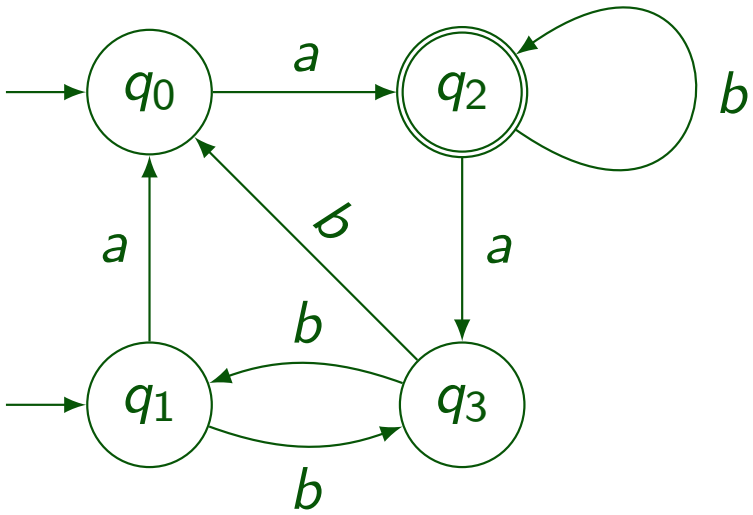
- Il y a deux différences (qui ne sont qu'une seule) entre un automate déterministe et un automate non déterministe. La première est que dans ce dernier il peux y avoir plusieurs états initiaux. La deuxième est que, partant d'un état donné, une même lettre nous permet d'atteindre zéro, un ou plusieurs états donnés par une partie de Q.
- Il n'y a plus de blocage au sens précédent, cependant, $\delta(\{q\}, x)$ peut être vide.

Remarques

- Il y a deux différences (qui ne sont qu'une seule) entre un automate déterministe et un automate non déterministe. La première est que dans ce dernier il peux y avoir plusieurs états initiaux. La deuxième est que, partant d'un état donné, une même lettre nous permet d'atteindre zéro, un ou plusieurs états donnés par une partie de Q.
- Il n'y a plus de blocage au sens précédent, cependant, $\delta(\{q\}, x)$ peut être vide.
- La plupart du temps, un automate (sans précision) désigne un automate **non**-déterministe. **CEPENDANT** il est essentiel de bien lire les notations / définitions dans l'introduction d'un problème sur les automates.

Représentation

On les représente comme précédemment



Langage reconnaissable

Définition

Soit $A = (Q, I, F, \delta)$ un automate (fini) non déterministe.

1. On étend la fonction δ à l'ensemble des mots de $\mathscr{A}^\star$ comme précédemment. Précisément on construit $\delta^\star : Q \times \mathscr{A}^\star \to \mathscr{P}(Q)$ par
   - $\forall q \in Q, \delta^*(q, \varepsilon) = \{q\}$

Langage reconnaissable

Définition

Soit $A = (Q, I, F, \delta)$ un automate (fini) non déterministe.

1. On étend la fonction δ à l'ensemble des mots de $\mathscr{A}^\star$ comme précédemment. Précisément on construit $\delta^\star : Q \times \mathscr{A}^\star \to \mathscr{P}(Q)$ par
   - $\forall q \in Q, \delta^*(q, \varepsilon) = \{q\}$
   - $\forall a \in \mathscr{A}, \forall w \in \mathscr{A}^\star, \delta^\star(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta^\star(q', w).$

Langage reconnaissable

Définition

Soit $A = (Q, I, F, \delta)$ un automate (fini) non déterministe.

1. On étend la fonction δ à l'ensemble des mots de $\mathscr{A}^\star$ comme précédemment. Précisément on construit $\delta^\star : Q \times \mathscr{A}^\star \to \mathscr{P}(Q)$ par
   - $\forall q \in Q, \delta^*(q, \varepsilon) = \{q\}$
   - $\forall a \in \mathscr{A}, \forall w \in \mathscr{A}^\star, \delta^\star(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta^\star(q', w).$
2. Soit w un mot de $\mathscr{A}^\star$, il est reconnu par l'automate s'il existe $q_0 \in I$ tel que $\delta(q_0, w) \cap F \neq \emptyset$.

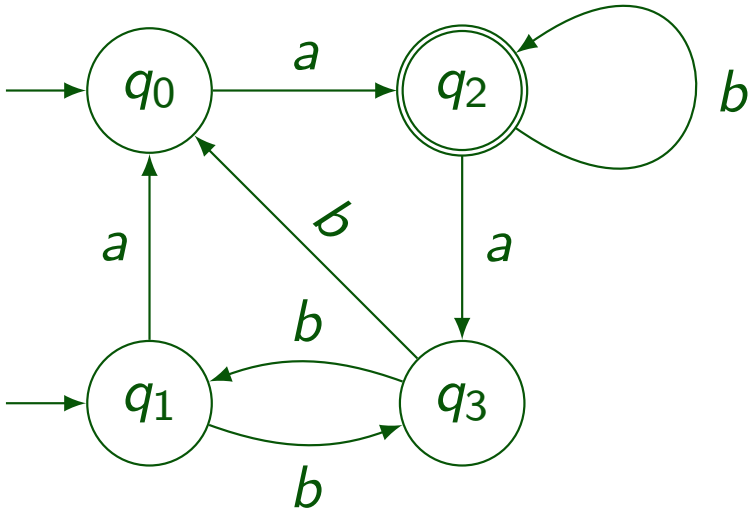
Attention

Attention

Un mot w est donc reconnu par A s'il existe **un** calcul dans A sur w qui aboutit à un état acceptant.

Exemple

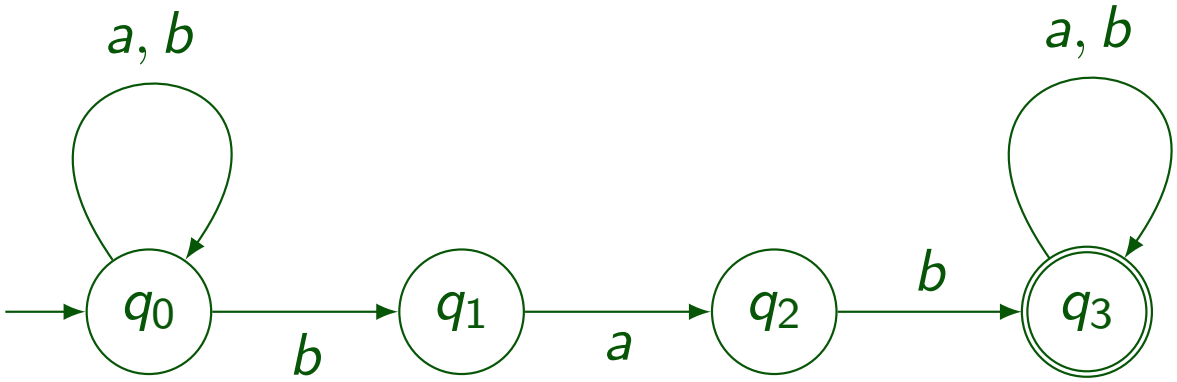
On reprend l'automate précédent



Les mots a, aa, bba sont reconnus. Par contre ba ne l'est pas.

Exercice 1

On considère l'automate suivant :



Déterminer des mots acceptés par l'automate et des mots qui ne sont pas acceptés. Quel est le langage des mots reconnus pas l'automate ?

Exercice 1

- Les mots *bab* et *ababab* sont reconnus

Exercice 1

- Les mots *bab* et *ababab* sont reconnus
- Le mot *abb* n'est pas reconnu

Exercice 1

- Les mots *bab* et *ababab* sont reconnus
- Le mot *abb* n'est pas reconnu
- Les mots reconnus sont les mots qui ont un facteur de la forme *bab*.

$$L(A) = L(\mathscr{A}^\star(bab)\mathscr{A}^\star).$$

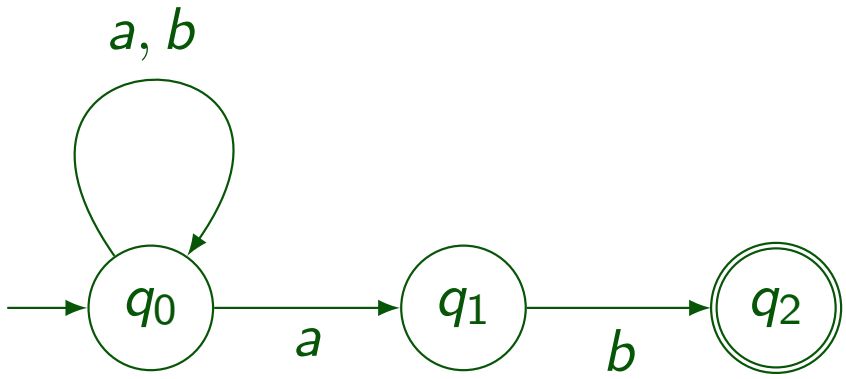
Exercice 2

Déterminer un automate (éventuellement non déterminé) reconnaissant les mots finissant par ab.

Exercice 2

Déterminer un automate (éventuellement non déterminé) reconnaissant les mots finissant par ab.

On peut prendre



Quelques considérations sur la complexité

Attention

Il est important dans les problèmes de complexité sur les automates de faire la différence entre :

- Le temps de **construire** l'automate (et la place qu'il occupe)
- Le temps pour **calculer** dans l'automate

Complexité

Théorème

Soit A un automate.

1. *On suppose que A est déterministe. L'appartenance d'un mot w à $L(A)$ à pour complexité :*
   - *Complexité temporelle :*
   - *Complexité en espace :*

   Dans le cas général. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - *Complexité temporelle :*
   - *Complexité en espace :*

Complexité

Théorème

Soit A un automate.

1. *On suppose que A est déterministe. L'appartenance d'un mot w à $L(A)$ à pour complexité :*
   - *Complexité temporelle : $O(|Q| + |w|)$*
   - *Complexité en espace :*

   Dans le cas général. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - *Complexité temporelle :*
   - *Complexité en espace :*

Complexité

Théorème

Soit A un automate.

1. *On suppose que A est déterministe. L'appartenance d'un mot w à $L(A)$ à pour complexité :*
   - *Complexité temporelle : $O(|Q| + |w|)$*
   - *Complexité en espace : $O(|Q|)$*

   Dans le cas général. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - *Complexité temporelle :*
   - *Complexité en espace :*

Complexité

Théorème

Soit A un automate.

1. *On suppose que A est déterministe. L'appartenance d'un mot w à $L(A)$ à pour complexité :*
   - *Complexité temporelle : $O(|Q| + |w|)$*
   - *Complexité en espace : $O(|Q|)$*

   Dans le cas général. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - *Complexité temporelle : $O(|Q|.|w|)$*
   - *Complexité en espace :*

Complexité

Théorème

Soit A un automate.

1. *On suppose que A est déterministe. L'appartenance d'un mot w à $L(A)$ à pour complexité :*
   - *Complexité temporelle : $O(|Q| + |w|)$*
   - *Complexité en espace : $O(|Q|)$*

   Dans le cas général. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - *Complexité temporelle : $O(|Q|.|w|)$*
   - *Complexité en espace : $O(|Q|)$*

Démonstration

- Dans le cas déterministe, il faut charger l'automate ($O(|Q|)$) puis lire le mot dans l'automate. La lecture de chaque lettre est en temps constant donc l'algorithme est en $O(|Q| + |w|)$.
- Dans le cas général, calculer $\delta(X, x)$ où $X \subset Q$ et $x \in \mathscr{A}$ est en $O(|X|)$. On en déduit une complexité dans le pire des cas en $O(|Q|.|w|)$.

Déterminisation

Déterminisation

Soit L un langage reconnu par un automate A non déterministe. On vient de voir qu'il est préférable d'avoir un automate déterministe reconnaissant ce langage. On va donc construire cet automate déterministe. L'idée principale est de remplacer l'ensemble Q des états par $\mathscr{P}(Q)$.

Automate des parties

Définition

Automate des parties Soit $A = (Q, I, F, \delta)$ un automate non déterministe. On appelle automate des parties l'automate déterministe $A' = (Q', I, F', \delta')$ où :

Automate des parties

Définition

Automate des parties Soit $A = (Q, I, F, \delta)$ un automate non déterministe. On appelle automate des parties l'automate déterministe $A' = (Q', I, F', \delta')$ où :

- $Q' = \mathscr{P}(Q)$.

Automate des parties

Définition

Automate des parties Soit $A = (Q, I, F, \delta)$ un automate non déterministe. On appelle automate des parties l'automate déterministe $A' = (Q', I, F', \delta')$ où :

- $Q' = \mathscr{P}(Q)$.
- $F' = \{X \in \mathscr{P}(Q) \mid X \cap F \neq \emptyset\}$

Automate des parties

Définition

Automate des parties Soit $A = (Q, I, F, \delta)$ un automate non déterministe. On appelle automate des parties l'automate déterministe $A' = (Q', I, F', \delta')$ où :

- $Q' = \mathscr{P}(Q)$.
- $F' = \{X \in \mathscr{P}(Q) \mid X \cap F \neq \emptyset\}$
- L'application δ' définie par $\forall X \in \mathscr{P}(Q), \forall x \in \mathscr{A}$,

Automate des parties

Définition

Automate des parties Soit $A = (Q, I, F, \delta)$ un automate non déterministe. On appelle automate des parties l'automate déterministe $A' = (Q', I, F', \delta')$ où :

- $Q' = \mathscr{P}(Q)$.
- $F' = \{X \in \mathscr{P}(Q) \mid X \cap F \neq \emptyset\}$
- L'application δ' définie par $\forall X \in \mathscr{P}(Q), \forall x \in \mathscr{A}$,

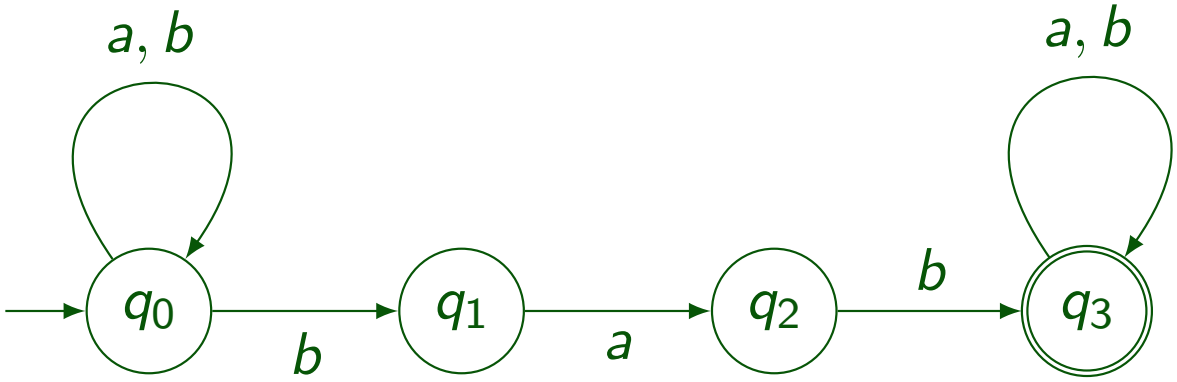
$$\delta'(X, x) = \bigcup_{q \in X} \delta(q, x).$$

Remarque

Notons qu'ici I est un **élément** de $\mathscr{P}(Q)$.

Exemple

Reprenons l'automate



Il a 4 états ce qui fait que l'automate des parties aura $2^4 = 16$ états. Cependant, de nombreux états ne seront pas accessibles

Exemple

En pratique :

Exemple

En pratique :

- On part de l'état initial I (qui est une partie de Q).

Exemple

En pratique :

- On part de l'état initial I (qui est une partie de Q).
- On considère tous les états que l'on obtient en partant de I pour tout lettre de l'alphabet

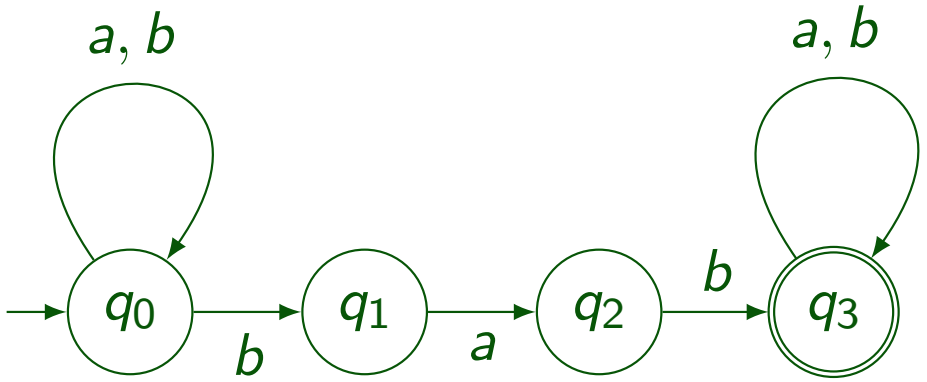
Exemple

En pratique :

- On part de l'état initial I (qui est une partie de Q).
- On considère tous les états que l'on obtient en partant de I pour tout lettre de l'alphabet
- Dès que l'on considère un nouvel état, on rajoute à la liste des états tous les états que l'on peut atteindre à partir de lui.

Exemple

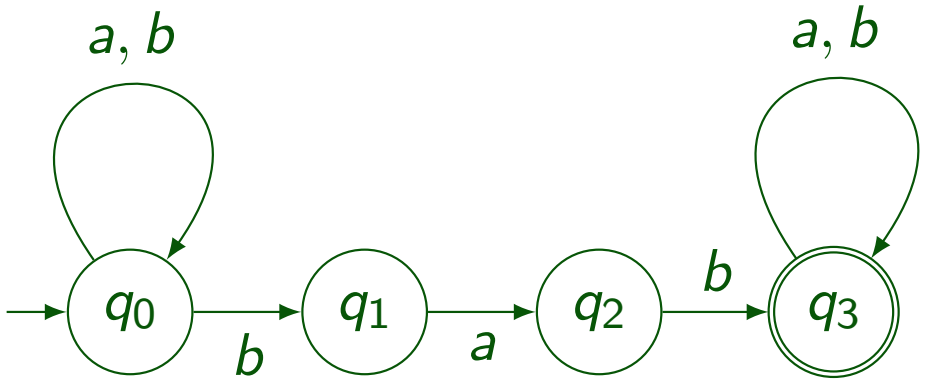
Sur l'automate



état	a	b
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$		

Exemple

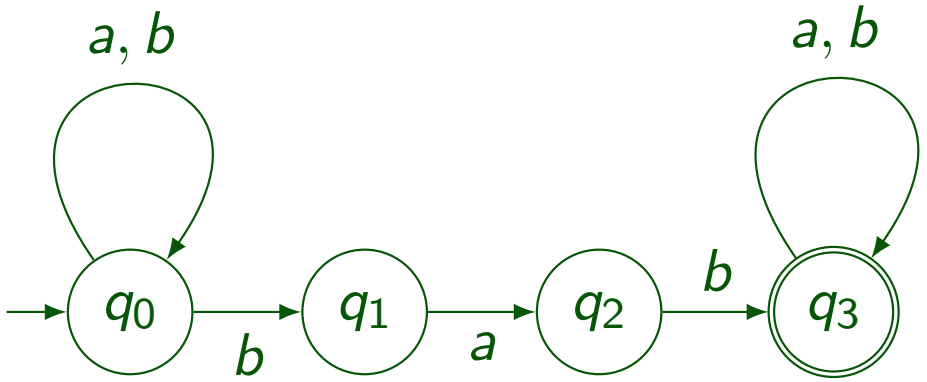
Sur l'automate



état	a	b
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$
$\{q_0, q_2\}$		

Exemple

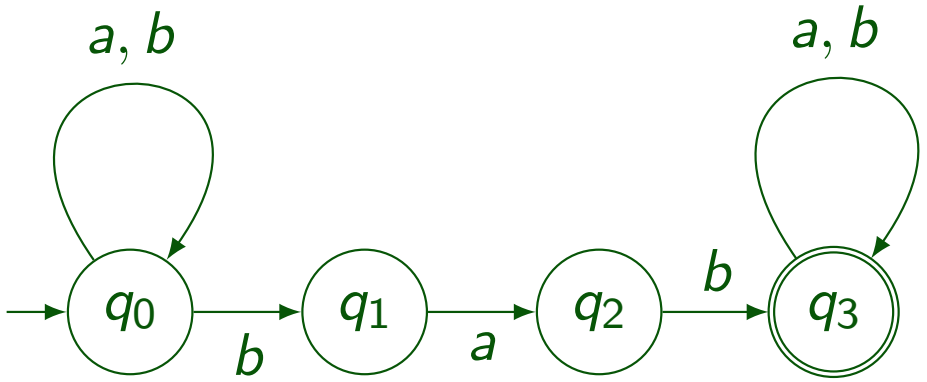
Sur l'automate



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

Exemple

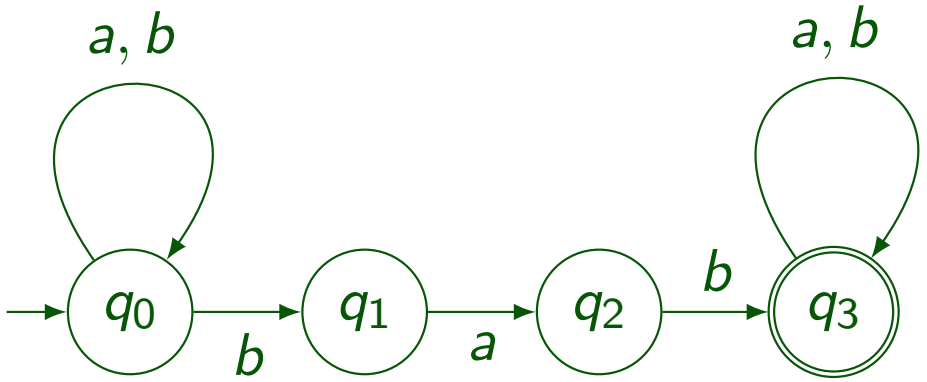
Sur l'automate



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

Exemple

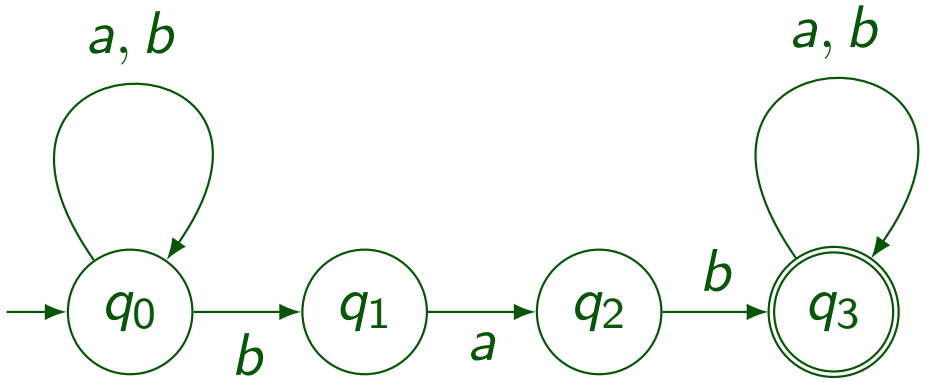
Sur l'automate



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

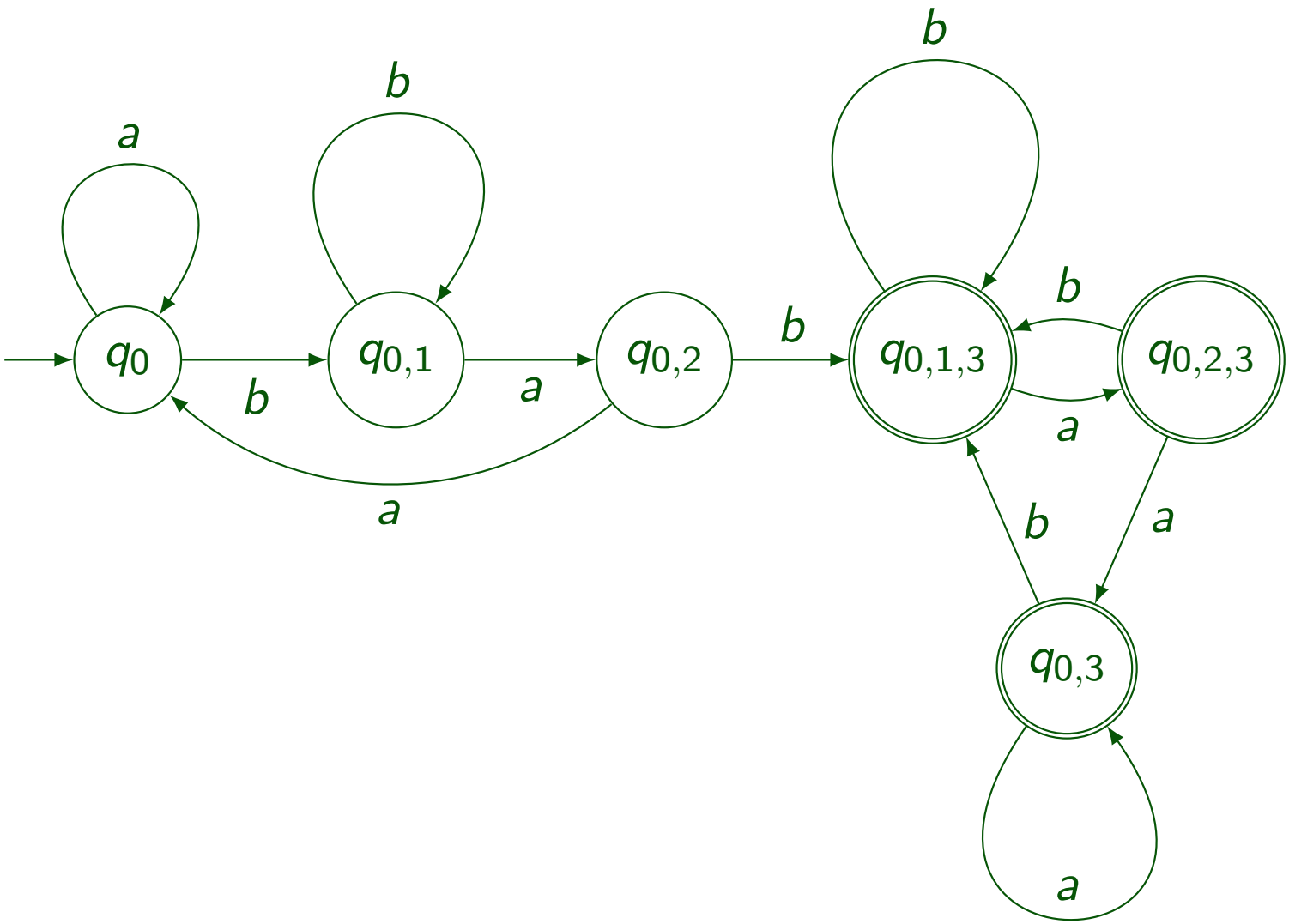
Exemple

Sur l'automate



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

Les états acceptants étant $\{q_0, q_1, q_3\}$ et $\{q_0, q_2, q_3\}$. On obtient donc



Déterminisation

Théorème

Soit $A = (Q, I, F, \delta)$ un automate non déterministe et A' l'automate des parties : $A' = (\mathscr{P}(Q), I, F', \delta')$.

1. *L'automate A' est déterminé*
2. *Les automates A et A' reconnaissent les mêmes langages*

Démonstration

1. Par construction
2. Soit $w \in \mathscr{A}^\star$, on veut vérifier que $w \in L(A) \iff w \in L(A')$.
   On sait que

$$w \in L(A) \iff \exists q_0 \in I, \delta^\star(q_0, w) \cap F \neq \emptyset$$

et

$$w \in L(A') \iff (\delta')^\star(I, w) \in F'.$$

Maintenant, $(\delta')^\star(I, w) = \bigcup_{q_0 \in I} \delta^\star(q_0, w)$ donc

$$\begin{aligned} (\delta')^\star(I, w) \in F' &\iff \left(\bigcup_{q_0 \in I} \delta^\star(q_0, w) \right) \cap F \neq \emptyset \\ &\iff \exists q_0 \in I, \delta^\star(q_0, w) \cap F \neq \emptyset \end{aligned}$$

Déterminisation

Corollaire

Tout langage L reconnu par un automate non déterministe est reconnu par un automate déterministe.

Complexité

On peut donc déterminiser un automate afin de pouvoir dire plus aisément / rapidement si un mot appartient (ou non) à un langage donné.
Cependant, il faut prendre en compte la complexité de la déterminisation et surtout le fait qu'il peut y avoir une explosion du nombre d'états.
A priori, le nombre d'états de l'automate des partie croit exponentiellement en fonction du nombre d'états de l'automate de départ.
Nous verrons par exemple dans l'exercice 13 un langage reconnu par un automate non déterministe à $N+2$ états mais qui n'est pas reconnu par un automate déterministe avec moins de 2^{N+1} états.

Le théorème de Kleene

Énoncé et exemples

Le théorème de Kleene

Nous allons maintenant faire le lien entre les langages réguliers et les automates finis.

Théorème (Théorème de Kleene)

Tout langage régulier est reconnaissable par une automate fini déterministe.

Remarques

- La réciproque est aussi vraie (tout langage reconnaissable par un automate est régulier) mais elle n'est pas au programme.

Remarques

- La réciproque est aussi vraie (tout langage reconnaissable par un automate est régulier) mais elle n'est pas au programme.
- La méthode de démonstration est constructive : on va construire une automate (automate de Glushkov) qui reconnait un langage régulier donné.

Exemple

Avant de se lancer dans la démonstration qui est longue, voyons que ce théorème permettra de montrer que des langages ne sont pas rationnels.
L'argument principal est qu'un automate reconnaissant un langage n'aura qu'un nombre fini d'états.

Exemple

Prenons par exemple le langage $L = \{a^p b^p \mid p \in \mathbf{N}\}$. Montrons qu'il n'est pas rationnel.

Nous allons donner deux rédactions plus ou moins équivalentes.

Exemple

Supposons qu'il est reconnu par un automate déterministe $A = (Q, q_0, F, \delta)$. Cet automate a un nombre fini d'état.
De ce fait, il existe $(n, p) \in \mathbf{N}^2$ avec $n < p$ tels que $q = \delta(q_0, a^n) = \delta(q_0, a^p)$.
On en déduit que $\delta(q, b^n) \in F$ et donc $a^p b^n$ est reconnu ce qui est absurde.

Exemple

Soit $A = (Q, q_0, F, \delta)$ un automate déterministe reconnaissant le langage L. On note $q_n = \delta(q_0, a^n)$ (qui existe car ces états sont accessibles puisque a^n est un préfixe d'un mot de L).
Ils sont deux à deux distincts car pour $n \neq n'$, $\delta(q_n, b^n) \in F$ et $\delta(q_{n'}, b^n) \notin F$.
L'automate a une infinité d'états. C'est absurde.

Remarque

On reformulera ceci à la fin du chapitre avec le lemme de l'étoile.

Exercice

Montrer que le langage des palindromes sur $\mathscr{A} = \{a, b\}$ n'est pas rationnel.

Exercice

Montrer que le langage des palindromes sur $\mathscr{A} = \{a, b\}$ n'est pas rationnel.

Soit $A = (Q, q_0, F, \delta)$ un automate déterministe reconnaissant le langage L. On note $q_n = \delta(q_0, a^n b)$ (qui existe car ces états sont accessibles puisque $a^n b$ est un préfixe d'un mot de L).
Ils sont deux à deux distincts car pour $n \neq n'$, $\delta(q_n, a^n) \in F$ et $\delta(q_{n'}, a^n) \notin F$.
L'automate a une infinité d'états. C'est absurde.

Langages locaux

Définition

Nous aurons besoin par la suite d'une sous-classe de langages : les langages *locaux*. Ce sont les langages qui sont tels que l'on puisse déterminer si un mot w appartienne au langage juste en regardant, la première lettre, la dernière lettre et les facteurs de longueurs 2 (d'où le caractère local).

Définition

Définition

Soit L un langage sur un alphabet $\mathscr{A}$. On définit :

- $P(L) = \{a \in \mathscr{A} \mid a\mathscr{A}^\star \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en tête d'un mot de L (les préfixes)

Définition

Définition

Soit L un langage sur un alphabet $\mathscr{A}$. On définit :

- $P(L) = \{a \in \mathscr{A} \mid a\mathscr{A}^\star \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en tête d'un mot de L (les préfixes)
- $S(L) = \{a \in \mathscr{A} \mid \mathscr{A}^\star a \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en fin d'un mot de L (les suffixes)

Définition

Définition

Soit L un langage sur un alphabet $\mathscr{A}$. On définit :

- $P(L) = \{a \in \mathscr{A} \mid a\mathscr{A}^\star \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en tête d'un mot de L (les préfixes)
- $S(L) = \{a \in \mathscr{A} \mid \mathscr{A}^\star a \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en fin d'un mot de L (les suffixes)
- $F(L) = \{u \in \mathscr{A}^2 \mid \mathscr{A}^\star u \mathscr{A}^\star \cap L \neq \emptyset\}$ l'ensembles des facteurs de longueur 2 d'un mot de L

Définition

Définition

Soit L un langage sur un alphabet $\mathscr{A}$. On définit :

- $P(L) = \{a \in \mathscr{A} \mid a\mathscr{A}^\star \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en tête d'un mot de L (les préfixes)
- $S(L) = \{a \in \mathscr{A} \mid \mathscr{A}^\star a \cap L \neq \emptyset\}$ l'ensembles des lettres qui apparaissent en fin d'un mot de L (les suffixes)
- $F(L) = \{u \in \mathscr{A}^2 \mid \mathscr{A}^\star u \mathscr{A}^\star \cap L \neq \emptyset\}$ l'ensembles des facteurs de longueur 2 d'un mot de L
- $N(L) = \mathscr{A}^2 \setminus F(L)$ les facteurs de longueur 2 qui n'apparaissent pas

Définition

Théorème

Soit L un langage,

$$L \setminus \{\varepsilon\} \subset (P(L)\mathscr{A}^\star \cap \mathscr{A}^\star S(L)) \setminus \mathscr{A}^\star N(L)\mathscr{A}^\star$$

Définition

Théorème

Soit L un langage,

$$L \setminus \{\varepsilon\} \subset (P(L)\mathscr{A}^\star \cap \mathscr{A}^\star S(L)) \setminus \mathscr{A}^\star N(L)\mathscr{A}^\star$$

Démonstration : Par définition.

Définition

Définition

Soit L un langage, il est dit **local** s'il existe des parties P et S de $\mathscr{A}$ et une partie N de $\mathscr{A}^2$ telles que

$$L \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$$

Remarques

- Si L est un langage local, nécessairement $P = P(L), S = S(L)$ et $N = N(L)$.

Remarques

- Si L est un langage local, nécessairement $P = P(L), S = S(L)$ et $N = N(L)$.
- On préfère souvent définir le langage par le triplet (P, S, F) plutôt que (P, S, N), c'est-à-dire avec les facteurs que l'on accepte plutôt que ceux que l'on refuse.

Exemple 1

Le langage $L = \{a^n b^m \mid (n, m) \in \mathbf{N}^2\}$ est un langage local.

Exemple 1

Le langage $L = \{a^n b^m \mid (n, m) \in \mathbf{N}^2\}$ est un langage local.

- On prend $P = \{a, b\}$ et $S = \{a, b\}$

Exemple 1

Le langage $L = \{a^n b^m \mid (n, m) \in \mathbf{N}^2\}$ est un langage local.

- On prend $P = \{a, b\}$ et $S = \{a, b\}$
- On prend $F = \{aa, ab, bb\}$ et donc $N = \{ba\}$.

Exemple 1

Le langage $L = \{a^n b^m \mid (n, m) \in \mathbf{N}^2\}$ est un langage local.

- On prend $P = \{a, b\}$ et $S = \{a, b\}$
- On prend $F = \{aa, ab, bb\}$ et donc $N = \{ba\}$.

On a bien $L \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$

Exemple 2

Le langage $L = (ab)\mathscr{A}^\star$ des mots qui commencent par ab n'est pas local.

Exemple 2

Le langage $L = (ab)\mathscr{A}^\star$ des mots qui commencent par ab n'est pas local.

S'il l'était on aurait

- $P = \{a\}$ et $S = \{a, b\}$

Exemple 2

Le langage $L = (ab)\mathscr{A}^\star$ des mots qui commencent par ab n'est pas local.

S'il l'était on aurait

- $P = \{a\}$ et $S = \{a, b\}$
- $F = \{aa, ab, bb, ba\}$ et donc $N = \emptyset$.

Exemple 2

Le langage $L = (ab)\mathscr{A}^\star$ des mots qui commencent par ab n'est pas local.

S'il l'était on aurait

- $P = \{a\}$ et $S = \{a, b\}$
- $F = \{aa, ab, bb, ba\}$ et donc $N = \emptyset$.

On a alors $(P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$ qui est le langage des mots qui commencent par a. Ce n'est pas L.

Propriétés de clôture

On veut savoir si l'ensemble des langages locaux est stable par les opérations ensemblistes sur les langages : intersection, union, concaténation et étoile.

Intersection

Théorème

L'intersection de deux langages locaux est un langage local.

Intersection

Théorème

L'intersection de deux langages locaux est un langage local.

Démonstration : Soit L_1 et L_2 deux langages locaux. On note pour $i \in \{1,2\}$ P_i, S_i et N_i les préfixes, suffixes et facteurs interdits de L_i. Soit $w \in \mathscr{A}^\star$

$$\begin{aligned} w \in L_1 \cap L_2 \setminus \{\varepsilon\} &\iff w \in (L_1 \setminus \{\varepsilon\}) \cap (L_2 \setminus \{\varepsilon\}) \\ &\iff w \in (P_1\mathscr{A}^\star \cap \mathscr{A}^\star S_1) \cap (P_2\mathscr{A}^\star \cap \mathscr{A}^\star S_2) \\ &\qquad \text{et } w \notin \mathscr{A}^\star N_1 \mathscr{A}^\star \text{ et } w \notin \mathscr{A}^\star N_2 \mathscr{A}^\star \\ &\iff w \in (P_1 \cap P_2)\mathscr{A}^\star \cap \mathscr{A}^\star(S_1 \cap S_2) \\ &\qquad \text{et } w \notin \mathscr{A}^\star(N_1 \cup N_2)\mathscr{A}^\star \end{aligned}$$

Intersection

Finalement,

$$L_1 \cap L_2 \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$$

où $P = P_1 \cap P_2, S = S_1 \cap S_2$ et $N = N_1 \cup N_2$ (et donc $F = \overline{N} = F_1 \cap F_2$). Le langage $L_1 \cap L_2$ est bien local.

Union et concatenation

Passons au cas de l'union et de la concaténation. Ce n'est pas aussi simple.

Exercice : Montrer que les langages $L_1 = L(a\star)$ et $L_2 = L((ab)\star)$ sont locaux mais que ni $L_1 L_2$ ni $L_1 \cup L_2$ ne le sont

Exercice

- Le langage L_1 est local. On pose $P_1 = \{a\}, S_1 = \{a\}$ et $F_1 = \{aa\}$ donc $N_1 = \{ab, ba, bb\}$.

Exercice

- Le langage L_1 est local. On pose $P_1 = \{a\}, S_1 = \{a\}$ et $F_1 = \{aa\}$ donc $N_1 = \{ab, ba, bb\}$.
- Le langage L_2 est local. On pose $P_2 = \{a\}, S_1 = \{b\}$ et $F_1 = \{ab, ba\}$ donc $N_1 = \{aa, bb\}$.

Exercice

- Le langage $L_1 \cup L_2 =$ n'est local. En effet
  $P(L_1 \cup L_2) = \{a\}, S(L_1 \cup L_2) = \{a, b\}$ et
  $F(L_1 \cup L_2) = \{aa, ab, ba\}$ donc $N(L_1 \cup L_2) = \{bb\}$.
  On a alors
  $aab \in P(L_1 \cup L_2)\mathscr{A}^\star \cap \mathscr{A}^\star S(L_1 \cup L_2) \setminus \mathscr{A}^\star N(L_1 \cup L_2)\mathscr{A}^\star$
  mais pour autant $aab \notin L_1 \cup L_2$.

Exercice

- Le langage $L_1 \cup L_2 =$ n'est local. En effet
  $P(L_1 \cup L_2) = \{a\}, S(L_1 \cup L_2) = \{a, b\}$ et
  $F(L_1 \cup L_2) = \{aa, ab, ba\}$ donc $N(L_1 \cup L_2) = \{bb\}$.
  On a alors
  $aab \in P(L_1 \cup L_2)\mathscr{A}^\star \cap \mathscr{A}^\star S(L_1 \cup L_2) \setminus \mathscr{A}^\star N(L_1 \cup L_2)\mathscr{A}^\star$
  mais pour autant $aab \notin L_1 \cup L_2$.
- Le langage $L_1 L_2$ n'est pas local. En effet
  $P(L_1 L_2) = \{a\}, S(L_1 L_2) = \{a, b\}$ et $F(L_1 L_2) = \{aa, ab, ba\}$
  donc $N(L_1 L_2) = \{bb\}$.
  On a alors $aba \in P(L_1 L_2)\mathscr{A}^\star \cap \mathscr{A}^\star S(L_1 L_2) \setminus \mathscr{A}^\star N(L_1 L_2)\mathscr{A}^\star$
  mais pour autant $aba \notin L_1 L_2$.

Union et concaténation

Théorème

Soit L_1 et L_2 deux langages locaux **sur des alphabets disjoints** *alors $L_1 \cup L_2$ et $L_1.L_2$ sont des langages locaux.*

Démonstration

On note comme d'habitude, pour $i \in \{1,2\}$ P_i, S_i et F_i les préfixes, suffixes et facteurs autorisés de L_i. On note $\mathscr{A}_1$ (resp. $\mathscr{A}_2$) l'alphabet sur lequel est défini L_1 (resp. L_2) ainsi que $\mathscr{A} = \mathscr{A}_1 \cup \mathscr{A}_2$.

On veut montrer que le langage $L_1 \cup L_2$ sur l'alphabet $\mathscr{A}$ est local.
On pose $P = P(L_1 \cup L_2) = P_1 \cup P_2$, $S = S(L_1 \cup L_2) = S_1 \cup S_2$ et $F = F(L_1 \cup F_2) = F_1 \cup F_2$.
On veut montrer que

$$L_1 \cup L_2 \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$$

où $N = (\mathscr{A})^2 \setminus F$.

Démonstration

Si on considère un mot $w = w_1 \dots w_n \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$ alors $w_1 \in P = P_1 \cup P_2$.
Supposons (par symétrie) que $w_1 \in P_1 \subset \mathscr{A}_1$.
Maintenant, $w_1 w_2 \in F = F_1 \cup F_2$ mais comme $w_1 \in \mathscr{A}_1$ et que L_1 et L_2 sont définis sur des alphabets distincts, nécessairement $w_1 w_2 \in F_1$ et donc $w_2 \in \mathscr{A}_1$. On montre ainsi, de proche en proche, que $w \in \mathscr{A}_1^\star$. Il appartient alors à $(P_1\mathscr{A}_1^\star \cap \mathscr{A}_1^\star S_1) \setminus \mathscr{A}_1^\star N_1 \mathscr{A}_1^\star = L_1 \setminus \{\varepsilon\}$ car L_1 est local.

Démonstration

Gardons les mêmes notations. On pose cette fois

$$P = P(L_1.L_2) = \begin{cases} P_1 & \text{si } \varepsilon \notin L_1 \\ P_1 \cup P_2 & \text{si } \varepsilon \in L_1 \end{cases}$$

De même on pose

$$S = S(L_1.L_2) = \begin{cases} S_2 & \text{si } \varepsilon \notin L_2 \\ S_1 \cup S_2 & \text{si } \varepsilon \in L_2 \end{cases}$$

Pour finir, on pose

$$F = F(L_1.L_2) = F_1 \cup F_2 \cup S_1 \times P_2$$

Démonstration

On procède alors de même soit
$w = w_1 w_2 \cdots w_n \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$.

Démonstration

On procède alors de même soit
$w = w_1 w_2 \cdots w_n \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$.

- Si $w \in \mathscr{A}_2^\star$ alors $w_1 \in P_2$ (c'est donc que $\varepsilon \in L_1$). On a aussi $w_n \in S_2$ et tous les facteurs sont dans F_2 donc $w \in L_2 \subset L_1.L_2$ car $\varepsilon \in L_1$.

Démonstration

On procède alors de même soit
$w = w_1 w_2 \cdots w_n \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$.

- Si $w \in \mathscr{A}_2^\star$ alors $w_1 \in P_2$ (c'est donc que $\varepsilon \in L_1$). On a aussi $w_n \in S_2$ et tous les facteurs sont dans F_2 donc $w \in L_2 \subset L_1.L_2$ car $\varepsilon \in L_1$.
- Si $w \in \mathscr{A}_1^\star$ on procède de même (cette fois $\varepsilon \in L_2$).

Démonstration

- Si $w \notin (\mathscr{A}_1^\star \cup \mathscr{A}_2^\star)$. On a alors $w_1 \in P_1$ et $w_n \in S_2$ car si une lettre w_k appartient à $\mathscr{A}_2$ toutes celles qui suivent aussi d'après la forme des facteurs autorisés. On note $u = w_1 \cdots w_p$ le plus grand préfixe appartenant à $\mathscr{A}_1^\star$. On a alors $w_p \in \mathscr{A}_1$ et $w_{p+1} \in \mathscr{A}_2$ donc le facteur $w_p w_{p+1} \in S_1 \times P_2$. On montre alors que $u \in L_1$ et $w_{p+1} \cdots w_n \in L_2$.

Dans tous les cas, $w \in L_1.L_2$.

Étoile de Kleene

Théorème

Soit L un langage local, le langage $L^\star$ est encore un langage local.

Démonstration : Si on pose encore $P = P(L), S = S(L)$ et $F = F(L)$.
On voit que $P(L^\star) = P$, $S(L^\star) = S$ et $F(L^\star) = F \cup S \times P$.
Comme précédemment, on peut montrer par récurrence sur la longueur du mot que si $w \in (P(L^\star)\mathscr{A}^\star \cap \mathscr{A}^\star S(L^\star)) \setminus \mathscr{A}^\star N(L^\star)\mathscr{A}^\star$ alors $w \in L^\star$. L'idée est de « couper » le mot $w = w_1 \cdots w_n$ dès que l'on voit un facteur $w_p w_{p+1} \in S \times P$. Dans ce cas, $w_1 \cdots w_p \in L$ et on recommence avec le suffixe $w_{p+1} \cdots w_n$.

Expressions rationnelles linéaires

Définition

Définition (Expressions rationnelles linéaires)

Une expression rationnelle est dite linéaire si toute lettre de l'alphabet apparait au plus une fois

Exemples : Les expressions $(a|b)\star$, $a\star|b$ sont linéaires.
L'expression $a\star|ba$ ne l'est pas.

Langage associé

Théorème

Le langage associé a une expression régulière linéaire est local.

Langage associé

Théorème

Le langage associé a une expression régulière linéaire est local.

Démonstration : Cela se démontre par récurrence sur la longueur de l'expression e ou par induction structurelle.

Démonstration

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.

Démonstration

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = \varepsilon$, $L(e) = \{\varepsilon\}$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.

Démonstration

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = \varepsilon$, $L(e) = \{\varepsilon\}$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = a$ avec $a \in \mathscr{A}$, $L(e) = \{a\}$ qui est local défini par $P = \{a\} S = \{a\}$ et $N = \mathscr{A}^2$

Démonstration

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = \varepsilon$, $L(e) = \{\varepsilon\}$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = a$ avec $a \in \mathscr{A}$, $L(e) = \{a\}$ qui est local défini par $P = \{a\} S = \{a\}$ et $N = \mathscr{A}^2$
- si $e = e_1|e_2$ alors $L(e) = L(e_1) \cup L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Alors $L(e_1) \cup L(e_2)$ est local d'après ce qui précède.

Démonstration

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = \varepsilon$, $L(e) = \{\varepsilon\}$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = a$ avec $a \in \mathscr{A}$, $L(e) = \{a\}$ qui est local défini par $P = \{a\} S = \{a\}$ et $N = \mathscr{A}^2$
- si $e = e_1|e_2$ alors $L(e) = L(e_1) \cup L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Alors $L(e_1) \cup L(e_2)$ est local d'après ce qui précède.
- si $e = e_1 e_2$ alors $L(e) = L(e_1)L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Là encore, $L(e_1)L(e_2)$ est local.

Démonstration

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = \varepsilon$, $L(e) = \{\varepsilon\}$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = a$ avec $a \in \mathscr{A}$, $L(e) = \{a\}$ qui est local défini par $P = \{a\} S = \{a\}$ et $N = \mathscr{A}^2$
- si $e = e_1|e_2$ alors $L(e) = L(e_1) \cup L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Alors $L(e_1) \cup L(e_2)$ est local d'après ce qui précède.
- si $e = e_1 e_2$ alors $L(e) = L(e_1)L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Là encore, $L(e_1)L(e_2)$ est local.
- Si $e = e_1\star$. Alors $L(e_1)$ est local et donc $L(e)$ aussi.

Remarque

La réciproque n'est pas vraie. Par exemple le langage $aa\star$ est un langage local ($P = \{a\}, S = \{\varepsilon\}, F = \{aa\}$) mais il n'est pas associé à une expression linéaire.

Méthode

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$				

Méthode

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$	F	$\emptyset$	$\emptyset$	$\emptyset$
$\{\varepsilon\}$				

Méthode

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$	F	$\emptyset$	$\emptyset$	$\emptyset$
$\{\varepsilon\}$	V	$\emptyset$	$\emptyset$	$\emptyset$
a				

Méthode

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$	F	$\emptyset$	$\emptyset$	$\emptyset$
$\{\varepsilon\}$	V	$\emptyset$	$\emptyset$	$\emptyset$
a	F	$\{a\}$	$\{a\}$	$\emptyset$
$f\star$				

Méthode

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$	F	$\emptyset$	$\emptyset$	$\emptyset$
$\{\varepsilon\}$	V	$\emptyset$	$\emptyset$	$\emptyset$
a	F	$\{a\}$	$\{a\}$	$\emptyset$
$f\star$	V	P	S	$F \cup S \times P$
$e_1 \mid e_2$				

Méthode

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$	F	$\emptyset$	$\emptyset$	$\emptyset$
$\{\varepsilon\}$	V	$\emptyset$	$\emptyset$	$\emptyset$
a	F	$\{a\}$	$\{a\}$	$\emptyset$
$f\star$	V	P	S	$F \cup S \times P$
$e_1 \vert e_2$	$\lambda(e_1) \vert\vert \lambda(e_2)$	$P(e_1) \cup P(e_2)$	$S(e_1) \cup S(e_2)$	$F(e_1) \cup F(e_2)$

Méthode

Pour $e_1 e_2$ c'est un peu plus compliqué. On a :

- $\lambda(e_1 e_2) = \lambda(e_1) \&\& \lambda(e_2)$

Méthode

Pour $e_1 e_2$ c'est un peu plus compliqué. On a :

- $\lambda(e_1 e_2) = \lambda(e_1) \&\& \lambda(e_2)$
- $P(e_1 e_2) = \begin{cases} P(e_1) & \text{si } \lambda(e_1) = F \\ P(e_1) \cup P(e_2) & \text{si } \lambda(e_1) = V \end{cases}$

Méthode

Pour $e_1 e_2$ c'est un peu plus compliqué. On a :

- $\lambda(e_1 e_2) = \lambda(e_1) \&\& \lambda(e_2)$
- $P(e_1 e_2) = \begin{cases} P(e_1) & \text{si } \lambda(e_1) = F \\ P(e_1) \cup P(e_2) & \text{si } \lambda(e_1) = V \end{cases}$
- $S(e_1 e_2) = \begin{cases} S(e_1) & \text{si } \lambda(e_2) = F \\ S(e_1) \cup S(e_2) & \text{si } \lambda(e_2) = V \end{cases}$

Méthode

Pour $e_1 e_2$ c'est un peu plus compliqué. On a :

- $\lambda(e_1 e_2) = \lambda(e_1) \&\& \lambda(e_2)$
- $P(e_1 e_2) = \begin{cases} P(e_1) & \text{si } \lambda(e_1) = F \\ P(e_1) \cup P(e_2) & \text{si } \lambda(e_1) = V \end{cases}$
- $S(e_1 e_2) = \begin{cases} S(e_1) & \text{si } \lambda(e_2) = F \\ S(e_1) \cup S(e_2) & \text{si } \lambda(e_2) = V \end{cases}$
- $F(e_1 e_2) = F(e_1) \cup F(e_2) \cup S(e_1) \times P(e_2)$

Exemple

Prenons l'expression $e = x_1 x_2 \star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2				

Exemple

Prenons l'expression $e = x_1 x_2 \star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2	F	$\{x_2\}$	$\{x_2\}$	$\emptyset$
$x_2\star$				

Exemple

Prenons l'expression $e = x_1 x_2 \star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2	F	$\{x_2\}$	$\{x_2\}$	$\emptyset$
$x_2\star$	V	$\{x_2\}$	$\{x_2\}$	$\{x_2x_2\}$
x_1				

Exemple

Prenons l'expression $e = x_1 x_2 \star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2	F	$\{x_2\}$	$\{x_2\}$	$\emptyset$
$x_2\star$	V	$\{x_2\}$	$\{x_2\}$	$\{x_2 x_2\}$
x_1	F	$\{x_1\}$	$\{x_1\}$	$\emptyset$
$x_1 x_2\star$				

Exemple

Prenons l'expression $e = x_1 x_2 \star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2	F	$\{x_2\}$	$\{x_2\}$	$\emptyset$
$x_2\star$	V	$\{x_2\}$	$\{x_2\}$	$\{x_2x_2\}$
x_1	F	$\{x_1\}$	$\{x_1\}$	$\emptyset$
$x_1x_2\star$	F	$\{x_1\}$	$\{x_1, x_2\}$	$\{x_1x_2, x_2x_2\}$

Exemple

Prenons l'expression $e = x_1 x_2 \star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2	F	$\{x_2\}$	$\{x_2\}$	$\emptyset$
$x_2\star$	V	$\{x_2\}$	$\{x_2\}$	$\{x_2x_2\}$
x_1	F	$\{x_1\}$	$\{x_1\}$	$\emptyset$
$x_1x_2\star$	F	$\{x_1\}$	$\{x_1, x_2\}$	$\{x_1x_2, x_2x_2\}$

Bien évidemment dans le cas présent on aurait pu donner directement la réponse.

Attention

On peut maintenant expliquer le principe de la démonstration du théorème de Kleene. Si on se donne un langage régulier L on peut lui associer une expression régulière e. Ensuite on va linéariser cette expression pour obtenir une expression linéaire f dont le langage associé sera local. On saura alors définir un automate reconnaissant ce langage. Il ne restera plus qu'à faire marche arrière.

Linéarisation

Définition (Linéarisation)

Soit e une expression rationnelle sur $\mathscr{A}$. On dit que l'on linéarise l'expression en remplaçant chaque lettre par la lettre x_k où k est la position de la lettre dans l'expression. L'expression obtenue est linéaire par construction

Exemple : Soit $e = (ab|b) \star ba$. La linéarisation est $(x_1x_2|x_3) \star x_4x_5$.

Linéarisation

Afin de pouvoir faire marche arrière, il faut garder en mémoire les lettres initiales. Cela peut se faire à l'aide d'une tableau en indiquant en position i la $\ll$ valeur $\gg$ de la lettre x_i.

Exemple : Dans notre exemple $e = (ab|b) \star ba$ on récupère le tableau

k	1	2	3	4	5
lettre(k)	a	b	b	b	a

Algorithme de Berry-Sethi

Automate local

Définition

Un automate fini déterministe $A = (Q, q_0, F, \delta)$ est dit local si pour toute lettre x de $\mathscr{A}$, il existe un état q' tel que pour tout état q, (q, x) est un blocage ou $\delta(q, x) = q'$.

Automate local

Définition

Un automate fini déterministe $A = (Q, q_0, F, \delta)$ est dit local si pour toute lettre x de $\mathscr{A}$, il existe un état q' tel que pour tout état q, (q, x) est un blocage ou $\delta(q, x) = q'$.

Remarque : Cela signifie que toutes les transitions indicées par une lettre x fixée arrivent au même état. C'est-à-dire que lors de la lecture d'un mot, en regardant la dernière lettre lue, on sait dans quel état on est arrivé.
Cela peut s'écrire

$$\forall x \in \mathscr{A}, \#\{q' \in Q \mid \exists q \in Q, \delta(q, x) = q'\} \leqslant 1$$

Automate local

Théorème

Tout langage local est reconnu par un automate local.

Démonstration

On se donne un langage local L sur $\mathscr{A}$ défini par (P, S, F) où F est l'ensemble des facteurs de L. On définit alors l'automate local suivant $A = (Q, q_0, S, \delta)$ où :

Démonstration

On se donne un langage local L sur $\mathscr{A}$ défini par (P, S, F) où F est l'ensemble des facteurs de L. On définit alors l'automate local suivant $A = (Q, q_0, S, \delta)$ où :

- L'ensemble des états $Q = \mathscr{A} \cup \{q_0\}$ est l'alphabet $\mathscr{A}$ sur lequel est défini le langage auquel on ajoute un élément q_0 (on suppose que q_0 n'est pas un élément de $\mathscr{A}$).

Démonstration

On se donne un langage local L sur $\mathscr{A}$ défini par (P, S, F) où F est l'ensemble des facteurs de L. On définit alors l'automate local suivant $A = (Q, q_0, S, \delta)$ où :

- L'ensemble des états $Q = \mathscr{A} \cup \{q_0\}$ est l'alphabet $\mathscr{A}$ sur lequel est défini le langage auquel on ajoute un élément q_0 (on suppose que q_0 n'est pas un élément de $\mathscr{A}$).
- L'état initial est q_0.

Démonstration

On se donne un langage local L sur $\mathscr{A}$ défini par (P, S, F) où F est l'ensemble des facteurs de L. On définit alors l'automate local suivant $A = (Q, q_0, S, \delta)$ où :

- L'ensemble des états $Q = \mathscr{A} \cup \{q_0\}$ est l'alphabet $\mathscr{A}$ sur lequel est défini le langage auquel on ajoute un élément q_0 (on suppose que q_0 n'est pas un élément de $\mathscr{A}$).
- L'état initial est q_0.
- Les états finaux sont les suffixes S si ε n'appartient pas à L et $S \cup \{q_0\}$ si $\varepsilon \in L$.

Démonstration

On se donne un langage local L sur $\mathscr{A}$ défini par (P, S, F) où F est l'ensemble des facteurs de L. On définit alors l'automate local suivant $A = (Q, q_0, S, \delta)$ où :

- L'ensemble des états $Q = \mathscr{A} \cup \{q_0\}$ est l'alphabet $\mathscr{A}$ sur lequel est défini le langage auquel on ajoute un élément q_0 (on suppose que q_0 n'est pas un élément de $\mathscr{A}$).
- L'état initial est q_0.
- Les états finaux sont les suffixes S si ε n'appartient pas à L et $S \cup \{q_0\}$ si $\varepsilon \in L$.
- On défini δ par $\delta(q_0, x) = x$ si $x \in P$ et $\delta(y, x) = x$ si $yx \in F$. Toutes les autres couples sont des blocages de l'automate.

Démonstration

On voit que cet automate est un automate fini, déterministe, standard (par contre il n'est pas complet à priori). De plus il est local car toutes les transitions définies par une lettre x arrivent sur l'état x.

Il est clair que tous les mots du langage L sont reconnus par l'automate par construction.

Démonstration

Réciproquement soit $w_1 w_2 \cdots w_n$ un mot (non vide) reconnu par A. On peut le schématiser par :

$$q_0 \xrightarrow{w_1} w_1 \xrightarrow{w_2} \cdots w_{n-1} \xrightarrow{w_n} w_n.$$

Alors

- $w_1 \in P$ (car sinon $\delta(q, w_1)$ est un blocage)
- $w_n \in S$ car c'est un état final et qu'il ne peut valoir q_0
- pour tout $k \in [\![1\,;\, n-1]\!]$ $w_k w_{k+1} \in F$ car $\delta(w_k, w_{k+1}) = w_{k+1}$.

Donc $w \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus (\mathscr{A}^\star N \mathscr{A}^\star) = L \setminus \{\varepsilon\}$.

Remarque

L'automate défini ainsi admet $\#\mathscr{A} + 1$ états.

Exemple

Si on reprend l'expression linéaire $(x_1x_2|x_3)\star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2	F	$\{x_1\}$	$\{x_2\}$	$\{x_1x_2\}$
x_3				
$(x_1x_2\|x_3)$				
$(x_1x_2\|x_3)\star$				
x_4x_5				
e				

Exemple

Si on reprend l'expression linéaire $(x_1x_2|x_3)\star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2	F	$\{x_1\}$	$\{x_2\}$	$\{x_1x_2\}$
x_3	F	$\{x_3\}$	$\{x_3\}$	$\emptyset$
$(x_1x_2\|x_3)$				
$(x_1x_2\|x_3)\star$				
x_4x_5				
e				

Exemple

Si on reprend l'expression linéaire $(x_1x_2|x_3)\star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2	F	$\{x_1\}$	$\{x_2\}$	$\{x_1x_2\}$
x_3	F	$\{x_3\}$	$\{x_3\}$	$\emptyset$
$(x_1x_2\|x_3)$	F	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2\}$
$(x_1x_2\|x_3)\star$				
x_4x_5				
e				

Exemple

Si on reprend l'expression linéaire $(x_1x_2|x_3)\star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2	F	$\{x_1\}$	$\{x_2\}$	$\{x_1x_2\}$
x_3	F	$\{x_3\}$	$\{x_3\}$	$\emptyset$
$(x_1x_2\|x_3)$	F	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2\}$
$(x_1x_2\|x_3)\star$	V	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2, x_2x_1, x_2x_3, x_3x_1, x_3x_3\}$
x_4x_5				
e				

Exemple

Si on reprend l'expression linéaire $(x_1x_2|x_3)\star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

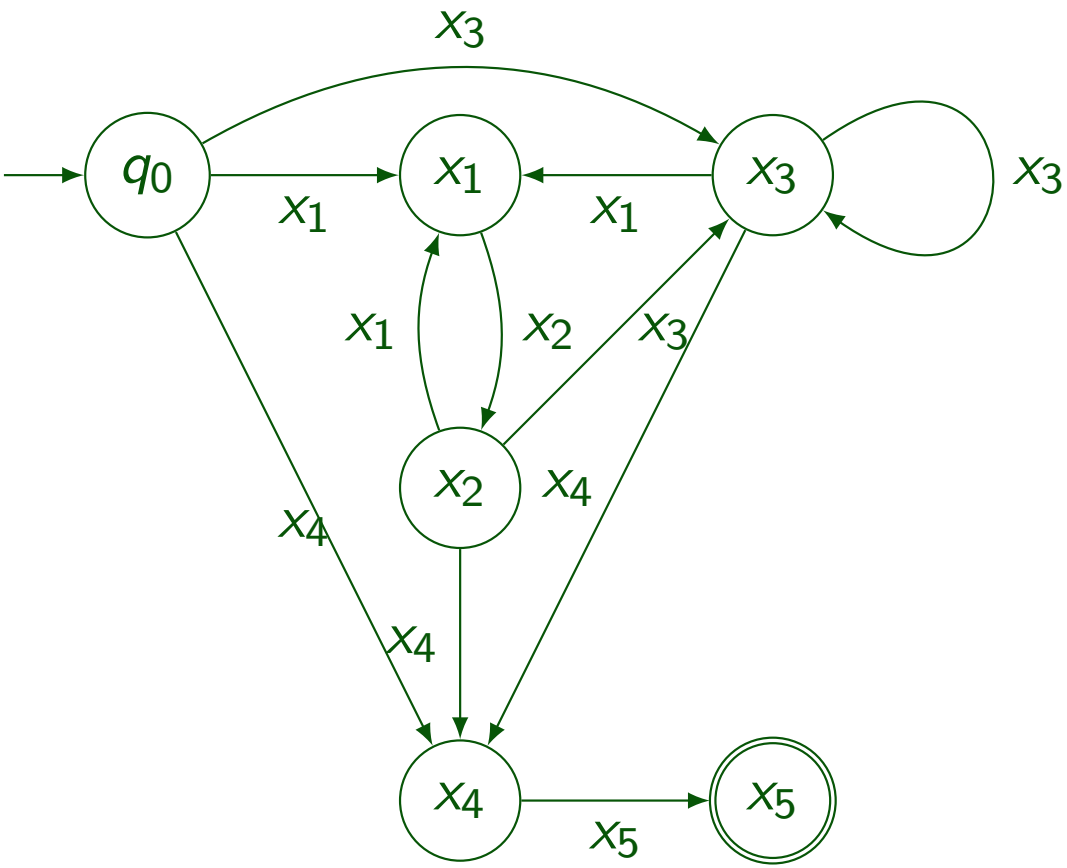
e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2	F	$\{x_1\}$	$\{x_2\}$	$\{x_1x_2\}$
x_3	F	$\{x_3\}$	$\{x_3\}$	$\emptyset$
$(x_1x_2\|x_3)$	F	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2\}$
$(x_1x_2\|x_3)\star$	V	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2, x_2x_1, x_2x_3, x_3x_1, x_3x_3\}$
x_4x_5	F	$\{x_4\}$	$\{x_5\}$	$\{x_4x_5\}$
e				

Exemple

Si on reprend l'expression linéaire $(x_1x_2|x_3)\star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

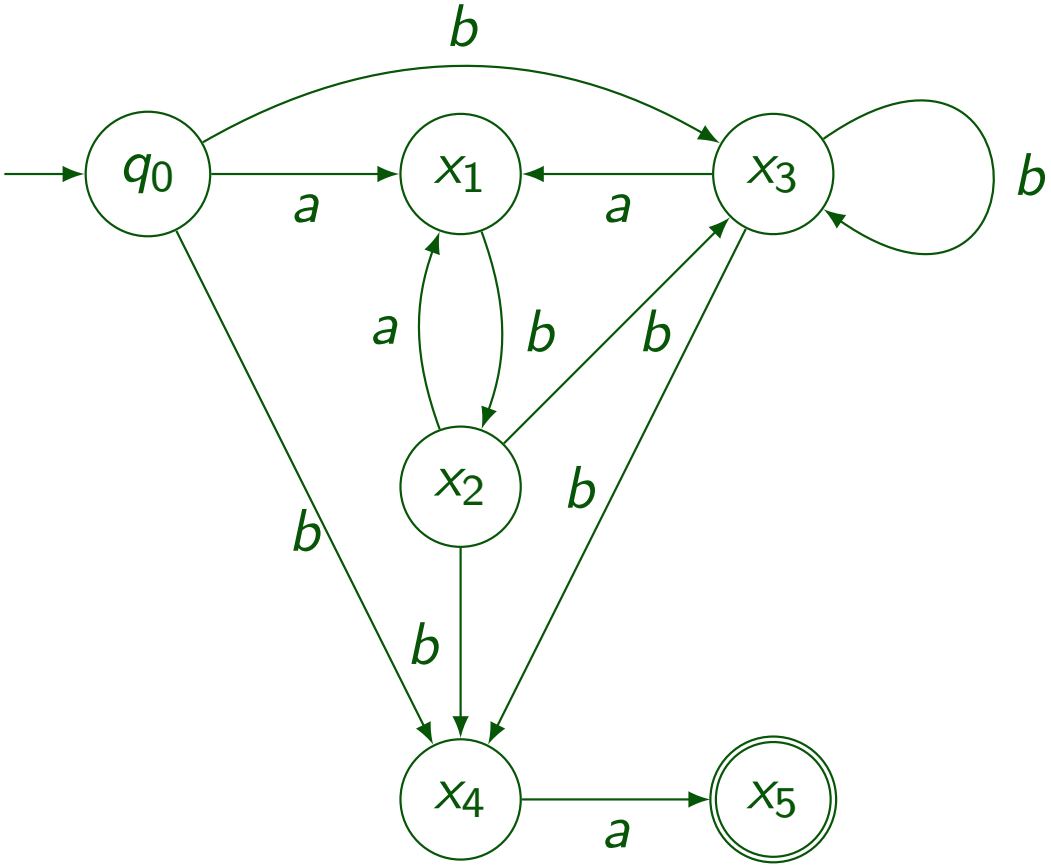
e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2	F	$\{x_1\}$	$\{x_2\}$	$\{x_1x_2\}$
x_3	F	$\{x_3\}$	$\{x_3\}$	$\emptyset$
$(x_1x_2\|x_3)$	F	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2\}$
$(x_1x_2\|x_3)\star$	V	$\{x_1, x_3\}$	$\{x_2, x_3\}$	$\{x_1x_2, x_2x_1, x_2x_3, x_3x_1, x_3x_3\}$
x_4x_5	F	$\{x_4\}$	$\{x_5\}$	$\{x_4x_5\}$
e	F	$\{x_1, x_3, x_4\}$	$\{x_5\}$	$\{x_1x_2, x_2x_1, x_2x_3, x_3x_1, x_3x_3, x_4x_5, x_2x_4, x_3x_4\}$

Exemple



Exemple

Il suffit alors de remplacer de les étiquettes des transitions par les lettres initiales de l'alphabet (que l'on avait pris garde de conserver).



ATTENTION

Attention

L'automate obtenu n'est à priori pas déterministe

Définition

Définition

On a associé ainsi à tout expression rationnelle un automate (à priori non déterministe). Il s'appelle l'automate de **Glushkov** de l'expression.

Théorème

L'automate de Glushkov associé à une expression rationnelle e reconnait le langage $L(e)$.

Démonstration

Soit $\mathscr{A}$ l'alphabet sur lequel est défini e. On note :

- e' l'expression rationnelle linéaire associée à e obtenue précédemment
- $L(e')$ le langage reconnu par cette expression régulière
- $\mathscr{B}$ l'alphabet sur lequel est défini e'.

On note f l'application de $\mathscr{B}$ dans $\mathscr{A}$ qui associe à une lettre la lettre dont elle est issue (c'est une application de « démarquage »).

Démonstration

On a construit un automate A' reconnaissant $L(e')$.
On note A l'automate de Glushkov obtenu en replaçant les x_i par $f(x_i)$.
Soit $w \in \mathscr{A}^\star$,

$$A \text{ reconnait } w \iff \text{il existe } w' \in \mathscr{B}^\star \text{ tel que } f(w') = w \text{ et } A' \text{ reconnait } w'.$$

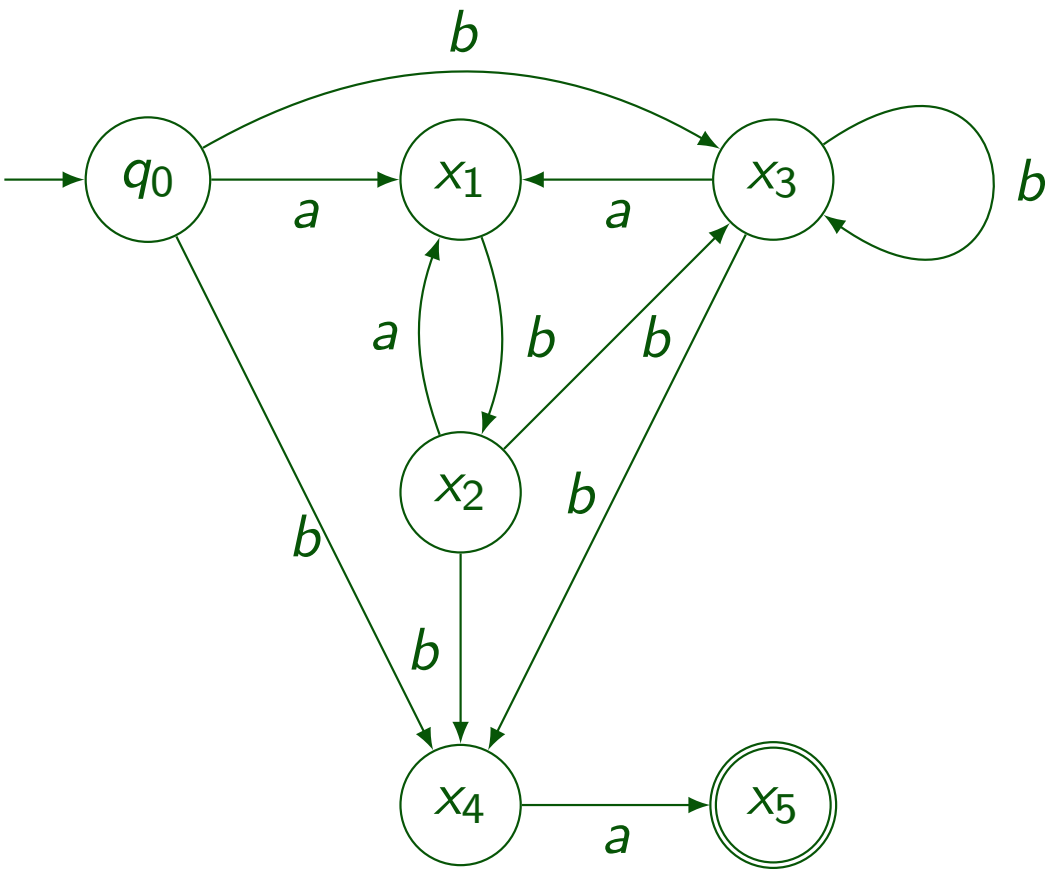
On en déduit que le langage reconnu par A est $f(L(e')) = L(e)$.

Attention

Si on veut obtenir un automate déterministe il suffit d'appliquer la détermination à l'automate de Glushkov.

Exemple

On reprend notre automate



Exemple

On le déterminise :

état	a	b
$\{q_0\}$		

Exemple

On le déterminise :

état	a	b
$\{q_0\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1\}$		
$\{x_3, x_4\}$		

Exemple

On le déterminise :

état	a	b
$\{q_0\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1\}$	$\emptyset$	$\{x_2\}$
$\{x_3, x_4\}$		
$\{x_2\}$		

Exemple

On le déterminise :

état	a	b
$\{q_0\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1\}$	$\emptyset$	$\{x_2\}$
$\{x_3, x_4\}$	$\{x_1, x_5\}$	$\{x_3, x_4\}$
$\{x_2\}$		
$\{x_1, x_5\}$		

Exemple

On le déterminise :

état	a	b
$\{q_0\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1\}$	$\emptyset$	$\{x_2\}$
$\{x_3, x_4\}$	$\{x_1, x_5\}$	$\{x_3, x_4\}$
$\{x_2\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1, x_5\}$		

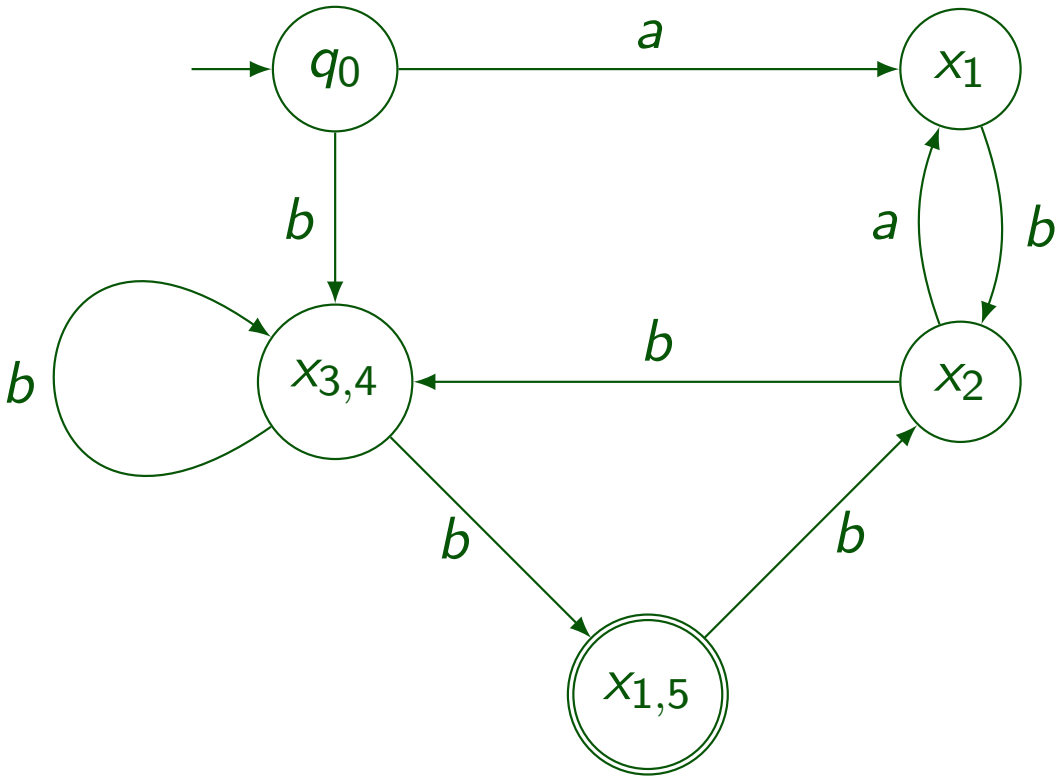
Exemple

On le déterminise :

état	a	b
$\{q_0\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1\}$	$\emptyset$	$\{x_2\}$
$\{x_3, x_4\}$	$\{x_1, x_5\}$	$\{x_3, x_4\}$
$\{x_2\}$	$\{x_1\}$	$\{x_3, x_4\}$
$\{x_1, x_5\}$	$\emptyset$	$\{x_2\}$

Exemple

On obtient alors l'automate **déterministe**



Théorème

On a bien démontré le sens direct du théorème de Kleene

Théorème (Théorème de Kleene)

Tout langage régulier est reconnaissable par une automate fini déterministe.

Propriétés de cloture

Introduction

Introduction

On a vu que tout langage rationnel était reconnu par un automate fini. La réciproque est aussi vraie (résultat hors programme - voir plus loin). De ce fait, les langages reconnaissables par un automates sont exactement les langages rationnels.

Introduction

Par définition on sait que $R(\mathscr{A})$ l'ensemble des langages rationnels est stable par union, concaténation et par étoile. On veut essayer de retrouver ces propriétés via les automates, c'est à dire que si on se donne deux automates A et A' reconnaissant les langages $L(A)$ et $L(A')$, on veut construire des automates reconnaissants les langages $L(A) \cup L(A')$, $L(A)L(A')$ et $L(A)\star$.

On en profitera aussi pour voir les propriétés de clôture pour d'autres opérations ensemblistes : intersection, complémentaire, différence et différence symétrique.

Automate produit et intersection

Automate produit

Définition

On se donne deux automates déterministes $A = (Q, q_0, F, \delta)$ et $A' = (Q', q_0', F', \delta')$. On appelle automate produit l'automate $A \times A' = (Q \times Q', (q_0, q_0'), F \times F', \Delta)$ où $\forall (q, q') \in Q \times Q', \forall x \in \mathscr{A}$,

$$\Delta((q, q'), x) = (\delta(q, x), \delta'(q', x)) \text{ si les deux existent.}$$

Remarques

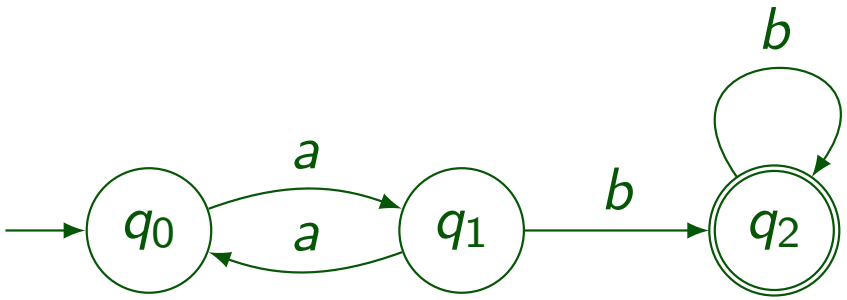
- La notation pour l'automate produit n'est pas standardisée. On trouve $A \times A'$, $A \oplus A'$, $A \otimes A'$,...

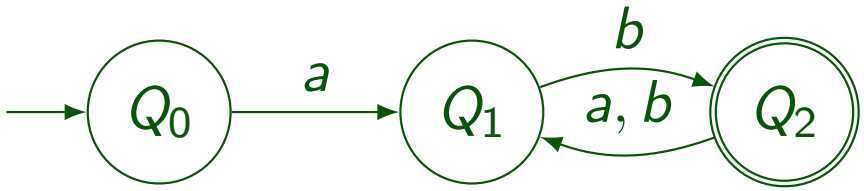
Remarques

- La notation pour l'automate produit n'est pas standardisée. On trouve $A \times A'$, $A \oplus A'$, $A \otimes A'$,...
- Avec les notations de la définition, $((q, q'), x)$ est un blocage dès que (q, x) ou (q', x) en est un

Exemple

On considère les deux automates suivants





Exemple

Ils reconnaissant les langages :

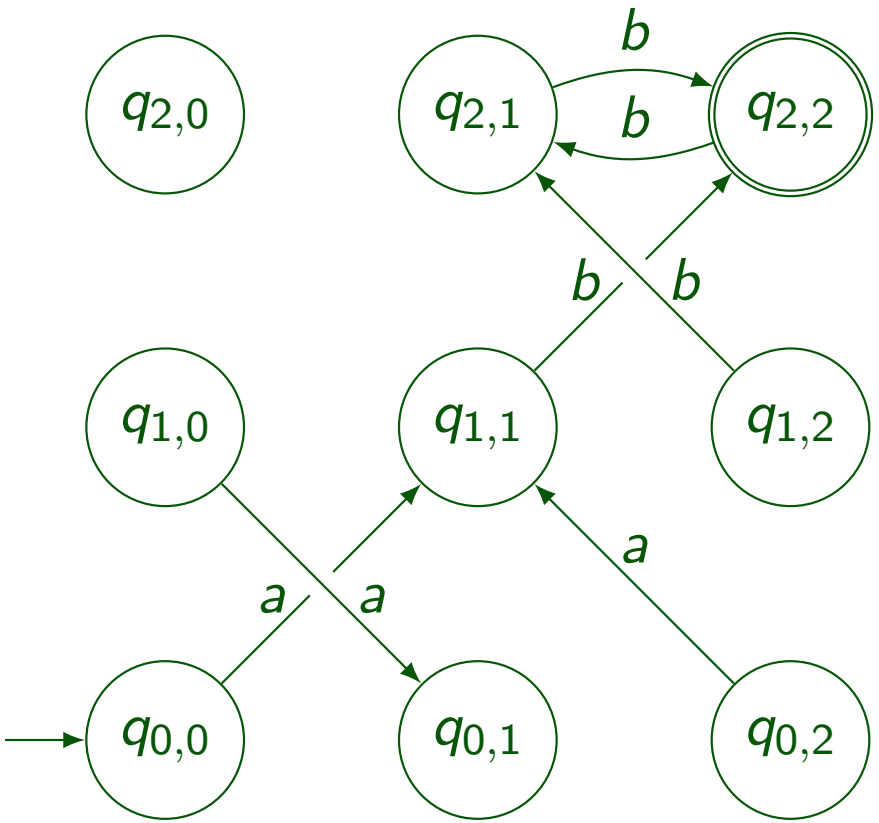
Exemple

Ils reconnaissant les langages :

$$aa\star\, bb\star \text{ et } a(ba|bb)\star\, b$$

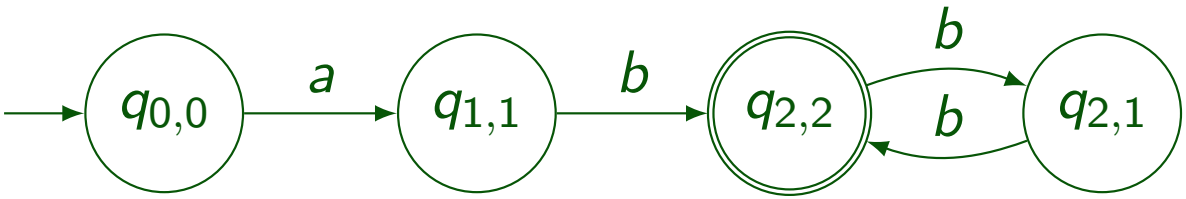
Exemple

On note $q_{i,j}$ l'état (q_i, Q_j). On a alors l'automate produit



Exemple

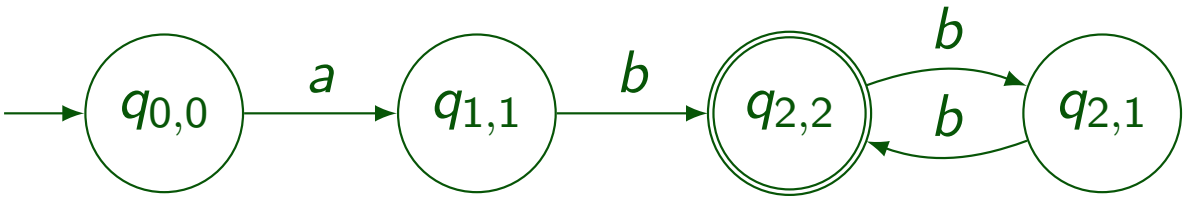
Automate que l'on peut émonder pour obtenir



Le langage reconnu est

Exemple

Automate que l'on peut émonder pour obtenir



Le langage reconnu est $abb\star$

Stabilité par intersection

Théorème

Avec les notations précédentes, le langage reconnu par l'automate produit est $L(A) \cap L(A')$

Démonstration

- Soit $w = w_0 w_1 \dots w_n \in L(A) \cap L(A')$. Il existe un calcul dans A

$$q_0 \xrightarrow{w_0} \cdots \xrightarrow{w_n} q$$

où $q \in F$ ainsi q'un calcul dans A'

$$q'_0 \xrightarrow{w_0} \cdots \xrightarrow{w_n} q'$$

où $q' \in F'$.
Cela nous donne donc un calcul dans l'automate produit :

$$(q_0, q'_0) \xrightarrow{w_0} \cdots \xrightarrow{w_n} (q, q')$$

et $(q, q') \in F \times F'$ est un état terminal de $A \times A'$ donc $w \in L(A \times A')$.

Démonstration

- Réciproquement si $w \in L(A \times A')$ il existe un calcul

$$(q_0, q_0') \xrightarrow{w_0} \cdots \xrightarrow{w_n} (q, q')$$

dans $A \times A'$ où (q, q') est terminal, c'est-à-dire $q \in F$ et $q' \in F'$. On peut alors « projeter » ce calcul dans A et dans A' pour obtenir que $w \in L(A) \cap L(A')$.

Stabilité par intersection

Théorème

L'ensemble des langages reconnus par un automate fini est stable par intersection.

Réunion

Réunion

On se donne encore deux automates déterministes $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut cette fois construire un automate $A \oplus A'$ tel que $L(A \oplus A') = L(A) \cup L(A')$. On peut penser faire la même chose que ci-dessus mais en prenant pour états finals les états

Réunion

On se donne encore deux automates déterministes $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut cette fois construire un automate $A \oplus A'$ tel que $L(A \oplus A') = L(A) \cup L(A')$. On peut penser faire la même chose que ci-dessus mais en prenant pour états finals les états

$$F \times Q' \cup Q \times F'.$$

Réunion

On se donne encore deux automates déterministes $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut cette fois construire un automate $A \oplus A'$ tel que $L(A \oplus A') = L(A) \cup L(A')$. On peut penser faire la même chose que ci-dessus mais en prenant pour états finals les états

$$F \times Q' \cup Q \times F'.$$

Cela « ne marche pas ». En effet si un mot est reconnu par A et donne un blocage dans A' il ne sera pas reconnu par $A \oplus A'$.

Réunion

Théorème

Soit L et L' deux langages et deux automates déterministes **complets** *$A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$ qui reconnaissent respectivement L et L'. Le langage $L \cup L'$ est reconnu par l'automate $A \otimes A' = (Q \times Q', (q_0, q'_0), F \times Q' \cup Q \times F', \Delta)$ où*

$$\forall (q, q') \in Q \times Q', \forall x \in \mathscr{A}, \Delta((q, q'), x) = (\delta(q, x), \delta'(q', x))$$

Réunion

Théorème

L'ensemble des langages reconnus par un automate fini est stable par union.

Réunion

Théorème

L'ensemble des langages reconnus par un automate fini est stable par union.

Démonstration : On sait que si L et L' sont des langages reconnus par des automates finis (déterministes). Ils sont reconnus par des automates complets (quitte à ajouter un « puit »). On peut alors considérer l'automate $A \oplus A'$ qui reconnait $L \cup L'$.

Complémentaire

Complémentaire

Théorème

Soit L un langage rationnel et $A = (Q, q_0, F, \delta)$ un automate **complet** *qui reconnait L. Le langage $\overline{L} = \mathscr{A}^\star \setminus L$ est reconnu par l'automate $A' = (Q, q_0, Q \setminus F, \delta)$.*

Complémentaire

Théorème

Soit L un langage rationnel et $A = (Q, q_0, F, \delta)$ un automate **complet** *qui reconnait L. Le langage $\overline{L} = \mathscr{A}^\star \setminus L$ est reconnu par l'automate $A' = (Q, q_0, Q \setminus F, \delta)$.*

Démonstration : Il suffit de voir que pour tout $w \in \mathscr{A}^\star$,

$$w \in L \iff \delta(q_0, w) \in F$$

puisqu'il n'y a pas de blocages.

Différence et différence symétrique

Différence et différence symétrique

Théorème

Soit L et L' deux langages et deux automates déterministes **complets** *$A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$ qui reconnaissent respectivement L et L'.*

- *L'automate :*

  reconnait $L \setminus L'$

- *L'automate*

  reconnait $L \Delta L' = (L \cup L') \setminus (L \cap L')$

Différence et différence symétrique

Théorème

Soit L et L' deux langages et deux automates déterministes **complets** *$A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$ qui reconnaissent respectivement L et L'.*

- *L'automate :*

$$(Q \times Q', (q_0, q'_0), F \times (Q' \setminus F'), \Delta)$$

reconnait $L \setminus L'$

- *L'automate*

reconnait $L \Delta L' = (L \cup L') \setminus (L \cap L')$

Différence et différence symétrique

Théorème

Soit L et L' deux langages et deux automates déterministes **complets** *$A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$ qui reconnaissent respectivement L et L'.*

- *L'automate :*

$$(Q \times Q', (q_0, q'_0), F \times (Q' \setminus F'), \Delta)$$

reconnait $L \setminus L'$

- *L'automate*

$$(Q \times Q', (q_0, q'_0), (F \times Q' \cup Q \times F') \setminus (F \times F'), \Delta)$$

reconnait $L \Delta L' = (L \cup L') \setminus (L \cap L')$

Différence et différence symétrique

Théorème

On peut décider si deux automates reconnaissent le même langages car $L = L' \iff L \Delta L' = \emptyset$

Concaténation

Concaténation

Soit L et L' deux langages reconnus par les automates $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut construire un automate reconnaissant LL'.

Concaténation

Soit L et L' deux langages reconnus par les automates $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut construire un automate reconnaissant LL'.
L'idée est de « rebrancher » les états finals de A sur l'état initial de A'. Il y a deux méthodes :

Concaténation

Soit L et L' deux langages reconnus par les automates $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut construire un automate reconnaissant LL'.
L'idée est de « rebrancher » les états finals de A sur l'état initial de A'. Il y a deux méthodes :

- On fait une « transition instantanée » ou ε-transition entre les états finals de A et q'_0. **Ce n'est plus au programme**.

Concaténation

Soit L et L' deux langages reconnus par les automates $A = (Q, q_0, F, \delta)$ et $A' = (Q', q_0', F', \delta')$. On veut construire un automate reconnaissant LL'.
L'idée est de « rebrancher » les états finals de A sur l'état initial de A'. Il y a deux méthodes :

- On fait une « transition instantanée » ou ε-transition entre les états finals de A et q_0'. **Ce n'est plus au programme**.
- On va fusionner les états finals de A avec q_0', pour cela on créer des transitions entre les états finals de A et les atteints à partir de q_0' dans A'.

Concaténation

On suppose que

- L'automate A' est standard (ce qui est toujours possible).
- Les états Q et Q' sont disjoints.

On construit un automate **non-déterministe** reconnaissant LL' par

$$(Q \cup Q' \setminus \{q_0'\}, \{q_0\}, \tilde{F}, \Delta)$$

où Δ est définie par $\forall (q,x) \in (Q \cup Q' \setminus \{q_0'\}) \times \mathscr{A}$,

$$\Delta(q,x) = \begin{cases} \{\delta(q,x)\} & \text{si } q \in Q \setminus F \\ \{\delta'(q,x)\} & \text{si } q \in Q' \setminus \{q_0'\} \\ \{\delta(q,x), \delta'(q_0',x)\} & \text{si } q \in F \end{cases}$$

Concaténation

Avec

$$\tilde{F} = \begin{cases} F' & \text{si } \varepsilon \notin L' \\ F \cup F' & \text{sinon.} \end{cases}$$

Remarques

- On voit que cet automate reconnait bien LL' - le prouver en exercice.
- Il est nécessaire que l'automate A' soit standard car on enlève q'_0.

Étoile

Étoile

Soit L un langage reconnus par l'automate $A = (Q, q_0, F, \delta)$. On peut construire un automate non déterministe reconnaissant le langage $L\star$. Il suffit pour cela :

- Ajouter l'état initial aux états finaux (pour accepter le mot vide s'il ne l'était pas).
- Pour tout $(p, x) \in Q \times \mathscr{A}$ tel que $p = \delta(q_0, x)$ et tout état final q on ajoute une flèche $q \xrightarrow{x} p$.

Compléments

Lemme de l'étoile et applications

Lemme de l'étoile

Nous avons déjà vu comment montrer qu'un langage n'était pas rationnel en se basant que s'il l'était alors il serait reconnu par un automate ayant un nombre fini d'états. Il existe une version un peu plus sophistiquée de cet argument.

Lemme de l'étoile

Nous avons déjà vu comment montrer qu'un langage n'était pas rationnel en se basant que s'il l'était alors il serait reconnu par un automate ayant un nombre fini d'états. Il existe une version un peu plus sophistiquée de cet argument.

L'idée est que si un mot de longueur n est reconnu par un automate ayant N états et que $n \geqslant N$ alors, tout calcul réussi dans l'automate sur ce mot passe par $n+1$ états et donc (au moins) deux fois par le même état.

Lemme de l'étoile

Théorème (Lemme de l'étoile)

Soit L un langage rationnel, il existe un entier N tel que tout mot w tel que $|w| \geqslant N$ peut s'écrire $w = xyz$ tel que

- $0 < |y| \leqslant N$ - *en particulier* $y \neq \varepsilon$.
- *Pour tout* $k \in \mathbf{N}, xy^k z \in L$

Remarques

- Cela s'appelle aussi le lemme de pompage.

Remarques

- Cela s'appelle aussi le lemme de pompage.
- Cela implique que $x(y\star)z \subset L$.

Démonstration

On sait qu'il existe un automate fini déterministe A reconnaissant L. Si on note N le nombre d'état de A. Tout mot w de L est reconnu par A. De ce fait, il existe un calcul réussi dans A sur w :

$$q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \cdots \xrightarrow{w_p} q_p$$

où $p = |w|$.

Démonstration

On sait qu'il existe un automate fini déterministe A reconnaissant L. Si on note N le nombre d'état de A. Tout mot w de L est reconnu par A. De ce fait, il existe un calcul réussi dans A sur w :

$$q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \cdots \xrightarrow{w_p} q_p$$

où $p = |w|$.

Maintenant, si on suppose $p \geqslant N$, on voit que le calcul ci-dessus passe par $p+1 > N$ états. Par le principe des tiroirs, il existe $(k, k') \in [\![0\,;\, p]\!]^2$ tels que $q_k = q_{k'}$.
Cela signifie que le calcul fait une « boucle » dans l'automate.

Démonstration

On pose alors

$$x = w_1 \cdots w_k ; y = w_{k+1} \cdots w_{k'} \text{ et } z = w_{k'+1} \cdots w_p.$$

On a donc

$$q_k \xrightarrow{y} q_{k'} = q_k.$$

De ce fait, pour tout $n \in \mathbb{N}$, $xy^n z$ est reconnu par l'automate et donc est dans L.

ATTENTION

Le lemme de l'étoile n'est pas explicitement au programme. Si vous voulez l'utiliser vous **devez** le redémontrer.

Exemple

On note L le langage

$$L = \{a, ab, abba, abbabaab, abbabaabbaababba, \cdots\}$$

défini comme étant le plus petit langage contenant a et tel que si $w \in L$ alors $w\overline{w} \in L$ où $\overline{w}$ est obtenu à partir de w en remplaçant les a par des b et réciproquement.
Montrons que L n'est pas rationnel.

Exemple

Supposons par l'absurde que L soit rationnel. Considérons alors un automate reconnaissant L et notons N son nombre d'états. On voit que L ne contient que des mots de longueur 2^p et que réciproquement pour tout entier p, L contient un mot de longueur 2^p.

Exemple

Supposons par l'absurde que L soit rationnel. Considérons alors un automate reconnaissant L et notons N son nombre d'états. On voit que L ne contient que des mots de longueur 2^p et que réciproquement pour tout entier p, L contient un mot de longueur 2^p.
Soit w un mot tel que $|w| \geqslant N$. Par le lemme de l'étoile, il existe $w = xyz$ tel que $\forall k \in \mathbf{N}, xy^k z \in L$. Si on note $2^p = |w|$ et $r = |y| > 0$, on en déduit que $2^p + r$ et $2^p + 2r$ sont des puissances de 2 ce qui est absurde car :

Exemple

Supposons par l'absurde que L soit rationnel. Considérons alors un automate reconnaissant L et notons N son nombre d'états. On voit que L ne contient que des mots de longueur 2^p et que réciproquement pour tout entier p, L contient un mot de longueur 2^p.
Soit w un mot tel que $|w| \geqslant N$. Par le lemme de l'étoile, il existe $w = xyz$ tel que $\forall k \in \mathbf{N}, xy^k z \in L$. Si on note $2^p = |w|$ et $r = |y| > 0$, on en déduit que $2^p + r$ et $2^p + 2r$ sont des puissances de 2 ce qui est absurde car :

- Si $r < 2^p$: on a $2^p < 2^p + r < 2^p + 2^p = 2^{p+1}$

Exemple

Supposons par l'absurde que L soit rationnel. Considérons alors un automate reconnaissant L et notons N son nombre d'états. On voit que L ne contient que des mots de longueur 2^p et que réciproquement pour tout entier p, L contient un mot de longueur 2^p.
Soit w un mot tel que $|w| \geqslant N$. Par le lemme de l'étoile, il existe $w = xyz$ tel que $\forall k \in \mathbf{N}, xy^k z \in L$. Si on note $2^p = |w|$ et $r = |y| > 0$, on en déduit que $2^p + r$ et $2^p + 2r$ sont des puissances de 2 ce qui est absurde car :

- Si $r < 2^p$: on a $2^p < 2^p + r < 2^p + 2^p = 2^{p+1}$
- Si $r = 2^p$: on a $2^p + 2r = 3.2^p$ qui est divisible par 3 et n'est donc pas une puissance de 2.

Finalement L n'est pas rationnel.

Réciproque du théorème de Kleene

Réciproque du théorème de Kleene

On veut justifier la réciproque du théorème de Kleene, à savoir que tout langage reconnu par un automate est rationnel.
Là encore, la preuve est algorithmique. Nous allons, à partir d'un automate fini déterministe A, construire une expression rationnelle e telle que $L(e) = L(A)$.

Réciproque du théorème de Kleene

On veut justifier la réciproque du théorème de Kleene, à savoir que tout langage reconnu par un automate est rationnel.
Là encore, la preuve est algorithmique. Nous allons, à partir d'un automate fini déterministe A, construire une expression rationnelle e telle que $L(e) = L(A)$. Présentons, sur un exemple l'algorithme BMC (Brzozowski-McCluskey). Il existe aussi un autre algorithme (McNaughton Yamada).

Réciproque du théorème de Kleene

On se donne un automate déterministe et on veut construire une expression rationnelle. Pour travailler on étiquette les flèches de l'automate par des expressions rationnelles et plus nécessairement des lettres de $\mathscr{A}$. L'algorithme est le suivant :

Réciproque du théorème de Kleene

1. On ajoute deux nouveaux états α et ω. On relie α à l'état initial q_0 par une transition étiquetée par ε. On relie les états terminaux à ω par une transition ε.
   Le but est d'obtenir un automate standard qui n'a qu'un état final. Si l'automate est standard et qu'il existe un unique état final on peut sauter cette étape

Réciproque du théorème de Kleene

1. On ajoute deux nouveaux états α et ω. On relie α à l'état initial q_0 par une transition étiquetée par ε. On relie les états terminaux à ω par une transition ε.
   Le but est d'obtenir un automate standard qui n'a qu'un état final. Si l'automate est standard et qu'il existe un unique état final on peut sauter cette étape
2. On simplifie l'automate en appliquant les deux méthodes ci-dessous autant que possible.

Réciproque du théorème de Kleene

1. On ajoute deux nouveaux états α et ω. On relie α à l'état initial q_0 par une transition étiquetée par ε. On relie les états terminaux à ω par une transition ε.
   Le but est d'obtenir un automate standard qui n'a qu'un état final. Si l'automate est standard et qu'il existe un unique état final on peut sauter cette étape
2. On simplifie l'automate en appliquant les deux méthodes ci-dessous autant que possible.
   - S'il existe deux états p, q et deux transition $p \xrightarrow{e_1} q$ et $p \xrightarrow{e_2} q$ on les remplace par $p \xrightarrow{e_1|e_2} q$. Ici p peut-être égal à q.

Réciproque du théorème de Kleene

1. On ajoute deux nouveaux états α et ω. On relie α à l'état initial q_0 par une transition étiquetée par ε. On relie les états terminaux à ω par une transition ε.
   Le but est d'obtenir un automate standard qui n'a qu'un état final. Si l'automate est standard et qu'il existe un unique état final on peut sauter cette étape
2. On simplifie l'automate en appliquant les deux méthodes ci-dessous autant que possible.
   - S'il existe deux états p, q et deux transition $p \xrightarrow{e_1} q$ et $p \xrightarrow{e_2} q$ on les remplace par $p \xrightarrow{e_1|e_2} q$. Ici p peut-être égal à q.
   - On supprime un état q (différent de α et ω), et pour tous les états (p, r) distincts de q (mais éventuellement égaux) tels qu'il existe des transitions $p \xrightarrow{e_1} q$, $q \xrightarrow{f} q$ et $q \xrightarrow{e_2} r$, on ajoute une transition $p \xrightarrow{e_1 f^\star e_2} r$

Réciproque du théorème de Kleene

On aboutit à un automate

Remarques :

Réciproque du théorème de Kleene

On aboutit à un automate

Remarques :

- Cet algorithme termine car il diminue le nombres d'états et le nombres de transitions (penser à un ordre lexicographique)

Réciproque du théorème de Kleene

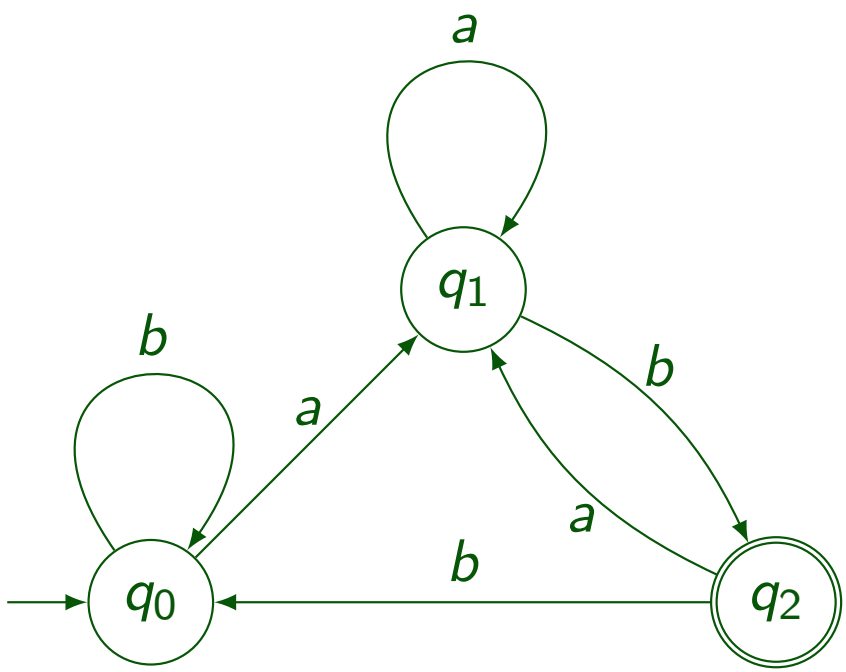
On aboutit à un automate

Remarques :

- Cet algorithme termine car il diminue le nombres d'états et le nombres de transitions (penser à un ordre lexicographique)
- Par construction, le language reconnu par l'automate est celui qui est définit par l'expression rationnelle trouvée.

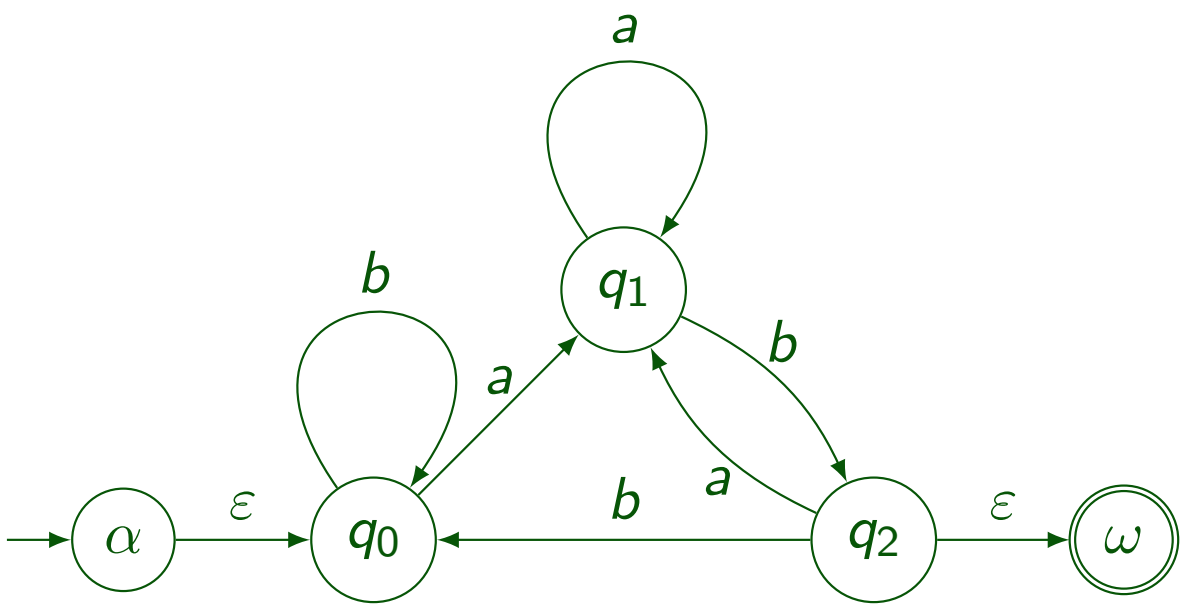
Exemple

Considérons l'automate suivant



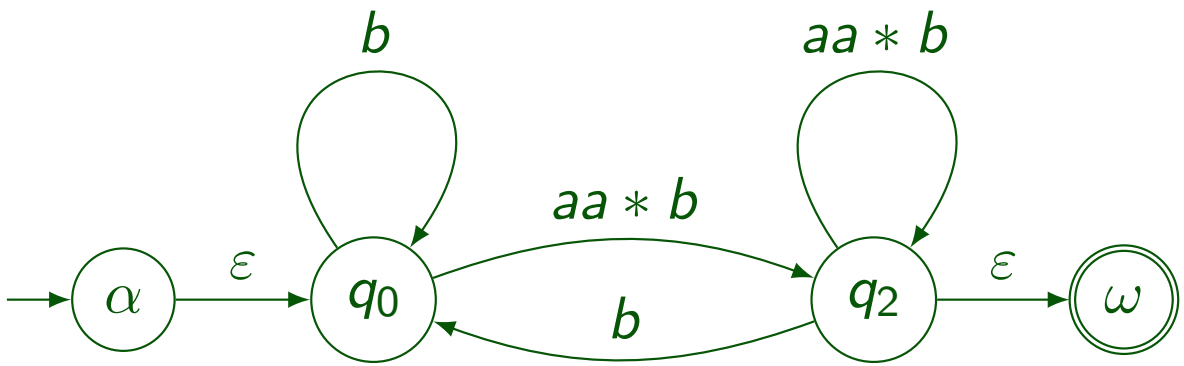
Exemple

On ajoute les états α et ω.



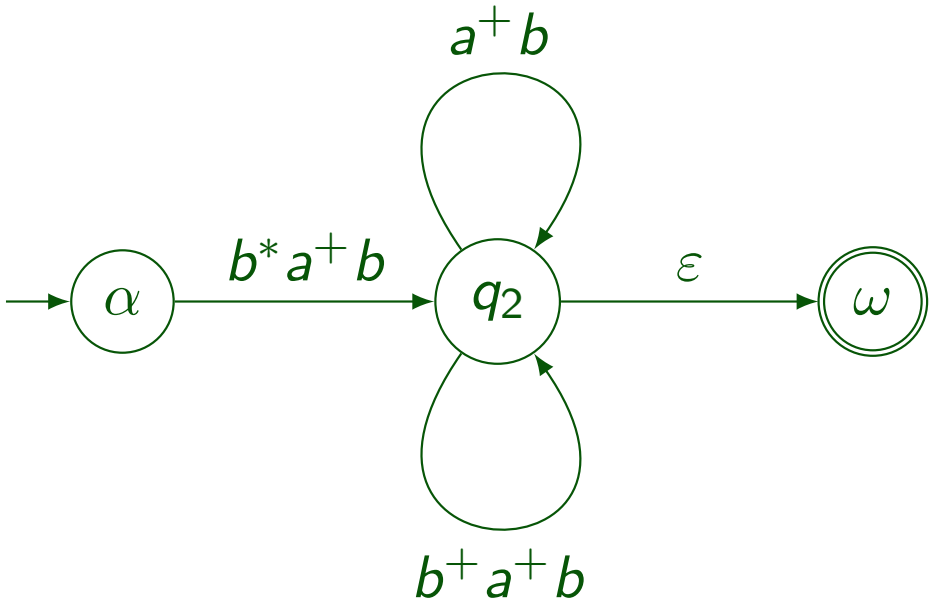
Exemple

On supprime alors l'état q_1



Exemple

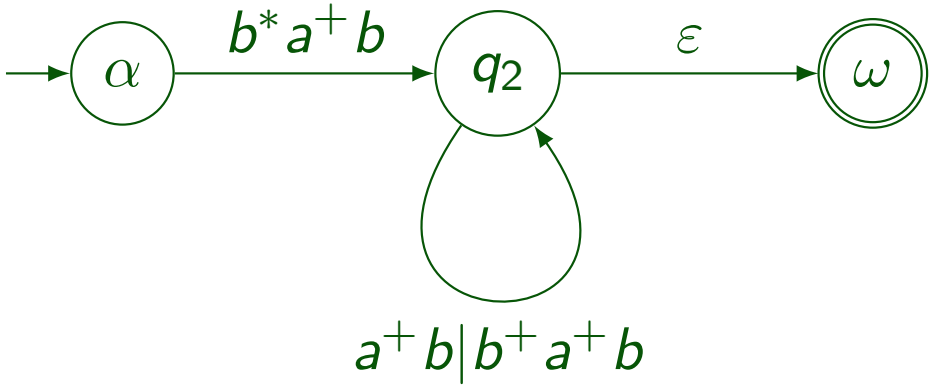
On supprime l'état q_0 :



On rappelle que l'expression rationnelle a^+ désigne $aa\star$.

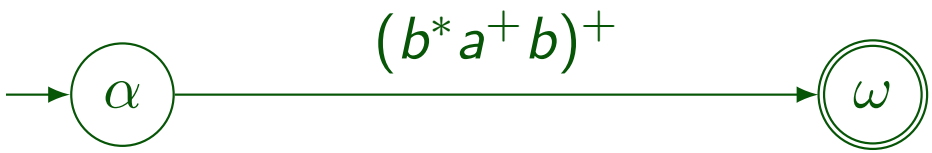
Exemple

On supprime la double flèche :



Exemple

On remarque que $a^+b|b^+a^+b = b^*a^+b$, puis on supprime q_2.



L'expression rationnelle est donc $(b^*a^+b)^+$.