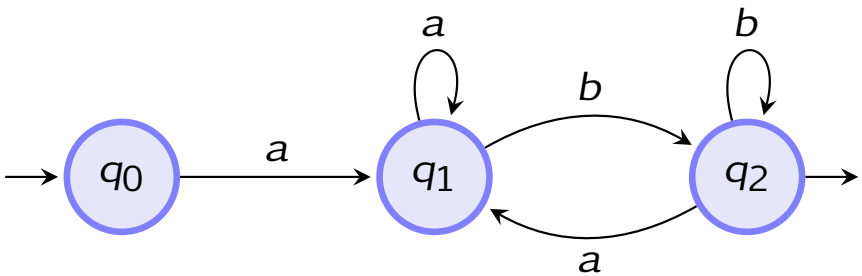
Automates

Jean-Pierre Becirspahic
Lycée Louis-Le-Grand

Automates finis déterministes

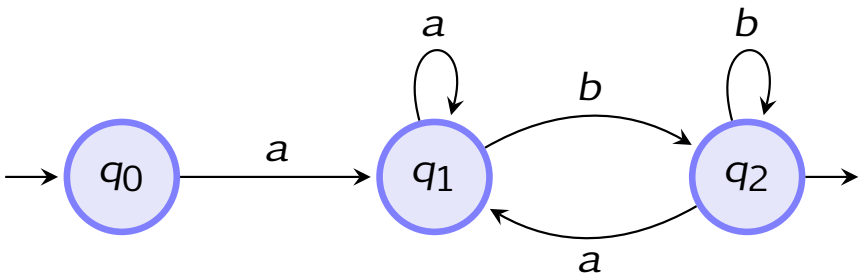
Un automate est une machine abstraite qui peut prendre un nombre fini d'états, qui reçoit en entrée un mot écrit sur un alphabet Σ, et qui change d'état à la lecture des lettres de ce dernier.
Certains états sont acceptants ; à la fin de la lecture du mot passé en entrée l'automate aura changé d'état ; on dira que le mot est *accepté* si l'état final est un état acceptant, et *rejeté* dans le cas contraire.



L'état initial q_0 est désigné par une flèche entrante ; les états acceptants (ici q_2) sont représentés par une flèche sortante.

Automates finis déterministes

Un automate est une machine abstraite qui peut prendre un nombre fini d'états, qui reçoit en entrée un mot écrit sur un alphabet Σ, et qui change d'état à la lecture des lettres de ce dernier.
Certains états sont acceptants ; à la fin de la lecture du mot passé en entrée l'automate aura changé d'état ; on dira que le mot est *accepté* si l'état final est un état acceptant, et *rejeté* dans le cas contraire.



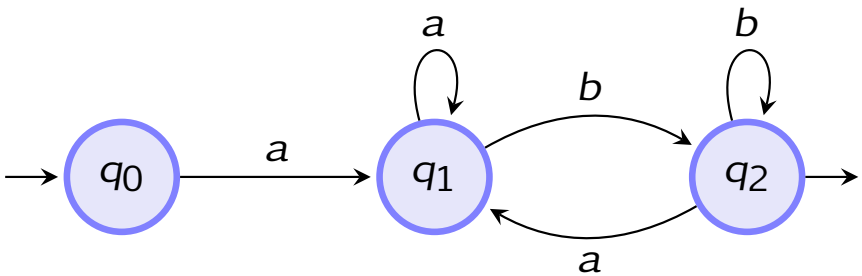
Pour qu'un mot soit lu et aboutisse à un état acceptant il faut et il suffit qu'il débute par un a et se termine par un b ; on dira que cet automate reconnait le langage des mots de $a\Sigma^*b$.

La lecture du mot abbaabab fait passer l'automate par les états :

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{b} q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1 \xrightarrow{b} q_2.$$

Automates finis déterministes

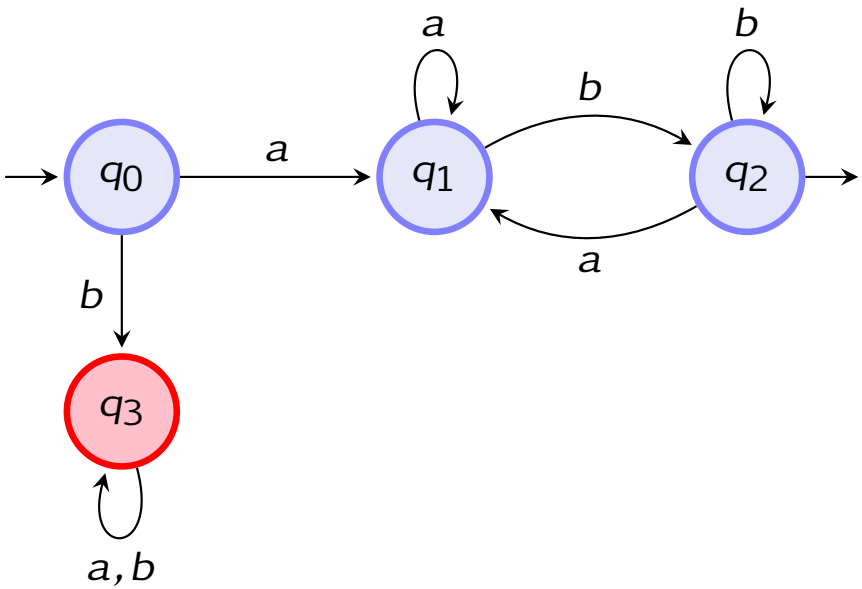
Un automate est une machine abstraite qui peut prendre un nombre fini d'états, qui reçoit en entrée un mot écrit sur un alphabet Σ, et qui change d'état à la lecture des lettres de ce dernier.
Certains états sont acceptants ; à la fin de la lecture du mot passé en entrée l'automate aura changé d'état ; on dira que le mot est *accepté* si l'état final est un état acceptant, et *rejeté* dans le cas contraire.



Le couple (q_0, b) est un blocage de l'automate. Un automate sans blocage est dit complet.

Automates finis déterministes

Un automate est une machine abstraite qui peut prendre un nombre fini d'états, qui reçoit en entrée un mot écrit sur un alphabet Σ, et qui change d'état à la lecture des lettres de ce dernier.
Certains états sont acceptants ; à la fin de la lecture du mot passé en entrée l'automate aura changé d'état ; on dira que le mot est *accepté* si l'état final est un état acceptant, et *rejeté* dans le cas contraire.

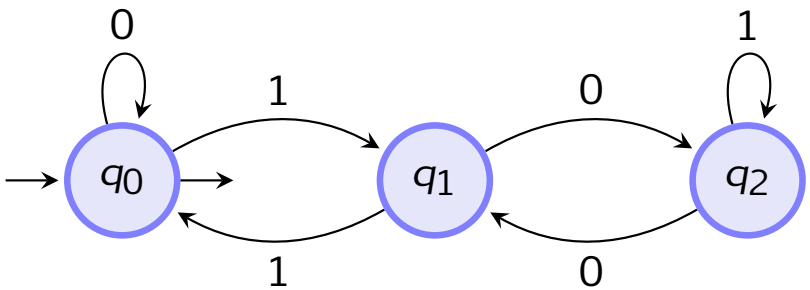


Il est toujours possible de rendre un automate complet en ajoutant un puit.

Automates finis déterministes

Un second exemple

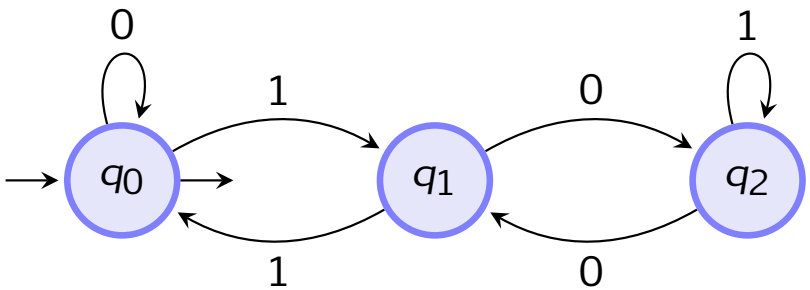
$\Sigma = \{0, 1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :



Automates finis déterministes

Un second exemple

$\Sigma = \{0, 1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :

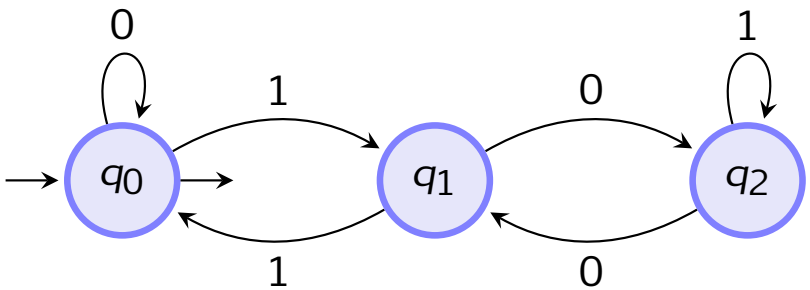


L'état final atteint est q_0 si $n \equiv 0 \bmod 3$, q_1 si $n \equiv 1 \bmod 3$ et q_2 si $n \equiv 2 \bmod 3$.

Automates finis déterministes

Un second exemple

$\Sigma = \{0, 1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :



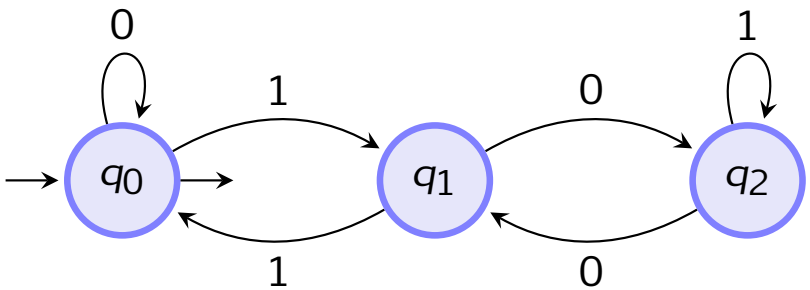
L'état final atteint est q_0 si $n \equiv 0 \bmod 3$, q_1 si $n \equiv 1 \bmod 3$ et q_2 si $n \equiv 2 \bmod 3$.

- C'est vrai si $n = 0$;

Automates finis déterministes

Un second exemple

$\Sigma = \{0,1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :



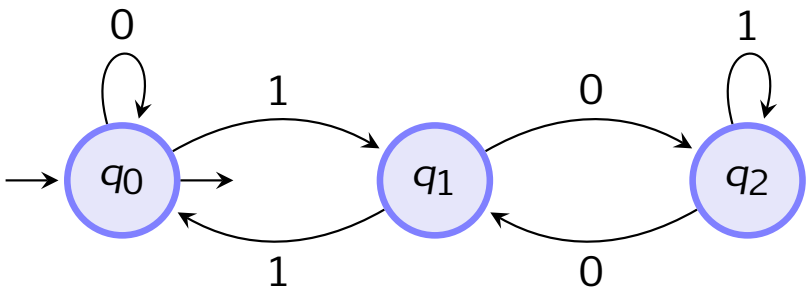
L'état final atteint est q_0 si $n \equiv 0 \bmod 3$, q_1 si $n \equiv 1 \bmod 3$ et q_2 si $n \equiv 2 \bmod 3$.

- C'est vrai si $n = 0$;
- si $n \geqslant 1$ on pose $n = 2p + r$ avec $r \in \{0,1\}$.
  Avant la lecture de r l'automate se trouve dans l'état q_i avec $p \equiv i \bmod 3$.

Automates finis déterministes

Un second exemple

$\Sigma = \{0, 1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :



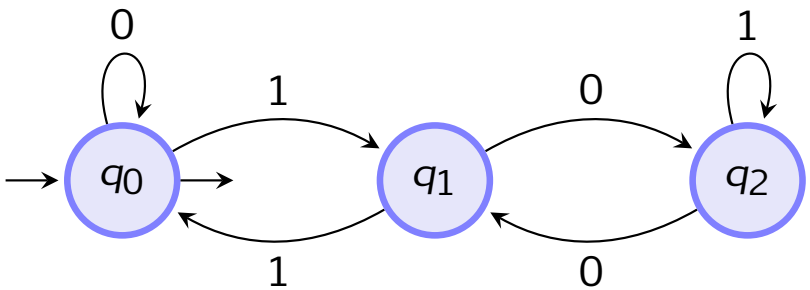
L'état final atteint est q_0 si $n \equiv 0 \bmod 3$, q_1 si $n \equiv 1 \bmod 3$ et q_2 si $n \equiv 2 \bmod 3$.

- C'est vrai si $n = 0$;
- si $n \geqslant 1$ on pose $n = 2p + r$ avec $r \in \{0, 1\}$.
  Avant la lecture de r l'automate se trouve dans l'état q_i avec $p \equiv i \bmod 3$.
  - Si $i = 0$ alors $n \equiv r \bmod 3$ et si $r = 0$ l'automate reste à l'état q_0, si $r = 1$ l'automate passe à l'état q_1 ;

Automates finis déterministes

Un second exemple

$\Sigma = \{0,1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :



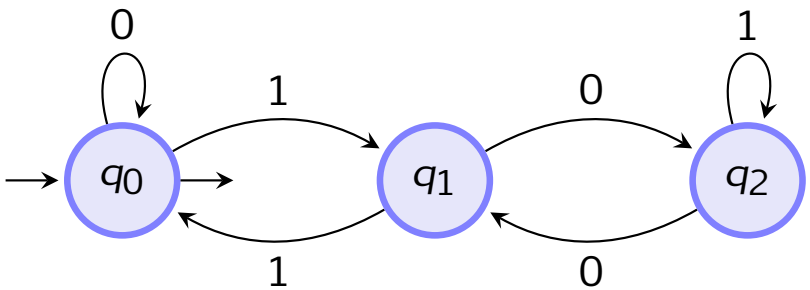
L'état final atteint est q_0 si $n \equiv 0 \bmod 3$, q_1 si $n \equiv 1 \bmod 3$ et q_2 si $n \equiv 2 \bmod 3$.

- C'est vrai si $n = 0$;
- si $n \geqslant 1$ on pose $n = 2p + r$ avec $r \in \{0,1\}$.
  Avant la lecture de r l'automate se trouve dans l'état q_i avec $p \equiv i \bmod 3$.
  - Si $i = 0$ alors $n \equiv r \bmod 3$ et si $r = 0$ l'automate reste à l'état q_0, si $r = 1$ l'automate passe à l'état q_1 ;
  - si $i = 1$ alors $n \equiv 2 + r \bmod 3$ et si $r = 0$ l'automate passe à l'état q_2, si $r = 1$ l'automate passe à l'état q_0 ;

Automates finis déterministes

Un second exemple

$\Sigma = \{0,1\}$; le mot lu par l'automate correspond à l'écriture binaire d'un entier n. Cet automate reconnait les entiers n qui sont divisibles par 3 :



L'état final atteint est q_0 si $n \equiv 0 \bmod 3$, q_1 si $n \equiv 1 \bmod 3$ et q_2 si $n \equiv 2 \bmod 3$.

- C'est vrai si $n = 0$;
- si $n \geqslant 1$ on pose $n = 2p + r$ avec $r \in \{0,1\}$.
  Avant la lecture de r l'automate se trouve dans l'état q_i avec $p \equiv i \bmod 3$.
  - Si $i = 0$ alors $n \equiv r \bmod 3$ et si $r = 0$ l'automate reste à l'état q_0, si $r = 1$ l'automate passe à l'état q_1 ;
  - si $i = 1$ alors $n \equiv 2 + r \bmod 3$ et si $r = 0$ l'automate passe à l'état q_2, si $r = 1$ l'automate passe à l'état q_0 ;
  - si $i = 2$ alors $n \equiv 1 + r \bmod 3$ et si $r = 0$ l'automate passe à l'état q_1, si $r = 1$ l'automate reste à l'état q_2.

Définition formelle d'un automate

Un automate à états finis déterministe est défini par $A = (\Sigma, Q, q_0, F, \delta)$.

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'*états* de A ;
- $q_0 \in Q$ est l'état *initial* ;
- $F \subset Q$ est l'ensemble des états *acceptants* (ou finaux) ;
- δ est une application d'une partie de $Q \times \Sigma$ dans Q, appelée *fonction de transition*.

Lorsque δ est définie sur $Q \times \Sigma$ tout entier, l'automate A est dit complet.

Définition formelle d'un automate

Un automate à états finis déterministe est défini par $A = (\Sigma, Q, q_0, F, \delta)$.

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'*états* de A ;
- $q_0 \in Q$ est l'état *initial* ;
- $F \subset Q$ est l'ensemble des états *acceptants* (ou finaux) ;
- δ est une application d'une partie de $Q \times \Sigma$ dans Q, appelée *fonction de transition*.

Lorsque δ est définie sur $Q \times \Sigma$ tout entier, l'automate A est dit complet.

Une transition $\delta(q_i, a) = q_j$ est représentée par $q_i \xrightarrow{a} q_j$. Un chemin dans A est une suite finie de transitions consécutives $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ débutant par l'état initial q_0. Le mot $a_1 a_2 \cdots a_n$ est appelé l'étiquette du chemin.

Définition formelle d'un automate

Un automate à états finis déterministe est défini par $A = (\Sigma, Q, q_0, F, \delta)$.

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'*états* de A ;
- $q_0 \in Q$ est l'état *initial* ;
- $F \subset Q$ est l'ensemble des états *acceptants* (ou finaux) ;
- δ est une application d'une partie de $Q \times \Sigma$ dans Q, appelée *fonction de transition*.

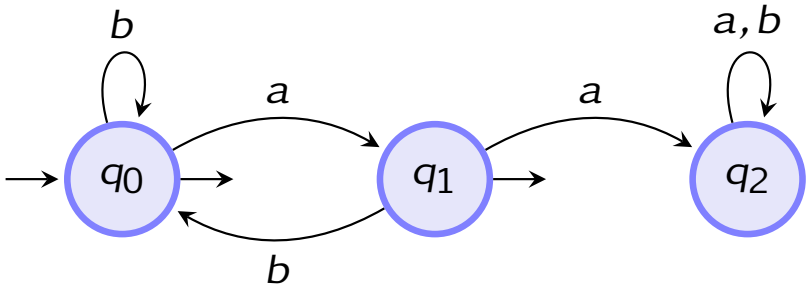
Lorsque δ est définie sur $Q \times \Sigma$ tout entier, l'automate A est dit complet.
Une transition $\delta(q_i, a) = q_j$ est représentée par $q_i \xrightarrow{a} q_j$. Un chemin dans A est une suite finie de transitions consécutives $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ débutant par l'état initial q_0. Le mot $a_1 a_2 \cdots a_n$ est appelé l'étiquette du chemin.
Un chemin est acceptant lorsque l'état d'arrivée est un état acceptant ; un mot de Σ^* est reconnu par A lorsqu'il est l'étiquette d'un chemin acceptant. Le langage $L(A)$ reconnu par A est l'ensemble des mots reconnus par A.

Définition formelle d'un automate

Exemple

- $\Sigma = \{a, b\}$;
- $Q = \{q_0, q_1, q_2\}$;
- $F = \{q_0, q_1\}$;

δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2



On note L_i l'ensemble des étiquettes des chemins débutant par q_i et finissant par un état acceptant.

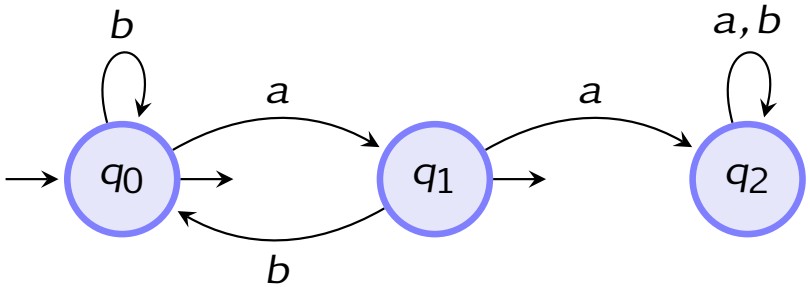
$$L_0 = \varepsilon + bL_0 + aL_1, \qquad L_1 = \varepsilon + bL_0 \quad \text{et} \quad L_2 = \emptyset.$$

Définition formelle d'un automate

Exemple

- $\Sigma = \{a, b\}$;
- $Q = \{q_0, q_1, q_2\}$;
- $F = \{q_0, q_1\}$;

δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2



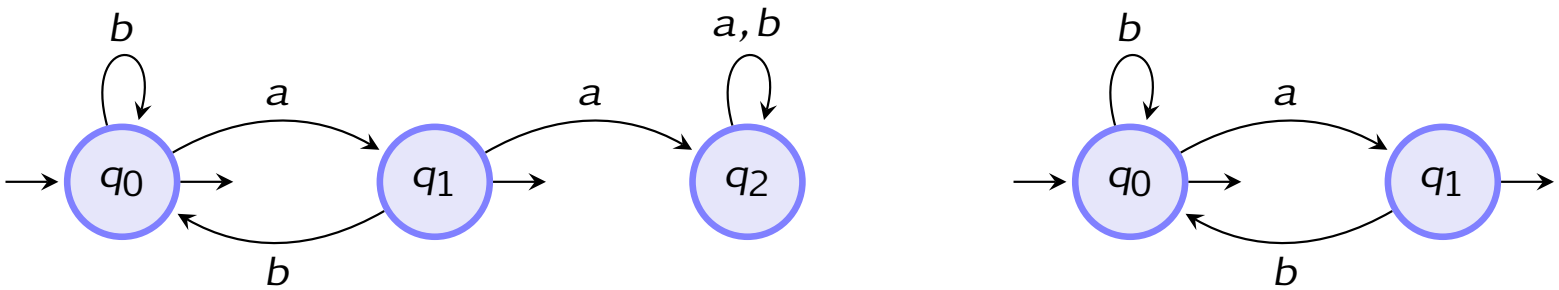
On note L_i l'ensemble des étiquettes des chemins débutant par q_i et finissant par un état acceptant.

$$L_0 = \varepsilon + bL_0 + aL_1, \qquad L_1 = \varepsilon + bL_0 \quad \text{et} \quad L_2 = \emptyset.$$

On en déduit que $L_0 = \varepsilon + bL_0 + a(\varepsilon + bL_0) = (b + ab)L_0 + (\varepsilon + a)$. D'après le lemme d'Arden, $L_0 = (b+ab)^*(\varepsilon+a)$. Le langage reconnu par cet automate est le langage des mots qui ne comportent pas deux a consécutifs.

Émondage

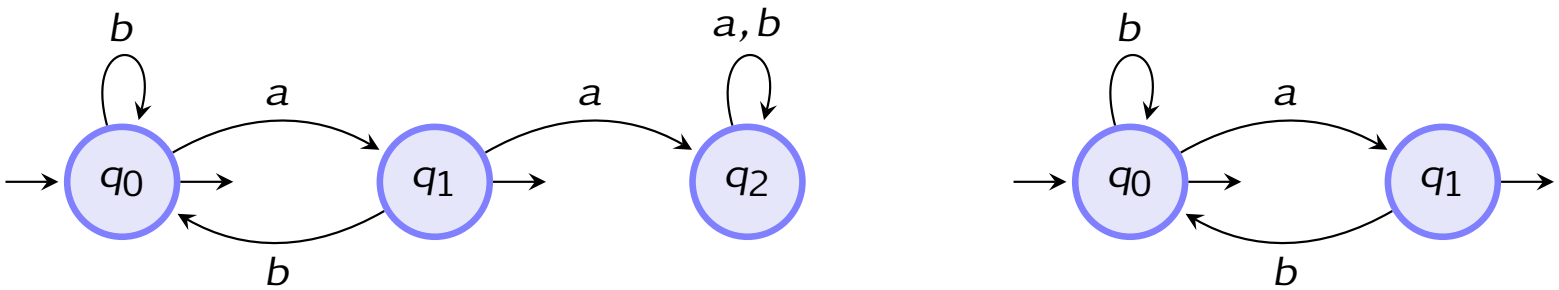
Deux automates sont dits équivalents lorsqu'ils reconnaissent le même langage :



Le premier est complet, le second ne l'est pas.

Émondage

Deux automates sont dits équivalents lorsqu'ils reconnaissent le même langage :

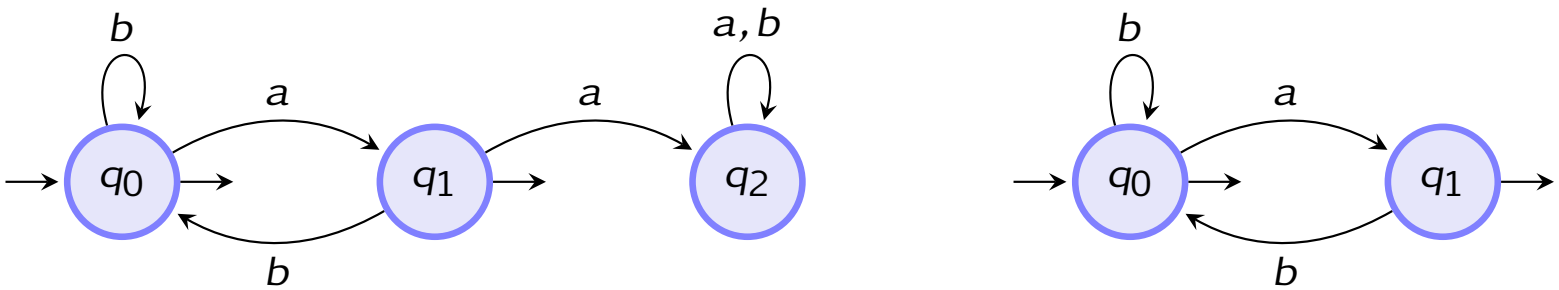


un état q qu'il est :

- accessible lorsqu'il existe un chemin menant de l'état initial q_0 à q ;
- co-accessible lorsqu'il existe un chemin menant de q à un état acceptant.

Émondage

Deux automates sont dits équivalents lorsqu'ils reconnaissent le même langage :



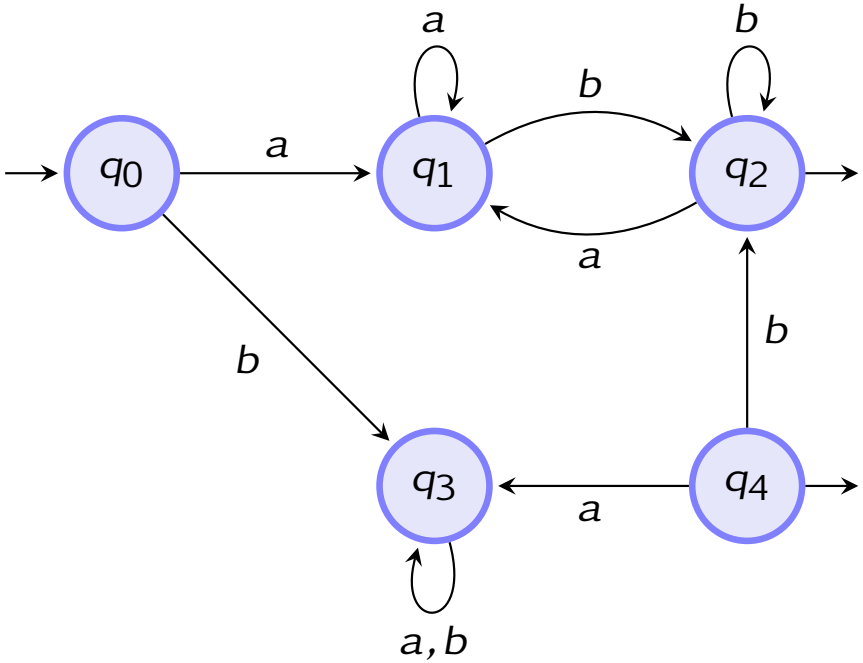
un état q qu'il est :

- accessible lorsqu'il existe un chemin menant de l'état initial q_0 à q ;
- co-accessible lorsqu'il existe un chemin menant de q à un état acceptant.

Un état accessible et co-accessible est dit utile. On obtient un automate équivalent en supprimant tous les états inutiles ainsi que les transitions qui les concernent.

Émondage

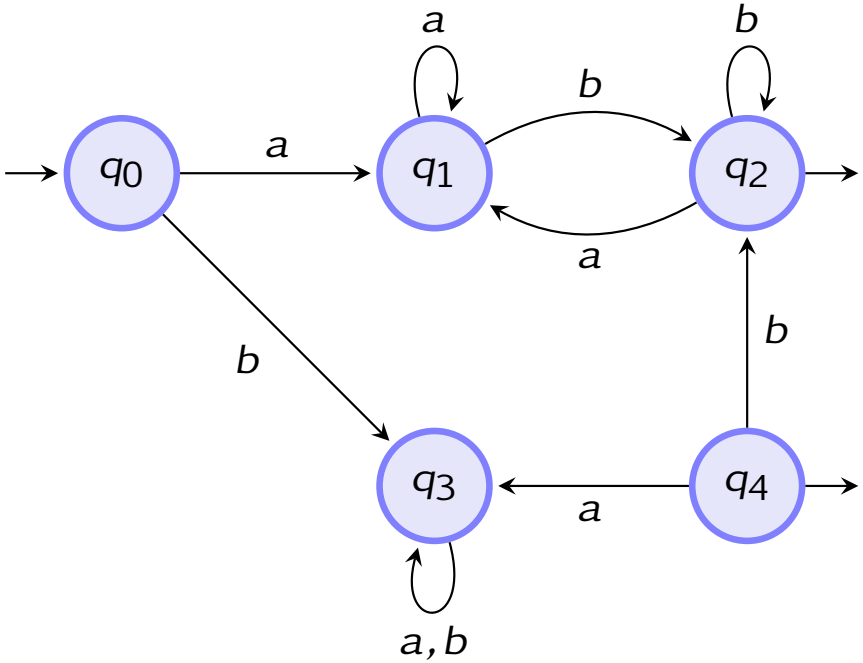
Exemple



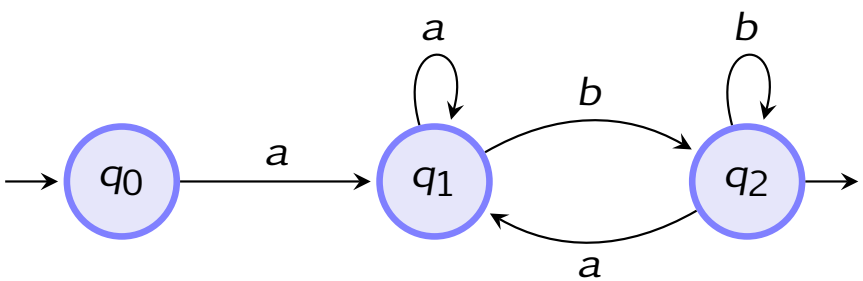
L'automate ci-dessus possède deux états inutiles : q_4 n'est pas accessible et q_3 n'est pas co-accessible.

Émondage

Exemple



L'automate ci-dessus possède deux états inutiles : q_4 n'est pas accessible et q_3 n'est pas co-accessible. Il est équivalent à :



et reconnait le langage dénoté par $a(a+b)^*b$.

Mise en œuvre

On utilise le type *char* pour représenter l'alphabet Σ, le type *int* pour l'ensemble des états et un dictionnaire pour énumérer la liste des transitions possibles. Pour simplifier, ce dictionnaire sera représenté par le type *(int * char) * int list* .

Mise en œuvre

On utilise le type *char* pour représenter l'alphabet Σ, le type *int* pour l'ensemble des états et un dictionnaire pour énumérer la liste des transitions possibles. Pour simplifier, ce dictionnaire sera représenté par le type *(int * char) * int list* .

```
type dfa = {Start: int ;
            Accept: int list ;
            Delta: ((int * char) * int) list} ;;

let chemin a m =
  let rec aux q = function
    | k when k = string_length m -> q
    | k                          -> aux (assoc (q, m.[k]) a.Delta) (k+1)
  in aux a.Start 0 ;;

let reconnu a m =
  try mem (chemin a m) a.Accept
  with Not_found -> false ;;
```

Automates non déterministes

Un NFA est défini par un quintuplet $A = (\Sigma, Q, I, F, \delta)$ où :

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'états de A ;
- $I \subset Q$ est l'ensemble des états initiaux ;
- $F \subset Q$ est l'ensemble des états acceptants ;
- δ est une application d'une partie de $Q \times \Sigma$ dans $\mathscr{P}(Q)$: la fonction de transition.

Automates non déterministes

Un NFA est défini par un quintuplet $A = (\Sigma, Q, I, F, \delta)$ où :

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'états de A ;
- $I \subset Q$ est l'ensemble des états initiaux ;
- $F \subset Q$ est l'ensemble des états acceptants ;
- δ est une application d'une partie de $Q \times \Sigma$ dans $\mathscr{P}(Q)$: la fonction de transition.

Une transition est un triplet (q_i, a, q_j) tel que $q_j \in \delta(q_i, a)$, représentée par $q_i \xrightarrow{a} q_j$. Un chemin est une suite finie de transitions consécutives $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$.

Automates non déterministes

Un NFA est défini par un quintuplet $A = (\Sigma, Q, I, F, \delta)$ où :

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'états de A ;
- $I \subset Q$ est l'ensemble des états initiaux ;
- $F \subset Q$ est l'ensemble des états acceptants ;
- δ est une application d'une partie de $Q \times \Sigma$ dans $\mathscr{P}(Q)$: la fonction de transition.

Une transition est un triplet (q_i, a, q_j) tel que $q_j \in \delta(q_i, a)$, représentée par $q_i \xrightarrow{a} q_j$. Un chemin est une suite finie de transitions consécutives $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$.

Un mot de Σ^* est reconnu par l'automate A s'il étiquette un chemin menant d'un état initial à un état acceptant. Le langage des mots reconnus par l'automate A est noté $L(A)$.

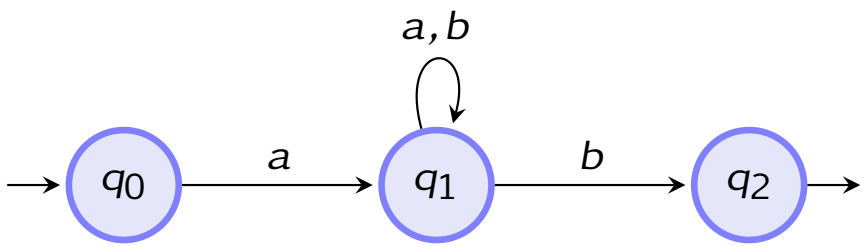
Automates non déterministes

Un NFA est défini par un quintuplet $A = (\Sigma, Q, I, F, \delta)$ où :

- Σ est un alphabet (fini) ;
- Q est un ensemble fini d'états de A ;
- $I \subset Q$ est l'ensemble des états initiaux ;
- $F \subset Q$ est l'ensemble des états acceptants ;
- δ est une application d'une partie de $Q \times \Sigma$ dans $\mathscr{P}(Q)$: la fonction de transition.

Exemple : l'automate ci dessous reconnait le langage dénoté par

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



Déterminisation

Considérons donc un automate non-déterministe $A = (\Sigma, Q, I, F, \delta)$ et posons $A' = (\Sigma, \mathscr{P}(Q), I, F', \delta')$ avec :

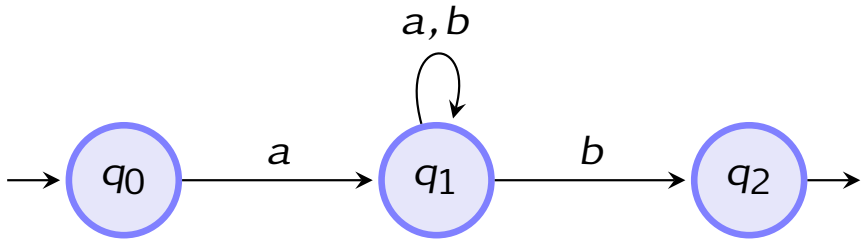
$$F' = \left\{ P \in \mathscr{P}(Q) \;\middle|\; P \cap F \neq \emptyset \right\} \quad \text{et} \quad \forall (P, a) \in \mathscr{P}(Q) \times \Sigma,\ \delta'(P, a) = \bigcup_{q \in P} \delta(q, a).$$

Si A est un automate non-déterministe à n états, A' est un automate déterministe à 2^n états.

Déterminisation

Considérons donc un automate non-déterministe $A = (\Sigma, Q, I, F, \delta)$ et posons $A' = (\Sigma, \mathscr{P}(Q), I, F', \delta')$ avec :

$$F' = \left\{ P \in \mathscr{P}(Q) \;\middle|\; P \cap F \neq \emptyset \right\} \quad \text{et} \quad \forall (P, a) \in \mathscr{P}(Q) \times \Sigma,\ \delta'(P, a) = \bigcup_{q \in P} \delta(q, a).$$



États et transitions de A' :

	δ'	a	b
	$\emptyset$	$\emptyset$	$\emptyset$
*	$\{q_0\}$	$\{q_1\}$	$\emptyset$
	$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$
	$\{q_2\}$	$\emptyset$	$\emptyset$

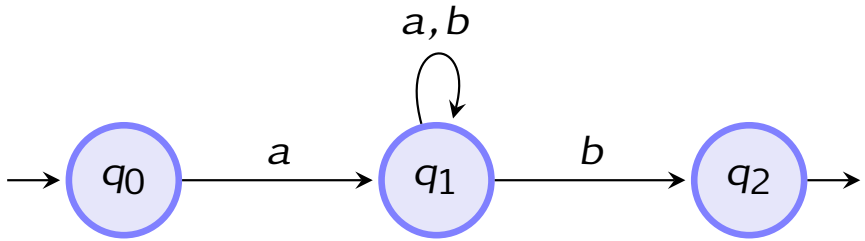
	δ'	a	b
	$\{q_0, q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$
*	$\{q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$
*	$\{q_0, q_2\}$	$\{q_1\}$	$\emptyset$
*	$\{q_0, q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$

* : état initial, * : états acceptants.

Déterminisation

Considérons donc un automate non-déterministe $A = (\Sigma, Q, I, F, \delta)$ et posons $A' = (\Sigma, \mathscr{P}(Q), I, F', \delta')$ avec :

$$F' = \left\{ P \in \mathscr{P}(Q) \;\middle|\; P \cap F \neq \emptyset \right\} \quad \text{et} \quad \forall (P, a) \in \mathscr{P}(Q) \times \Sigma,\ \delta'(P, a) = \bigcup_{q \in P} \delta(q, a).$$



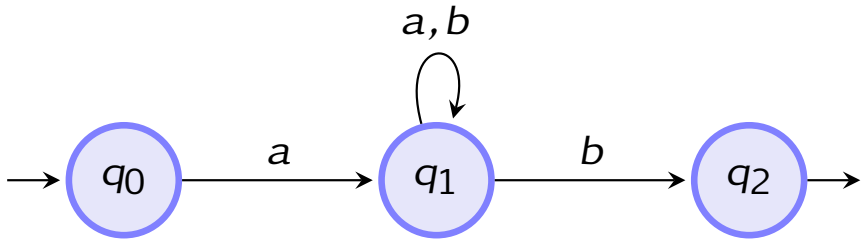
Dans la pratique on écrit que les états accessibles :

	δ'	a	b	
*	$\{q_0\}$	$\{q_1\}$	–	$\to q'_0$
	$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$	$\to q'_1$
*	$\{q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$	$\to q'_2$

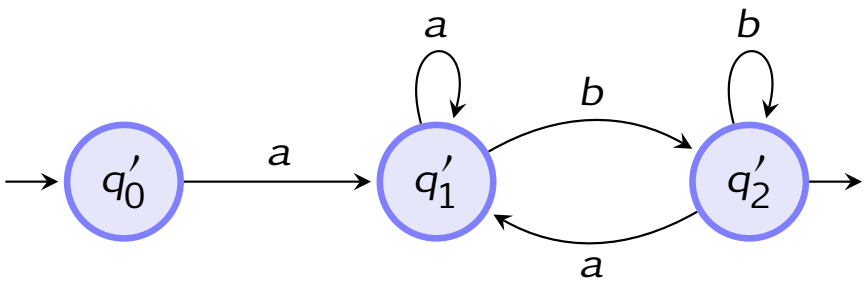
Déterminisation

Considérons donc un automate non-déterministe $A = (\Sigma, Q, I, F, \delta)$ et posons $A' = (\Sigma, \mathscr{P}(Q), I, F', \delta')$ avec :

$$F' = \left\{ P \in \mathscr{P}(Q) \;\middle|\; P \cap F \neq \emptyset \right\} \quad \text{et} \quad \forall (P, a) \in \mathscr{P}(Q) \times \Sigma,\ \delta'(P, a) = \bigcup_{q \in P} \delta(q, a).$$



On obtient l'automate déterminisé suivant :



A' reconnait lui aussi le langage dénoté par $a(a+b)^*b$.

Déterminisation

Les automates A et A' reconnaissent le même langage.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Considérons un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ dans A.
Il existe dans A' un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} P_{n-1}$ tel que $q_{n-1} \in P_{n-1}$.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Considérons un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ dans A.
Il existe dans A' un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} P_{n-1}$ tel que $q_{n-1} \in P_{n-1}$.

$q_n \in \delta(q_{n-1}, a_n)$ donc $q_n \in \delta'(P_{n-1}, a_n)$. En posant $P_n = \delta'(P_{n-1}, a_n)$ on établit un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$ tel que $q_n \in P_n$.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Réciproquement, considérons un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$ dans A, et considérons un élément $q_n \in P_n$.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Réciproquement, considérons un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$ dans A, et considérons un élément $q_n \in P_n$.
Il existe un état $q_{n-1} \in P_{n-1}$ tel que $q_n \in \delta(q_{n-1}, a_n)$, et par hypothèse de récurrence il existe un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_{n-1}$ dans A.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Réciproquement, considérons un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$ dans A, et considérons un élément $q_n \in P_n$.
Il existe un état $q_{n-1} \in P_{n-1}$ tel que $q_n \in \delta(q_{n-1}, a_n)$, et par hypothèse de récurrence il existe un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_{n-1}$ dans A.
Ceci prouve l'existence d'un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$.

Déterminisation

Les automates A et A' reconnaissent le même langage.

On montre par récurrence sur $|u|$ que pour tout mot $u \in \Sigma^*$, il existe dans A un chemin étiqueté par u menant à un état q si et seulement s'il existe dans A' un chemin étiqueté par u menant à un état P contenant q.

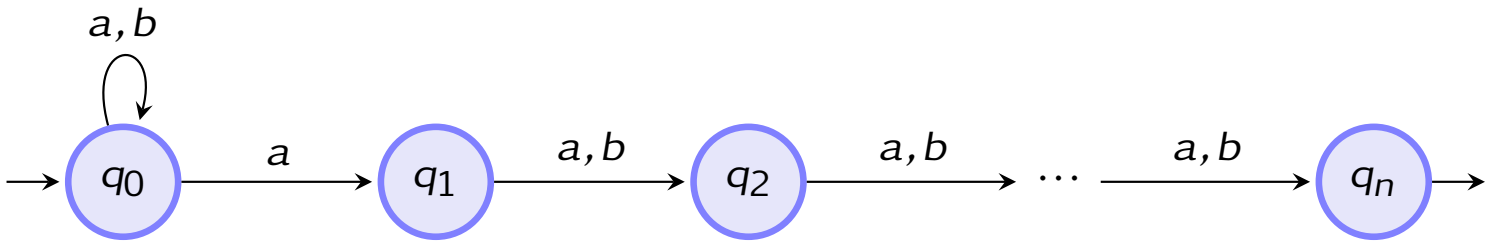
- Dans A tout chemin étiqueté par ε mène à un élément de I ; dans A' tout chemin étiqueté par ε mène à l'état I.
- Si $u \neq \varepsilon$, supposons le résultat acquis pour tout mot de longueur strictement inférieure, et posons $u = a_1 a_2 \cdots a_n$.

Réciproquement, considérons un chemin $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$ dans A, et considérons un élément $q_n \in P_n$.
Il existe un état $q_{n-1} \in P_{n-1}$ tel que $q_n \in \delta(q_{n-1}, a_n)$, et par hypothèse de récurrence il existe un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_{n-1}$ dans A.
Ceci prouve l'existence d'un chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$.

Sachant que $F' = \left\{ P \in \mathscr{P}(Q) \;\middle|\; P \cap F \neq \emptyset \right\}$ ceci prouve qu'un mot est reconnu par A si et seulement s'il est reconnu par A'.

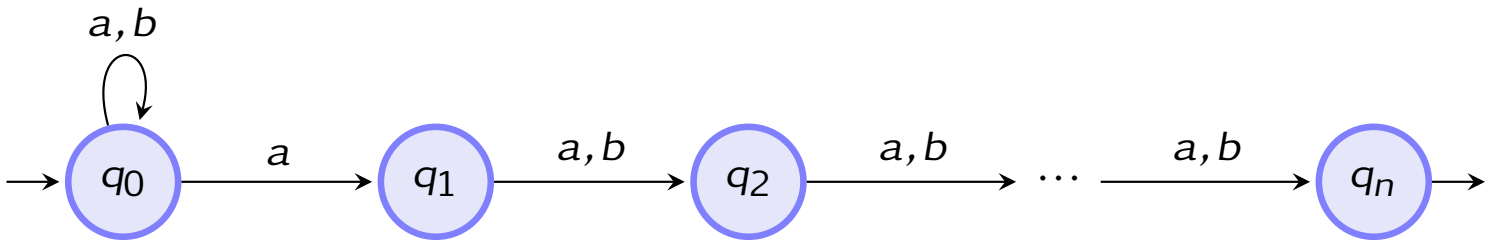
Un exemple de coût exponentiel

Considérons le langage L dénoté par $(a+b)^*a(a+b)^{n-1}$. Il est facile d'obtenir un automate non-déterministe à $n+1$ états qui le reconnait :



Un exemple de coût exponentiel

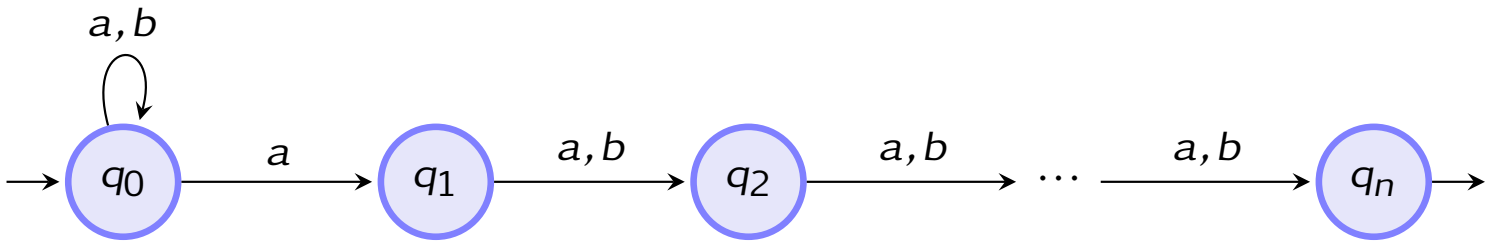
Considérons le langage L dénoté par $(a+b)^*a(a+b)^{n-1}$. Il est facile d'obtenir un automate non-déterministe à $n+1$ états qui le reconnait :



Tout automate déterministe qui reconnait L possède au moins 2^n états.

Un exemple de coût exponentiel

Considérons le langage L dénoté par $(a+b)^*a(a+b)^{n-1}$. Il est facile d'obtenir un automate non-déterministe à $n+1$ états qui le reconnait :

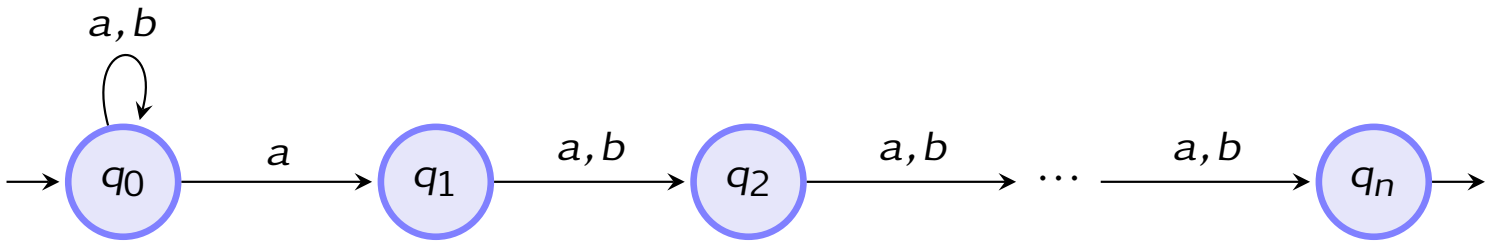


Tout automate déterministe qui reconnait L possède au moins 2^n états.

Considérons un tel automate A et notons q_0 son état initial. À tout mot u de n lettres on associe l'état $q(u)$ auquel aboutit le chemin étiqueté par u. On définit ainsi une application $q : \{a,b\}^n \to Q$.

Un exemple de coût exponentiel

Considérons le langage L dénoté par $(a+b)^* a (a+b)^{n-1}$. Il est facile d'obtenir un automate non-déterministe à $n+1$ états qui le reconnait :

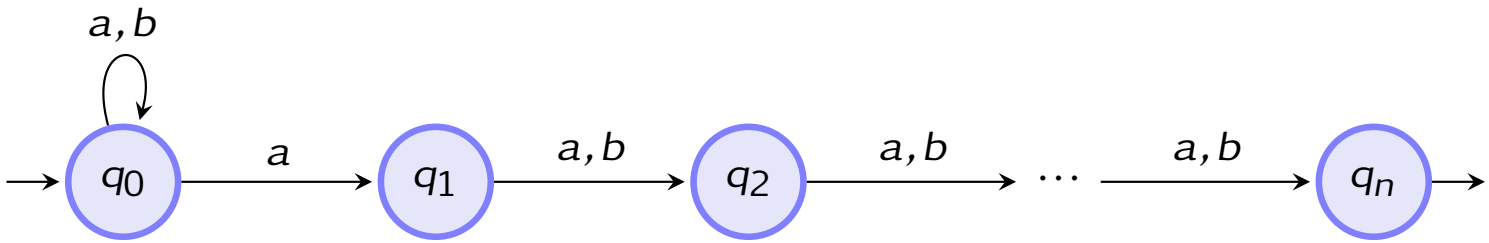


Tout automate déterministe qui reconnait L possède au moins 2^n états.

Considérons un tel automate A et notons q_0 son état initial. À tout mot u de n lettres on associe l'état $q(u)$ auquel aboutit le chemin étiqueté par u. On définit ainsi une application $q : \{a,b\}^n \to Q$.
On considère deux mots $u \neq v$ tels que $q(u) = q(v)$ et le plus long suffixe w commun à u et à v. Sans perte de généralité on peut poser $u = u'aw$ et $v = v'bw$.

Un exemple de coût exponentiel

Considérons le langage L dénoté par $(a+b)^*a(a+b)^{n-1}$. Il est facile d'obtenir un automate non-déterministe à $n+1$ états qui le reconnait :

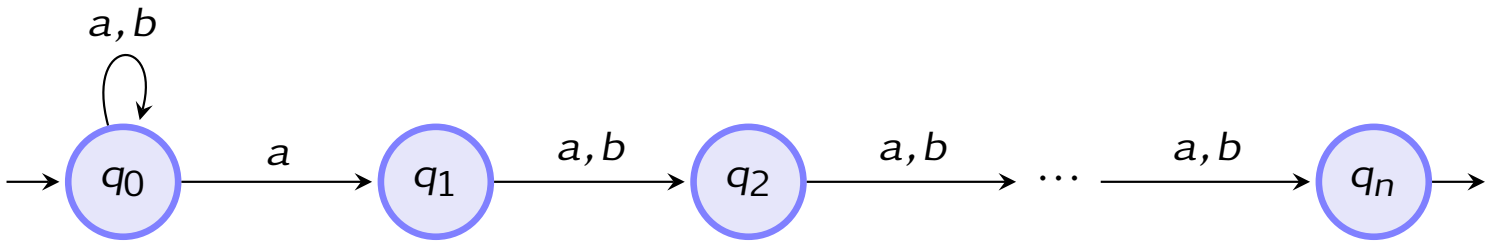


Tout automate déterministe qui reconnait L possède au moins 2^n états.

Considérons un tel automate A et notons q_0 son état initial. À tout mot u de n lettres on associe l'état $q(u)$ auquel aboutit le chemin étiqueté par u. On définit ainsi une application $q : \{a,b\}^n \to Q$.
On considère deux mots $u \neq v$ tels que $q(u) = q(v)$ et le plus long suffixe w commun à u et à v. Sans perte de généralité on peut poser $u = u'aw$ et $v = v'bw$.
On complète w pour former un mot ww' de longueur $n-1$.
Le mot $uw' = u'aww'$ est reconnu par A mais pas $vw' = v'bww'$.

Un exemple de coût exponentiel

Considérons le langage L dénoté par $(a+b)^*a(a+b)^{n-1}$. Il est facile d'obtenir un automate non-déterministe à $n+1$ états qui le reconnait :



Tout automate déterministe qui reconnait L possède au moins 2^n états.

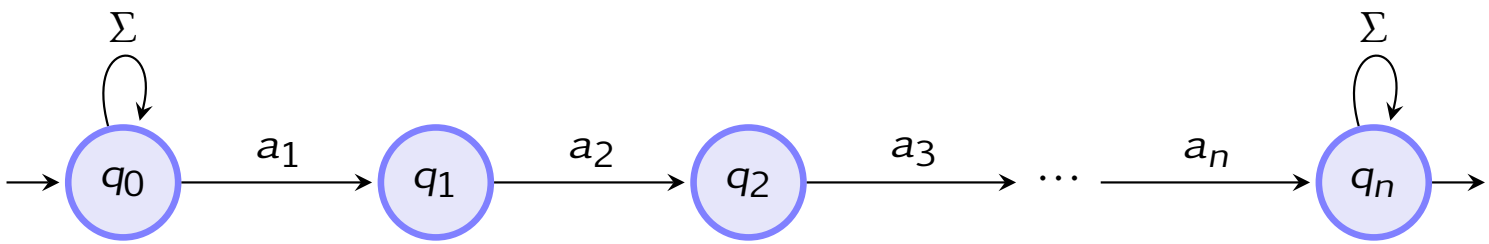
Considérons un tel automate A et notons q_0 son état initial. À tout mot u de n lettres on associe l'état $q(u)$ auquel aboutit le chemin étiqueté par u. On définit ainsi une application $q : \{a,b\}^n \to Q$.
On considère deux mots $u \neq v$ tels que $q(u) = q(v)$ et le plus long suffixe w commun à u et à v. Sans perte de généralité on peut poser $u = u'aw$ et $v = v'bw$.
On complète w pour former un mot ww' de longueur $n-1$.
Le mot $uw' = u'aww'$ est reconnu par A mais pas $vw' = v'bww'$.
Mais A est déterministe et $q(u) = q(v)$ donc les chemins étiquetés par uw' et vw' doivent mener au même état (ou être tous deux bloquants) ce qui est absurde.

Recherche d'un mot dans un texte

Si $u \in \Sigma^*$ est un mot donné, on souhaite un automate déterministe qui reconnait le langage $\Sigma^* u \Sigma^*$.

Recherche d'un mot dans un texte

Si $u \in \Sigma^*$ est un mot donné, on souhaite un automate déterministe qui reconnait le langage $\Sigma^* u \Sigma^*$.
Ce problème est aisément résolu à l'aide d'un automate non-déterministe : si $u = a_1 a_2 \cdots a_n$ il suffit de considérer :

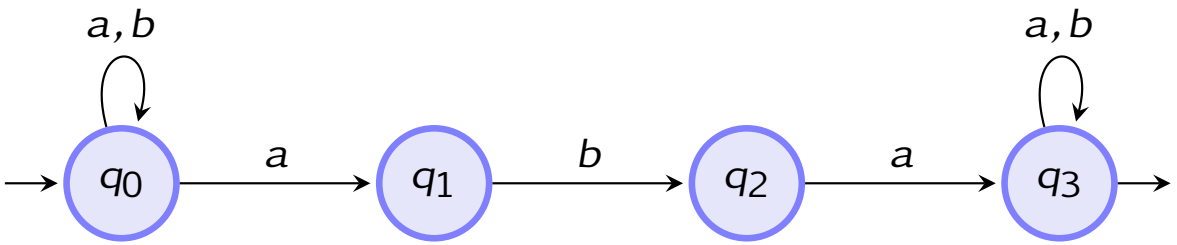


Il reste ensuite à déterminiser cet automate.

Recherche d'un mot dans un texte

Si $u \in \Sigma^*$ est un mot donné, on souhaite un automate déterministe qui reconnait le langage $\Sigma^* u \Sigma^*$.

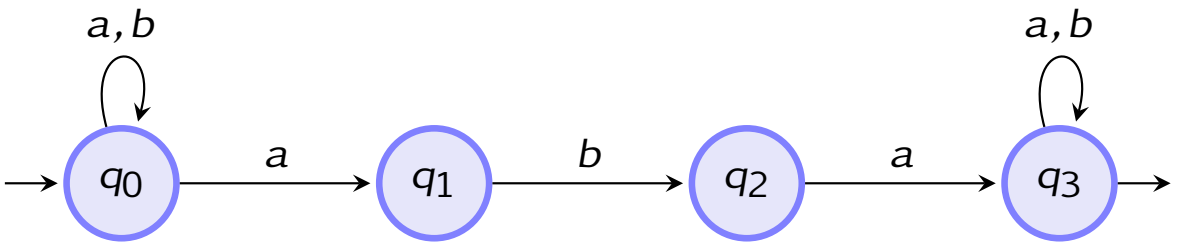
Exemple : recherche du mot aba sur l'alphabet $\{a, b\}$.



Recherche d'un mot dans un texte

Si $u \in \Sigma^*$ est un mot donné, on souhaite un automate déterministe qui reconnait le langage $\Sigma^* u \Sigma^*$.

Exemple : recherche du mot *aba* sur l'alphabet $\{a, b\}$.



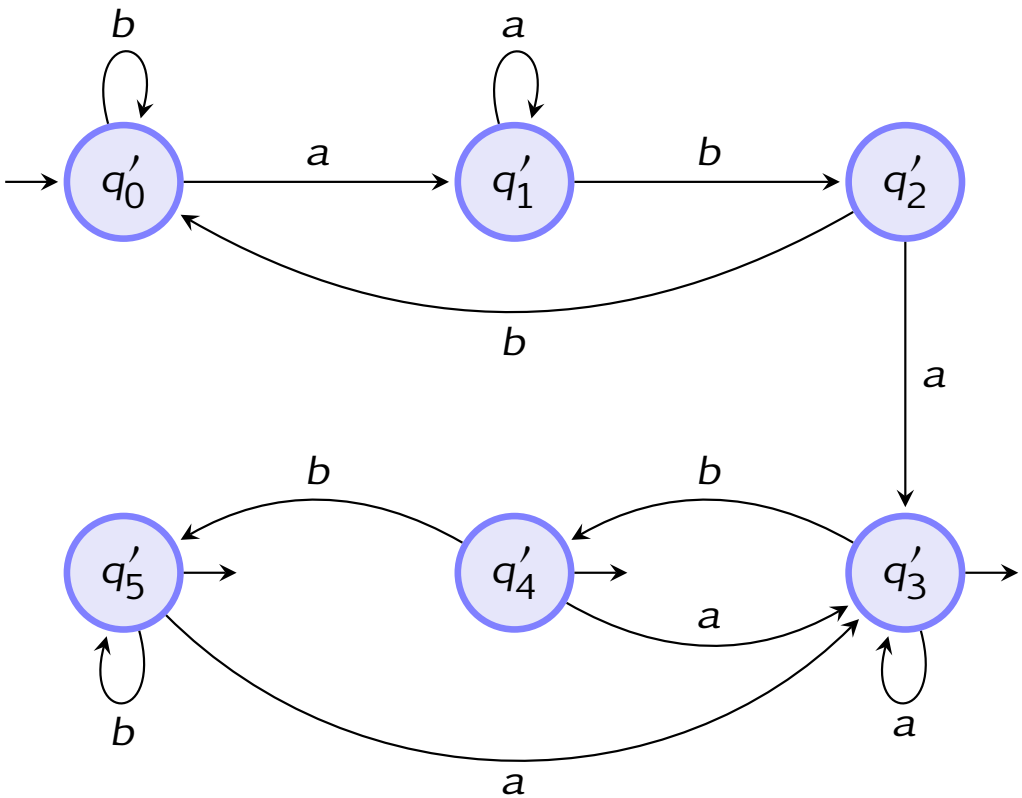
	δ'	a	b	
*	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$	$\to q'_0$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\to q'_1$
	$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$	$\to q'_2$
*	$\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_2, q_3\}$	$\to q'_3$
*	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$	$\to q'_4$
*	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$	$\to q'_5$

Recherche d'un mot dans un texte

Si $u \in \Sigma^*$ est un mot donné, on souhaite un automate déterministe qui reconnait le langage $\Sigma^* u \Sigma^*$.

Exemple : recherche du mot *aba* sur l'alphabet $\{a, b\}$.

On obtient l'automate déterminisé suivant :

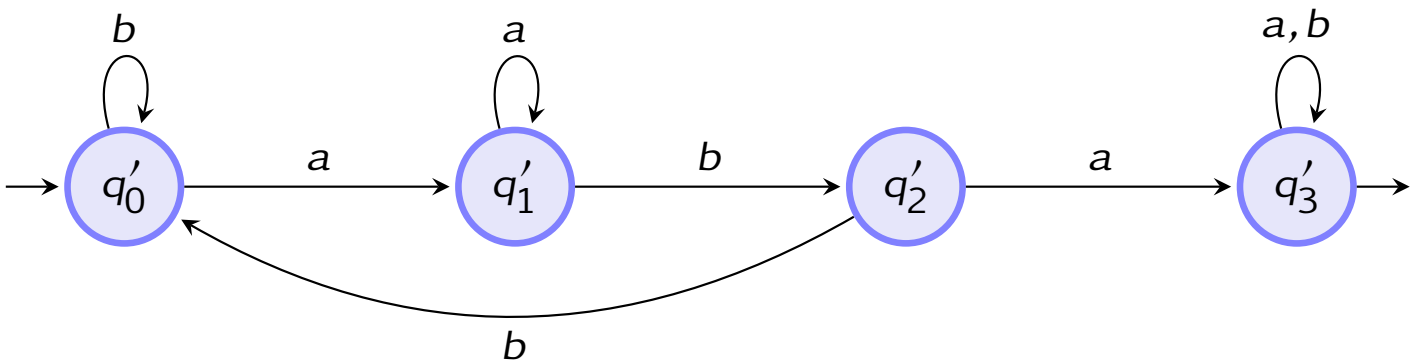


Recherche d'un mot dans un texte

Si $u \in \Sigma^*$ est un mot donné, on souhaite un automate déterministe qui reconnait le langage $\Sigma^* u \Sigma^*$.

Exemple : recherche du mot aba sur l'alphabet $\{a, b\}$.

Les états q'_4 et q'_5 peuvent être supprimés :



Recherche d'un mot dans un texte

Algorithme KMP

On note $P(u)$ l'ensemble des préfixes de u, et on note $s(v)$ le plus long suffixe de v qui soit dans $P(u)$.
On considère l'automate déterministe $A = (\Sigma, P(u), \{\varepsilon\}, \{u\}, \delta)$ où la fonction de transition est définie par :

$$\forall p \in P(u), \quad \forall x \in \Sigma, \qquad \delta(p, x) = s(px)$$

Recherche d'un mot dans un texte

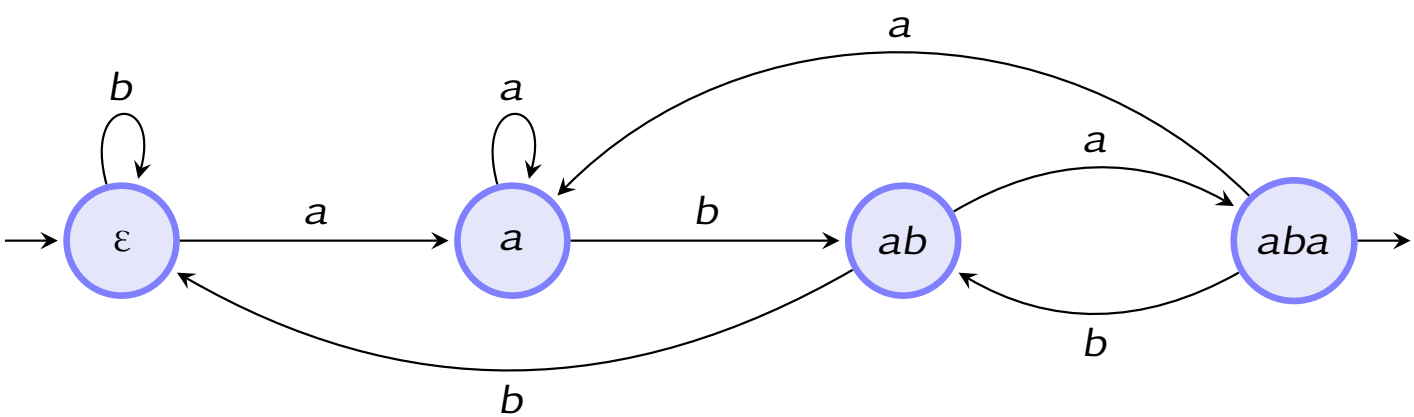
Algorithme KMP

On note $P(u)$ l'ensemble des préfixes de u, et on note $s(v)$ le plus long suffixe de v qui soit dans $P(u)$.
On considère l'automate déterministe $A = (\Sigma, P(u), \{\varepsilon\}, \{u\}, \delta)$ où la fonction de transition est définie par :

$$\forall p \in P(u), \quad \forall x \in \Sigma, \qquad \delta(p,x) = s(px)$$

Exemple. Le cas du mot $u = aba$.

δ	a	b
ε	a	ε
a	a	ab
ab	aba	ε
aba	a	ab



Recherche d'un mot dans un texte

Algorithme KMP

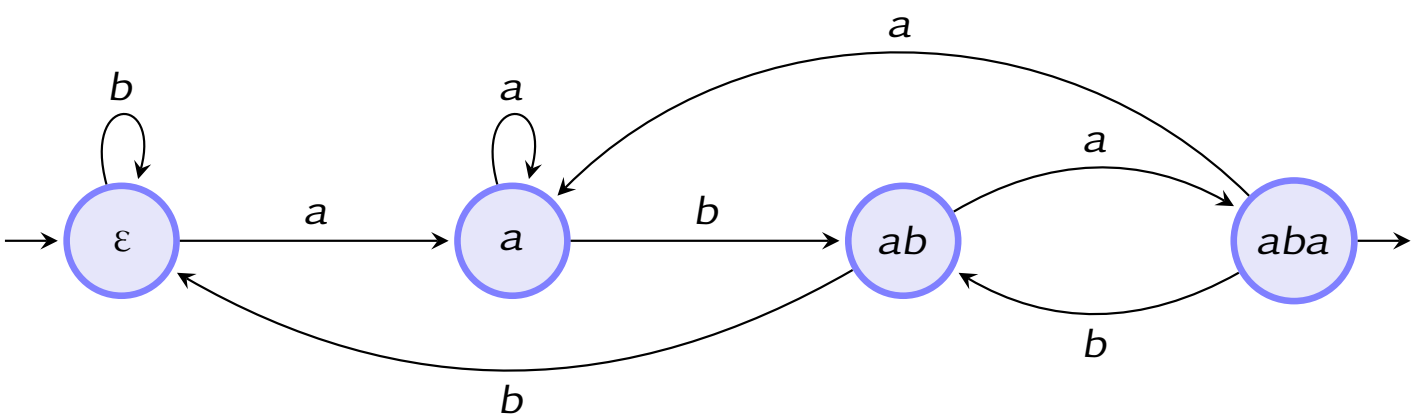
On note $P(u)$ l'ensemble des préfixes du mot u.
On note $s(v)$ le plus long suffixe de v dans $P(u)$.
On pose $A = (\Sigma, P(u), \{\varepsilon\}, \{u\}, \delta)$ avec $\delta(p, x) = s(px)$.

Recherche d'un mot dans un texte

Algorithme KMP

On note $P(u)$ l'ensemble des préfixes du mot u.
On note $s(v)$ le plus long suffixe de v dans $P(u)$.
On pose $A = (\Sigma, P(u), \{\varepsilon\}, \{u\}, \delta)$ avec $\delta(p, x) = s(px)$.
Exemple : $u = aba$.

δ	a	b
ε	a	ε
a	a	ab
ab	aba	ε
aba	a	ab



Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Si $p \in P(u)$ et $v \in \Sigma^*$ on prouve par récurrence sur $|v|$ que le chemin qui part de l'état p et qui est étiqueté par v mène à l'état $s(pv)$.

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Si $p \in P(u)$ et $v \in \Sigma^*$ on prouve par récurrence sur $|v|$ que le chemin qui part de l'état p et qui est étiqueté par v mène à l'état $s(pv)$.

- Si $v = \varepsilon$ on a $s(p) = p$ puisque $p \in P(u)$.

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Si $p \in P(u)$ et $v \in \Sigma^*$ on prouve par récurrence sur $|v|$ que le chemin qui part de l'état p et qui est étiqueté par v mène à l'état $s(pv)$.

- Si $v = \varepsilon$ on a $s(p) = p$ puisque $p \in P(u)$.
- Si $v \neq \varepsilon$, on pose $v = v'a$ et on suppose que le chemin qui part de p étiqueté par v' mène à l'état $s(pv')$. Celui étiqueté par v mène à l'état $s(s(pv')a)$.

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Si $p \in P(u)$ et $v \in \Sigma^*$ on prouve par récurrence sur $|v|$ que le chemin qui part de l'état p et qui est étiqueté par v mène à l'état $s(pv)$.

- Si $v = \varepsilon$ on a $s(p) = p$ puisque $p \in P(u)$.
- Si $v \neq \varepsilon$, on pose $v = v'a$ et on suppose que le chemin qui part de p étiqueté par v' mène à l'état $s(pv')$. Celui étiqueté par v mène à l'état $s(s(pv')a)$.

Soit $w = s(s(pv')a)$ et $w' = s(pv')$. w est suffixe de $w'a$ et w' suffixe de pv' donc w est suffixe de $pv'a = pv$. De plus $w \in P(u)$ donc w est suffixe de $s(pv)$.

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Si $p \in P(u)$ et $v \in \Sigma^*$ on prouve par récurrence sur $|v|$ que le chemin qui part de l'état p et qui est étiqueté par v mène à l'état $s(pv)$.

- Si $v = \varepsilon$ on a $s(p) = p$ puisque $p \in P(u)$.
- Si $v \neq \varepsilon$, on pose $v = v'a$ et on suppose que le chemin qui part de p étiqueté par v' mène à l'état $s(pv')$. Celui étiqueté par v mène à l'état $s(s(pv')a)$.

Soit $w = s(s(pv')a)$ et $w' = s(pv')$. w est suffixe de $w'a$ et w' suffixe de pv' donc w est suffixe de $pv'a = pv$. De plus $w \in P(u)$ donc w est suffixe de $s(pv)$.

Si $s(pv) = \varepsilon$ alors $w = \varepsilon$. Si $s(pv) \neq \varepsilon$ on pose $s(pv) = xa$. Alors $x \in P(u)$ et x est suffixe de pv' donc x est suffixe de $s(pv') = w'$ et xa suffixe de $w'a$. Puisque $xa \in P(u)$ alors xa est suffixe de $s(w'a) = w$.

Dans les deux cas $s(pv)$ est suffixe de w donc $w = s(pv)$.

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Si $p \in P(u)$ et $v \in \Sigma^*$ on prouve par récurrence sur $|v|$ que le chemin qui part de l'état p et qui est étiqueté par v mène à l'état $s(pv)$.

- Si $v = \varepsilon$ on a $s(p) = p$ puisque $p \in P(u)$.
- Si $v \neq \varepsilon$, on pose $v = v'a$ et on suppose que le chemin qui part de p étiqueté par v' mène à l'état $s(pv')$. Celui étiqueté par v mène à l'état $s(s(pv')a)$.

Soit $w = s(s(pv')a)$ et $w' = s(pv')$. w est suffixe de $w'a$ et w' suffixe de pv' donc w est suffixe de $pv'a = pv$. De plus $w \in P(u)$ donc w est suffixe de $s(pv)$.

Si $s(pv) = \varepsilon$ alors $w = \varepsilon$. Si $s(pv) \neq \varepsilon$ on pose $s(pv) = xa$. Alors $x \in P(u)$ et x est suffixe de pv' donc x est suffixe de $s(pv') = w'$ et xa suffixe de $w'a$. Puisque $xa \in P(u)$ alors xa est suffixe de $s(w'a) = w$.

Dans les deux cas $s(pv)$ est suffixe de w donc $w = s(pv)$.

Ainsi le chemin qui part de l'état ε et étiqueté par un mot v mène à l'état $s(v)$, et

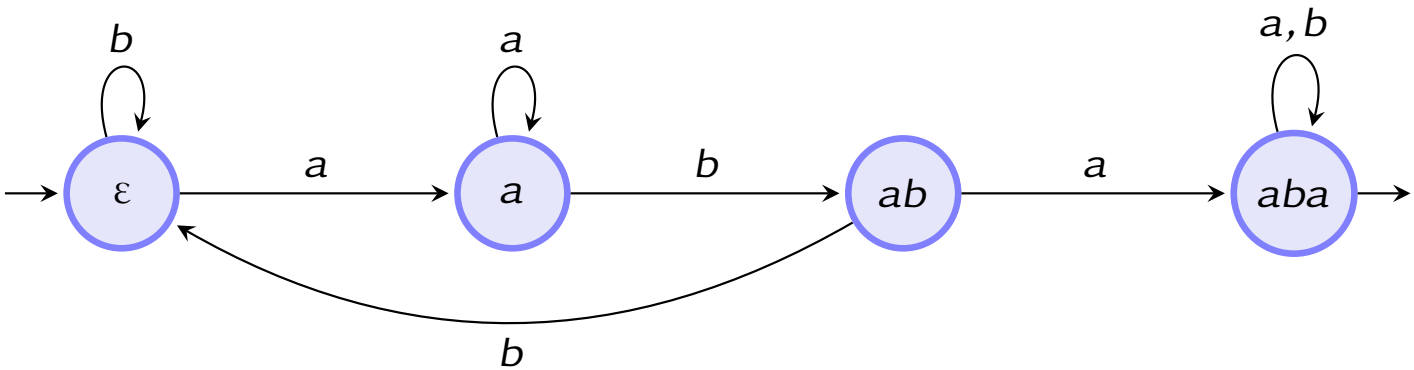
$$s(v) = u \iff v \in \Sigma^* u.$$

Recherche d'un mot dans un texte

Algorithme KMP

L'automate A reconnait le langage $\Sigma^* u$.

Pour obtenir un automate qui reconnait le langage $\Sigma^* u \Sigma^*$ il suffit de considérer l'automate A et de transformer l'état u en puit.



Un automate qui reconnait $\Sigma^* aba \Sigma^*$.

Automates finis et langages rationnels

Théorème de Kleene

Un langage L sur un alphabet Σ est rationnel si et seulement s'il existe un automate fini A tel que $L(A) = L$.

Automates finis et langages rationnels

Théorème de Kleene

Un langage L sur un alphabet Σ est rationnel si et seulement s'il existe un automate fini A tel que $L(A) = L$.

Deux étapes de la preuve :

- l'algorithme de Berry-Sethi construit explicitement l'automate de Glushkov associé à une expression rationnelle ;

Automates finis et langages rationnels

Théorème de Kleene

Un langage L sur un alphabet Σ est rationnel si et seulement s'il existe un automate fini A tel que $L(A) = L$.

Deux étapes de la preuve :

- l'algorithme de Berry-Sethi construit explicitement l'automate de Glushkov associé à une expression rationnelle ;
- l'algorithme de Brzozowski et McCluskey fournit une expression rationnelle qui dénote le langage reconnu par un automate.

(Ce deuxième point n'est pas explicitement au programme.)

Algorithme de Berry-Sethi

Un automate déterministe $A = (\Sigma, Q, q_0, F, \delta)$ est local lorsque pour chaque lettre x toutes les transitions étiquetées par x arrivent dans un même état. Il est standard lorsqu'il n'existe pas de transition aboutissant à l'état initial.

Algorithme de Berry-Sethi

Un automate déterministe $A = (\Sigma, Q, q_0, F, \delta)$ est local lorsque pour chaque lettre x toutes les transitions étiquetées par x arrivent dans un même état. Il est standard lorsqu'il n'existe pas de transition aboutissant à l'état initial.

Si L est un langage local, le langage $L \setminus \{\varepsilon\}$ est reconnaissable par un automate local standard.

Algorithme de Berry-Sethi

Un automate déterministe $A = (\Sigma, Q, q_0, F, \delta)$ est local lorsque pour chaque lettre x toutes les transitions étiquetées par x arrivent dans un même état. Il est standard lorsqu'il n'existe pas de transition aboutissant à l'état initial.

Si L est un langage local, le langage $L \setminus \{\varepsilon\}$ est reconnaissable par un automate local standard.

Considérons l'automate $A = (\Sigma, \Sigma \cup \{\varepsilon\}, \varepsilon, S, \delta)$ avec :

$$\forall x \in P,\ \delta(\varepsilon, x) = x \qquad \text{et} \qquad \forall xy \in F,\ \delta(x, y) = y.$$

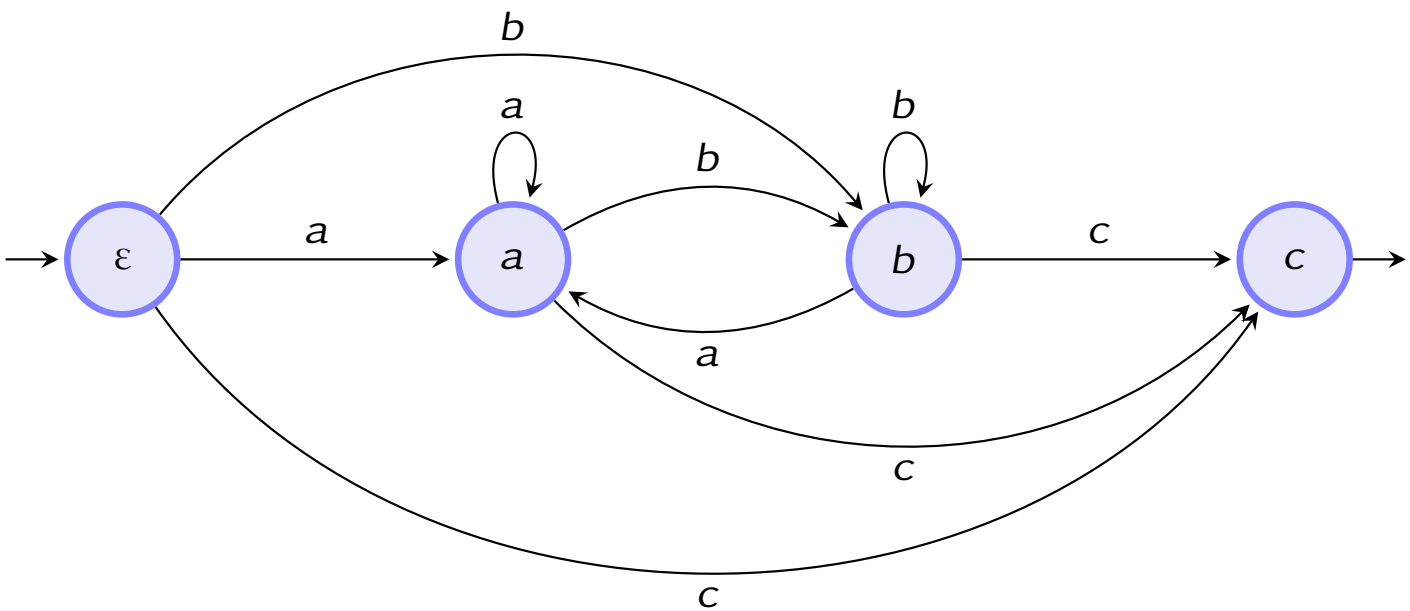
$u = a_1 a_2 \cdots a_n$ est reconnu par A ssi $a_1 \in P$, $a_i a_{i+1} \in F$ et $a_n \in S$.

Algorithme de Berry-Sethi

Un automate déterministe $A = (\Sigma, Q, q_0, F, \delta)$ est local lorsque pour chaque lettre x toutes les transitions étiquetées par x arrivent dans un même état. Il est standard lorsqu'il n'existe pas de transition aboutissant à l'état initial.

Si L est un langage local, le langage $L \setminus \{\varepsilon\}$ est reconnaissable par un automate local standard.

Exemple : $L = (a+b)^* c$, $P = \{a, b, c\}$, $S = \{c\}$ et $F = \{aa, ab, ba, bb, ac, bc\}$.



Algorithme de Berry-Sethi

Construction de l'automate de Glushkov

On considère une expression rationnelle sans $\emptyset$ ni ε : $e = (ab + b)^* ba$.

Algorithme de Berry-Sethi

Construction de l'automate de Glushkov

On considère une expression rationnelle sans $\emptyset$ ni ε : $e = (ab + b)^* ba$.

1. on linéarise e par marquage : $e' = (c_1 c_2 + c_3)^* c_4 c_5$;

Algorithme de Berry-Sethi

Construction de l'automate de Glushkov

On considère une expression rationnelle sans $\emptyset$ ni ε : $e = (ab + b)^* ba$.

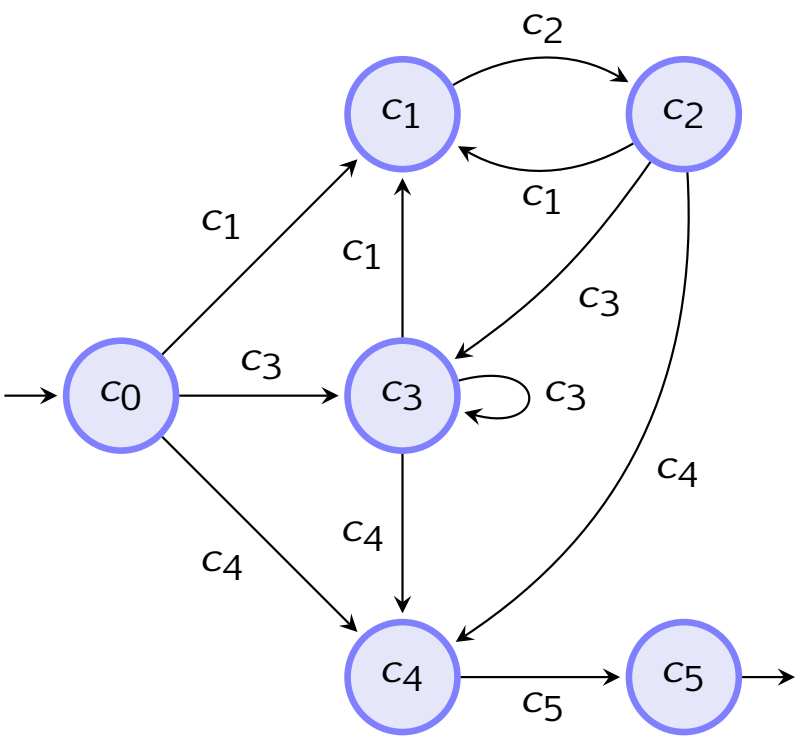
1. on linéarise e par marquage : $e' = (c_1 c_2 + c_3)^* c_4 c_5$;
2. $P = \{c_1, c_3, c_4\}$, $S = \{c_5\}$, $F = \{c_1 c_2, c_2 c_1, c_2 c_3, c_2 c_4, c_3 c_1, c_3 c_3, c_3 c_4, c_4 c_5\}$;

Algorithme de Berry-Sethi

Construction de l'automate de Glushkov

On considère une expression rationnelle sans $\emptyset$ ni ε : $e = (ab + b)^* ba$.

1. on linéarise e par marquage : $e' = (c_1 c_2 + c_3)^* c_4 c_5$;
2. $P = \{c_1, c_3, c_4\}$, $S = \{c_5\}$, $F = \{c_1c_2, c_2c_1, c_2c_3, c_2c_4, c_3c_1, c_3c_3, c_3c_4, c_4c_5\}$;
3. on construit l'automate local associé ;

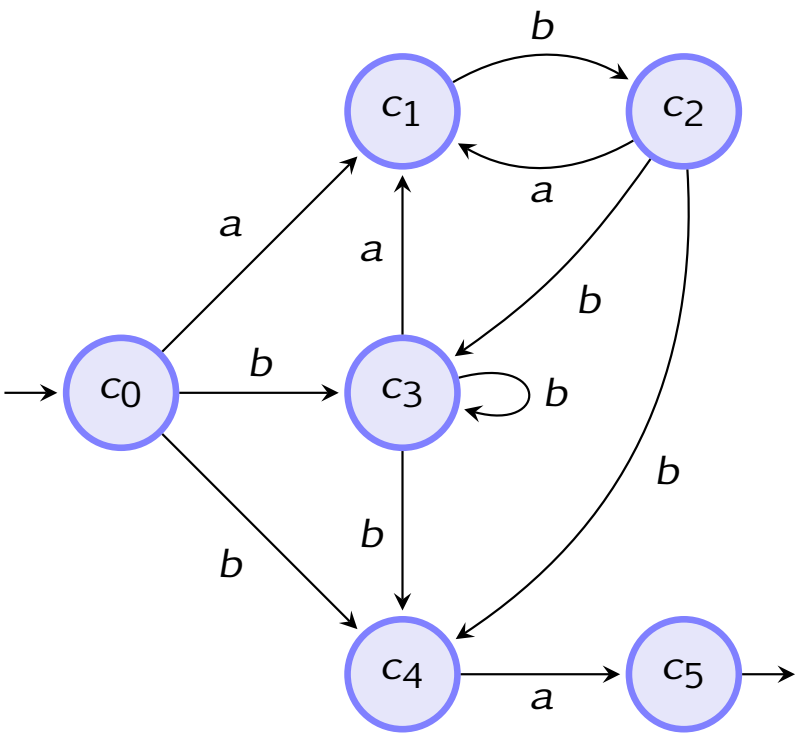


Algorithme de Berry-Sethi

Construction de l'automate de Glushkov

On considère une expression rationnelle sans $\emptyset$ ni ε : $e = (ab + b)^* ba$.

1. on linéarise e par marquage : $e' = (c_1 c_2 + c_3)^* c_4 c_5$;
2. $P = \{c_1, c_3, c_4\}$, $S = \{c_5\}$, $F = \{c_1 c_2, c_2 c_1, c_2 c_3, c_2 c_4, c_3 c_1, c_3 c_3, c_3 c_4, c_4 c_5\}$;
3. on construit l'automate local associé ;
4. on supprime le marquage ;



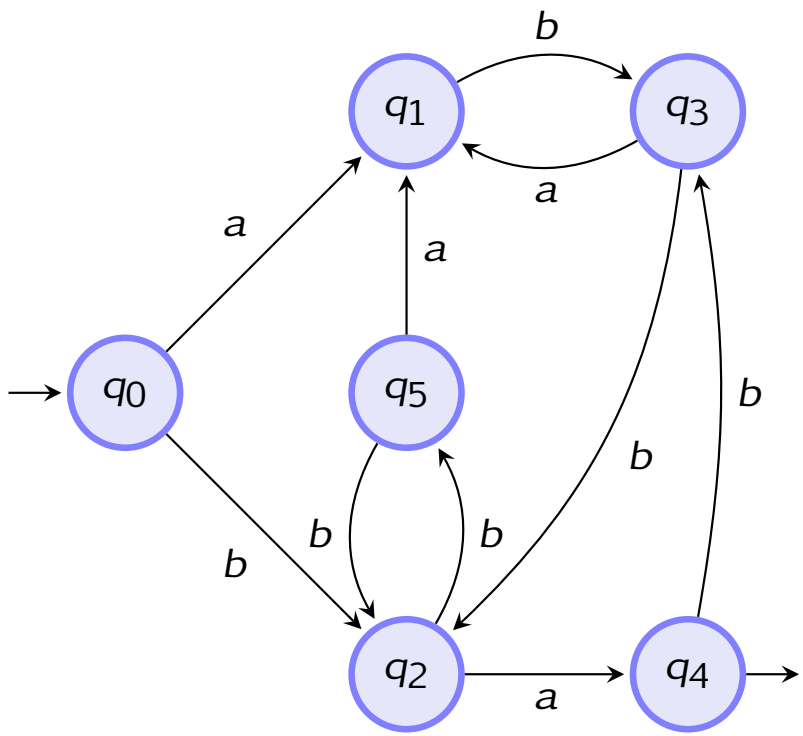
Algorithme de Berry-Sethi

Construction de l'automate de Glushkov

On considère une expression rationnelle sans $\emptyset$ ni ε : $e = (ab + b)^* ba$.

1. on linéarise e par marquage : $e' = (c_1 c_2 + c_3)^* c_4 c_5$;
2. $P = \{c_1, c_3, c_4\}$, $S = \{c_5\}$, $F = \{c_1 c_2, c_2 c_1, c_2 c_3, c_2 c_4, c_3 c_1, c_3 c_3, c_3 c_4, c_4 c_5\}$;
3. on construit l'automate local associé ;
4. on supprime le marquage ;
5. on déterminise l'automate.

	δ'	a	b
*	$\{c_0\}$	$\{c_1\}$	$\{c_3, c_4\}$
	$\{c_1\}$	–	$\{c_2\}$
	$\{c_3, c_4\}$	$\{c_1, c_5\}$	$\{c_3\}$
	$\{c_2\}$	$\{c_1\}$	$\{c_3, c_4\}$
*	$\{c_1, c_5\}$	–	$\{c_2\}$
	$\{c_3\}$	$\{c_1\}$	$\{c_3, c_4\}$



Algorithme de Brzozowski et McCluskey

Un automate généralisé est un automate $A = (\mathrm{Rat}(\Sigma), Q, I, F, \delta)$ dont les états et les transitions sont en nombres finis et dont les transitions sont étiquetées par des expressions rationnelles.

Algorithme de Brzozowski et McCluskey

Un automate généralisé est un automate $A = (\mathrm{Rat}(\Sigma), Q, I, F, \delta)$ dont les états et les transitions sont en nombres finis et dont les transitions sont étiquetées par des expressions rationnelles.

Un mot u est reconnu s'il existe un chemin $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ menant d'un état initial à un état acceptant tel que u appartienne au langage dénoté par $e_1 e_2 \cdots e_n$.

Algorithme de Brzozowski et McCluskey

Un automate généralisé est un automate $A = (\mathrm{Rat}(\Sigma), Q, I, F, \delta)$ dont les états et les transitions sont en nombres finis et dont les transitions sont étiquetées par des expressions rationnelles.

Un mot u est reconnu s'il existe un chemin $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ menant d'un état initial à un état acceptant tel que u appartienne au langage dénoté par $e_1 e_2 \cdots e_n$.

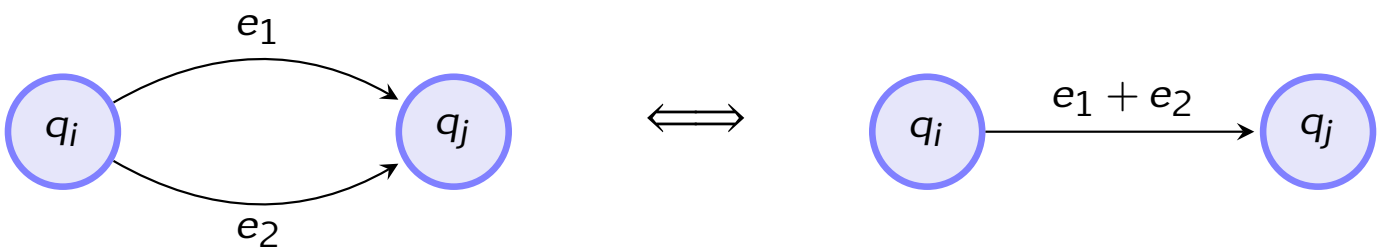
- Quitte à ajouter un état initial i et un état final f on peut supposer qu'un automate généralisé ne possède qu'un seul état initial et un seul état acceptant ;

Algorithme de Brzozowski et McCluskey

Un automate généralisé est un automate $A = (\mathrm{Rat}(\Sigma), Q, I, F, \delta)$ dont les états et les transitions sont en nombres finis et dont les transitions sont étiquetées par des expressions rationnelles.

Un mot u est reconnu s'il existe un chemin $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ menant d'un état initial à un état acceptant tel que u appartienne au langage dénoté par $e_1 e_2 \cdots e_n$.

- Quitte à ajouter un état initial i et un état final f on peut supposer qu'un automate généralisé ne possède qu'un seul état initial et un seul état acceptant ;
- on peut aussi supposer qu'entre deux états il n'existe qu'au plus une transition.



Algorithme de Brzozowski et McCluskey

Tout langage reconnu par un automate généralisé est rationnel.

Algorithme de Brzozowski et McCluskey

Tout langage reconnu par un automate généralisé est rationnel.

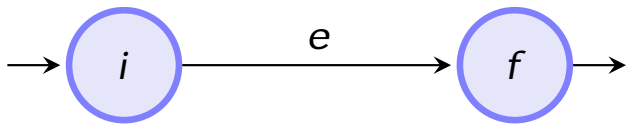
On raisonne par récurrence sur le nombre d'états $|Q|$.

Algorithme de Brzozowski et McCluskey

Tout langage reconnu par un automate généralisé est rationnel.

On raisonne par récurrence sur le nombre d'états $|Q|$.

- Si $|Q| = 2$ l'automate A est de la forme :



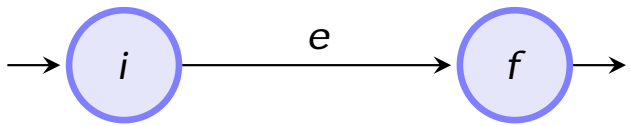
et le langage reconnu par A est dénoté par e.

Algorithme de Brzozowski et McCluskey

Tout langage reconnu par un automate généralisé est rationnel.

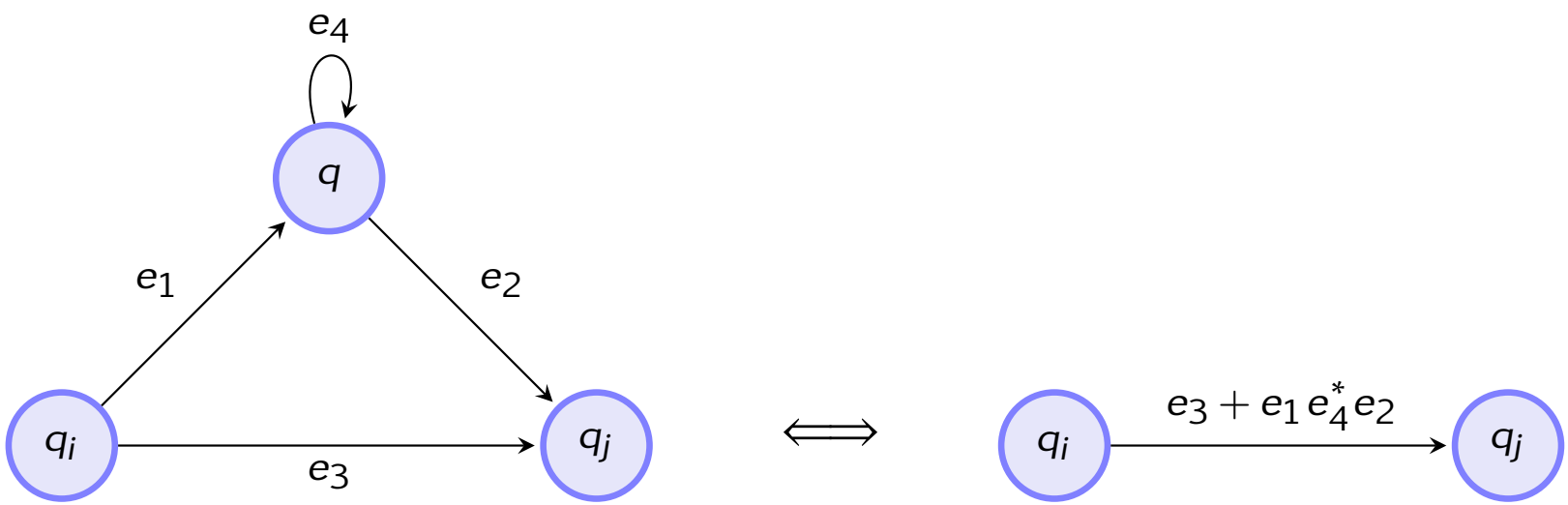
On raisonne par récurrence sur le nombre d'états $|Q|$.

- Si $|Q| = 2$ l'automate A est de la forme :



  et le langage reconnu par A est dénoté par e.

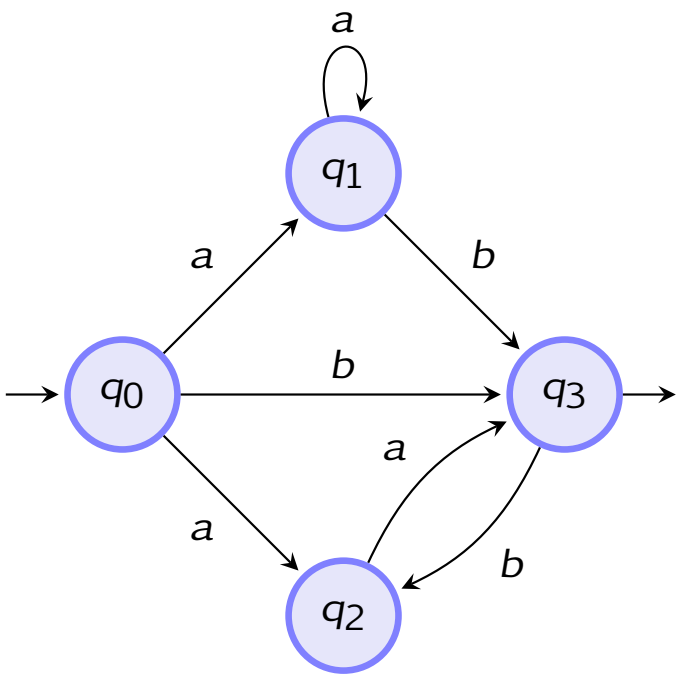
- Si $|Q| > 2$, on élimine un état $q \in Q \setminus \{i, f\}$ en effectuant la transformation suivante pour chaque couple d'états (q_i, q_j) tel que $q_i \neq q$ et $q_j \neq q$:



Algorithme de Brzozowski et McCluskey

Exemple

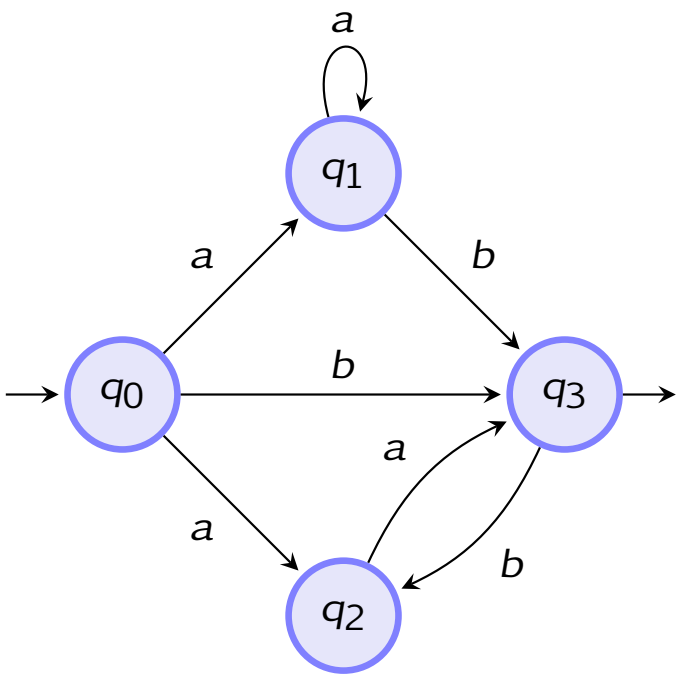
On considère l'automate suivant :



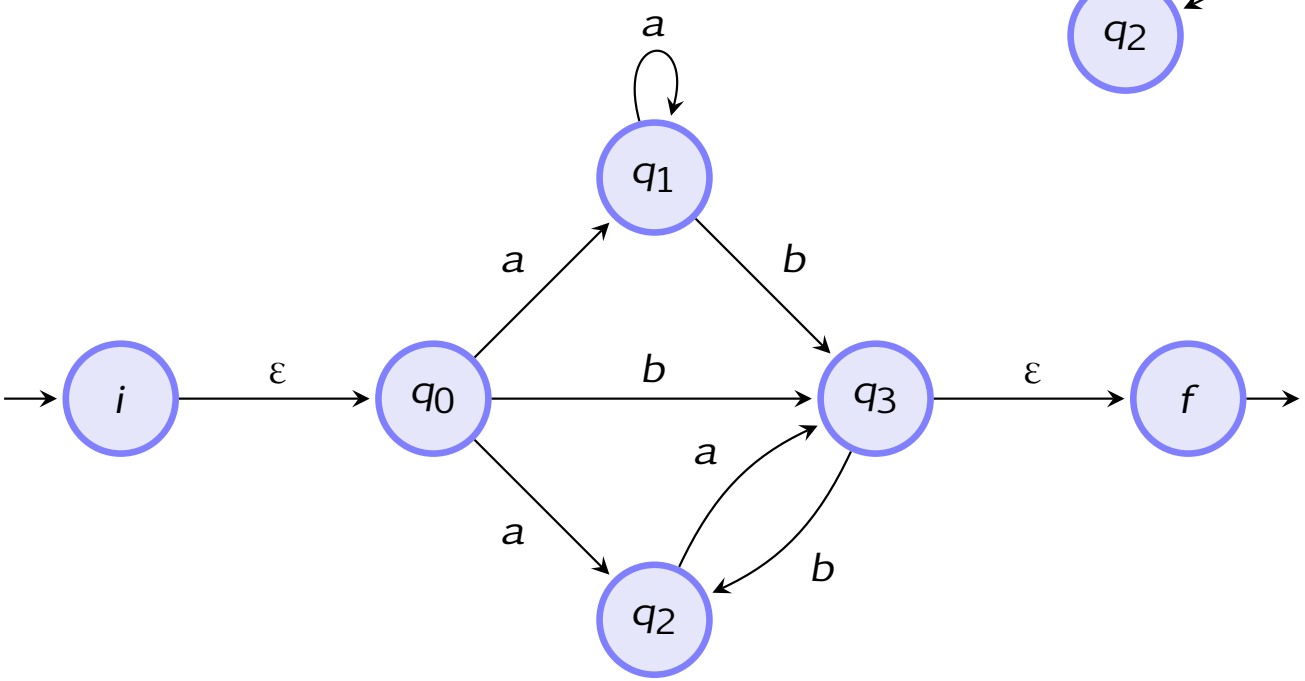
Algorithme de Brzozowski et McCluskey

Exemple

On considère l'automate suivant :



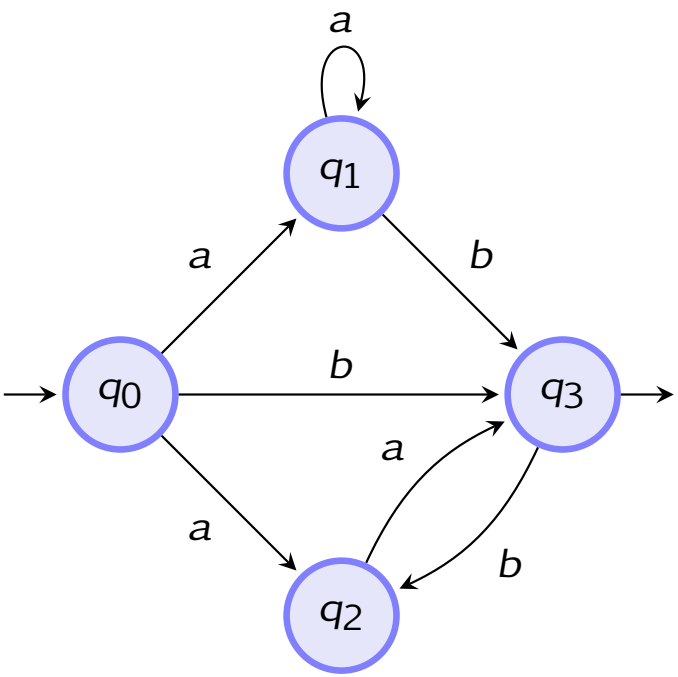
On ajoute un état initial et un état final :



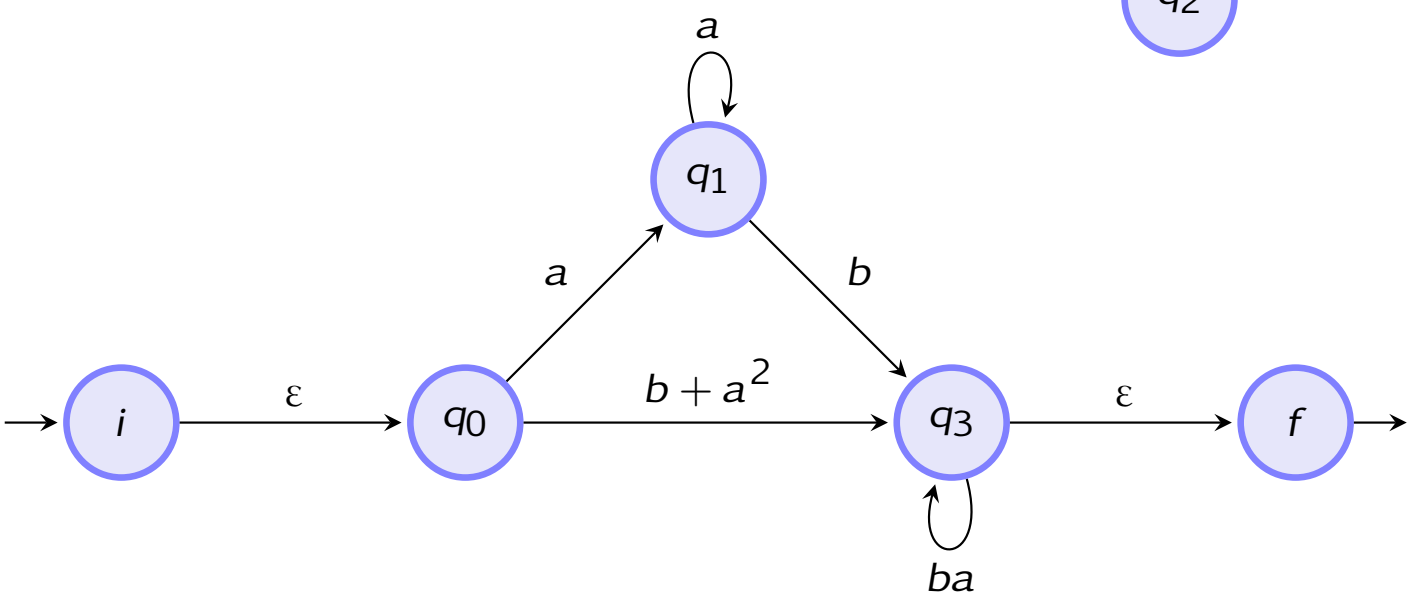
Algorithme de Brzozowski et McCluskey

Exemple

On considère l'automate suivant :



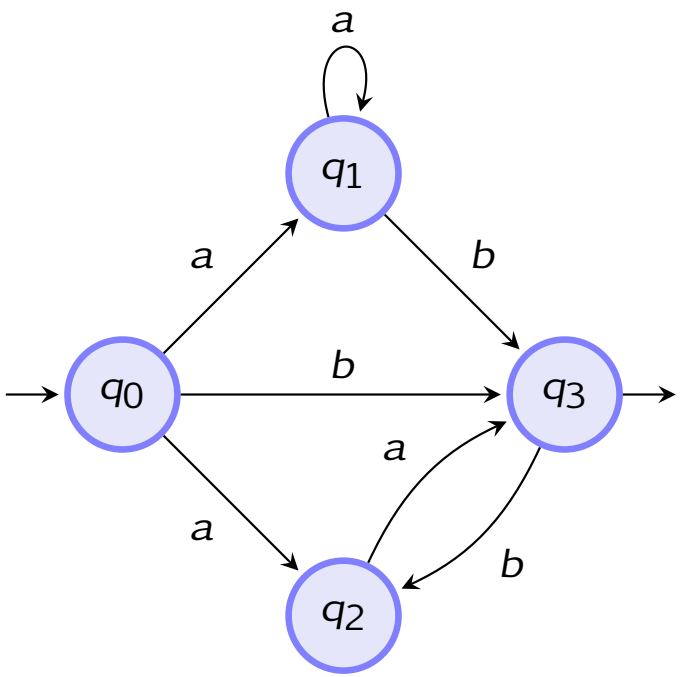
On élimine l'état q_2 :



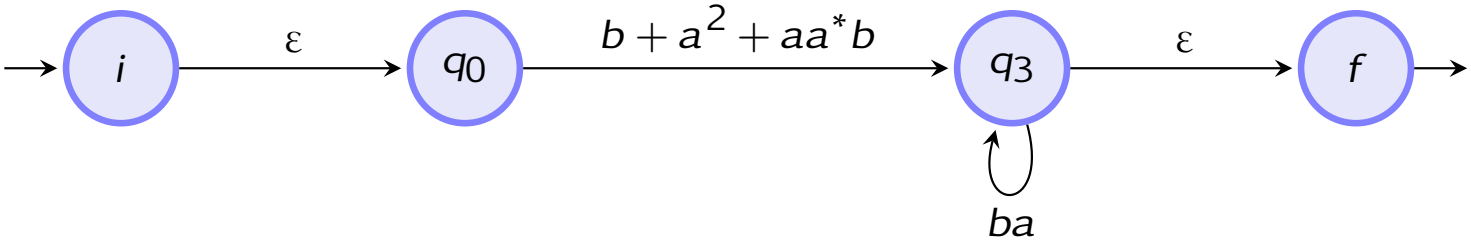
Algorithme de Brzozowski et McCluskey

Exemple

On considère l'automate suivant :



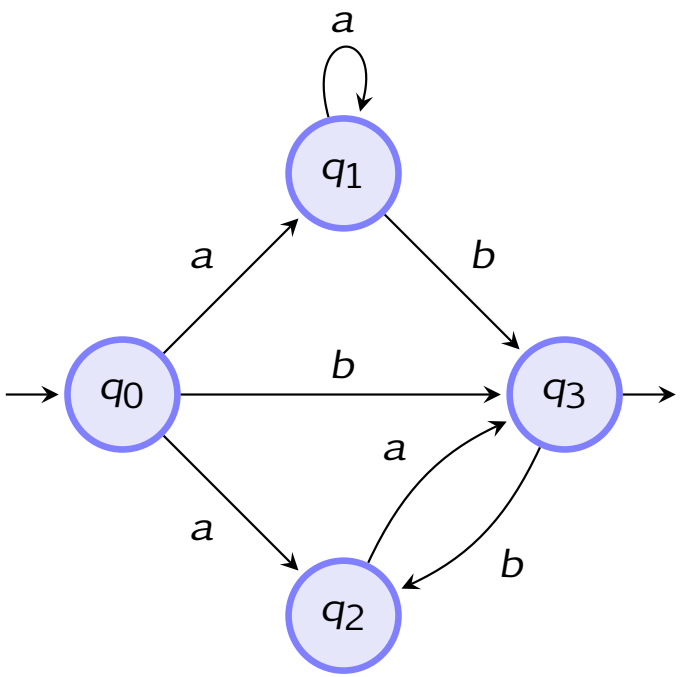
On élimine l'état q_1 :



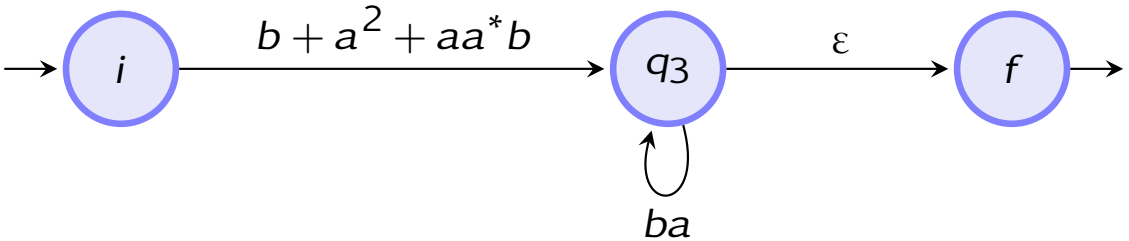
Algorithme de Brzozowski et McCluskey

Exemple

On considère l'automate suivant :



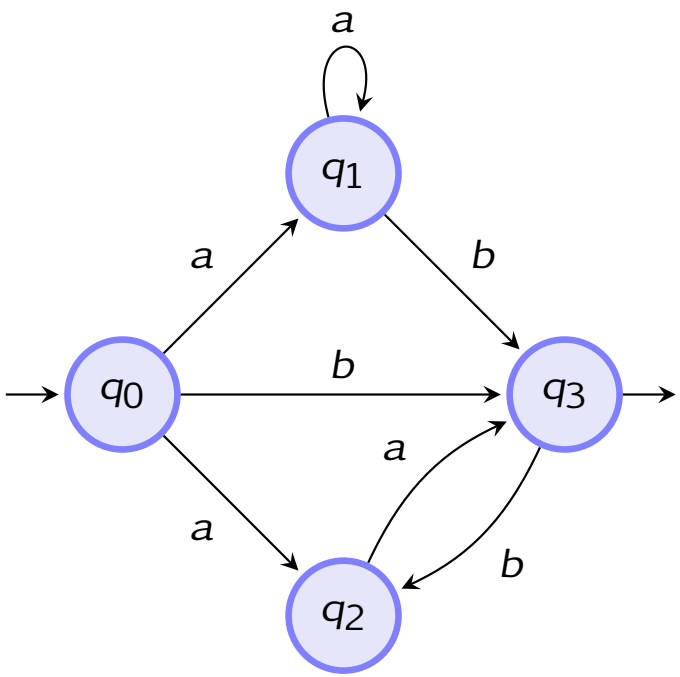
On élimine l'état q_0 :



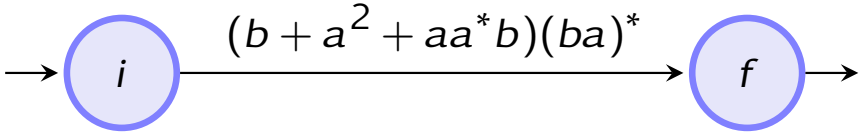
Algorithme de Brzozowski et McCluskey

Exemple

On considère l'automate suivant :



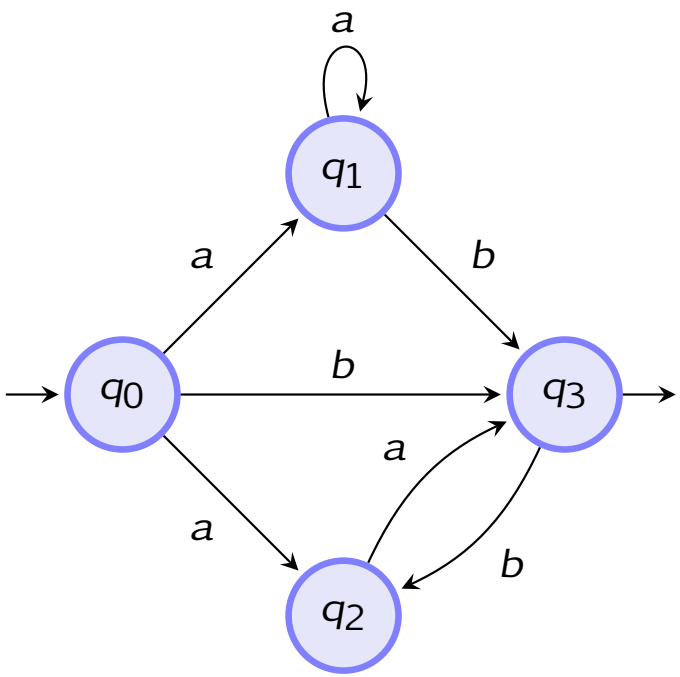
On élimine l'état q_3 :



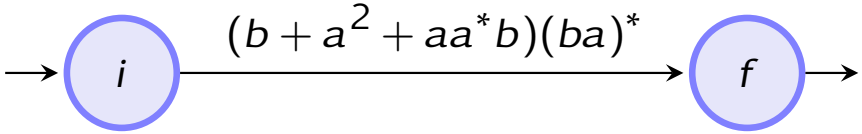
Algorithme de Brzozowski et McCluskey

Exemple

On considère l'automate suivant :



On élimine l'état q_3 :



L'automate reconnait le langage dénoté par $(b + a^2 + aa^*b)(ba)^*$.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Les langages reconnaissables sont clos par intersection.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Les langages reconnaissables sont clos par intersection.

Soient L_1 et L_2 deux langages reconnus par les automates déterministes complets $A_1 = (\Sigma, Q_1, q_{1,0}, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, q_{2,0}, F_2, \delta_2)$. On pose $Q = Q_1 \times Q_2$, $q_0 = (q_{1,0}, q_{2,0})$, $F = F_1 \times F_2$ et $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta(q_2, a))$. Alors $A = (\Sigma, Q, q_0, F, \delta)$ reconnait le langage $L_1 \cap L_2$.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Les langages reconnaissables sont clos par intersection.

Soient L_1 et L_2 deux langages reconnus par les automates déterministes complets $A_1 = (\Sigma, Q_1, q_{1,0}, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, q_{2,0}, F_2, \delta_2)$. On pose $Q = Q_1 \times Q_2$, $q_0 = (q_{1,0}, q_{2,0})$, $F = F_1 \times F_2$ et $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta(q_2, a))$. Alors $A = (\Sigma, Q, q_0, F, \delta)$ reconnait le langage $L_1 \cap L_2$.
En effet, soit $u \in \Sigma^*$ et (q_1, q_2) l'état auquel aboutit le chemin étiqueté par u.

- Si $u \notin L_1$ alors $q_1 \notin F_1$ donc $(q_1, q_2) \notin S$;
- si $u \notin L_2$ alors $q_2 \notin F_2$ donc $(q_1, q_2) \notin S$;
- si $u \in L_1 \cap L_2$ alors $q_1 \in F_1$ et $q_2 \in F_2$ donc $(q_1, q_2) \in F$.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Les langages reconnaissables sont clos par intersection.

Soient L_1 et L_2 deux langages reconnus par les automates déterministes complets $A_1 = (\Sigma, Q_1, q_{1,0}, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, q_{2,0}, F_2, \delta_2)$. On pose $Q = Q_1 \times Q_2$, $q_0 = (q_{1,0}, q_{2,0})$, $F = F_1 \times F_2$ et $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta(q_2, a))$. Alors $A = (\Sigma, Q, q_0, F, \delta)$ reconnait le langage $L_1 \cap L_2$.

Conséquence : *Les langages rationnels sont clos par complémentation et par intersection.*

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Les langages reconnaissables sont clos par intersection.

Soient L_1 et L_2 deux langages reconnus par les automates déterministes complets $A_1 = (\Sigma, Q_1, q_{1,0}, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, q_{2,0}, F_2, \delta_2)$. On pose $Q = Q_1 \times Q_2$, $q_0 = (q_{1,0}, q_{2,0})$, $F = F_1 \times F_2$ et $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta(q_2, a))$. Alors $A = (\Sigma, Q, q_0, F, \delta)$ reconnait le langage $L_1 \cap L_2$.

Les langages reconnaissables sont clos par passage au miroir.

Propriétés des langages reconnaissables

Les langages reconnaissables sont clos par complémentation.

Soit L un langage reconnu par un automate déterministe *complet* $A = (\Sigma, Q, q_0, F, \delta)$ et $\overline{L} = \Sigma^* \setminus L$. Posons $A' = (\Sigma, Q, q_0, Q \setminus F, \delta)$; alors l'automate A' reconnait $\overline{L}$.

Les langages reconnaissables sont clos par intersection.

Soient L_1 et L_2 deux langages reconnus par les automates déterministes complets $A_1 = (\Sigma, Q_1, q_{1,0}, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, q_{2,0}, F_2, \delta_2)$. On pose $Q = Q_1 \times Q_2$, $q_0 = (q_{1,0}, q_{2,0})$, $F = F_1 \times F_2$ et $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta(q_2, a))$. Alors $A = (\Sigma, Q, q_0, F, \delta)$ reconnait le langage $L_1 \cap L_2$.

Les langages reconnaissables sont clos par passage au miroir.

Considérons un automate *non déterministe* $A = (\Sigma, Q, I, F, \delta)$ et définissons l'automate $A' = (\Sigma, Q, F, I, \delta')$ avec :

$$\delta'(q, x) = q' \iff \delta(x, q') = q$$

Alors le langage reconnu par A' est l'image miroir du langage reconnu par A.

Lemme de l'étoile

Si L est un langage rationnel il existe un entier k tel que tout mot $m \in L$ de longueur supérieure ou égale à k se factorise sous la forme $m = uvw$ avec : (i) $|v| \geqslant 1$ (ii) $|uv| \leqslant k$ (iii) $\forall n \in \mathbb{N}$, $uv^n w \in L$.

Lemme de l'étoile

Si L est un langage rationnel il existe un entier k tel que tout mot $m \in L$ de longueur supérieure ou égale à k se factorise sous la forme $m = uvw$ avec : (i) $|v| \geqslant 1$ (ii) $|uv| \leqslant k$ (iii) $\forall n \in \mathbb{N}$, $uv^n w \in L$.

Soit $A = (\Sigma, Q, q_0, F, \delta)$ et $k = |Q|$. Soit m un mot de L tel que $|m| \geqslant k$. Le chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_p} q_p$ reconnaissant m implique $p+1$ états donc passe nécessairement deux fois par le même état $q_i = q_j$ avec $0 \leqslant i < j \leqslant k$:

$$q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_i} q_i \cdots\cdots \xrightarrow{a_j} q_j \longrightarrow \cdots \xrightarrow{a_p} q_p$$

Posons $u = a_1 \cdots a_i$, $v = a_{i+1} \cdots a_j$ et $w = a_{j+1} \cdots a_p$. Alors pour tout $n \in \mathbb{N}$ le chemin étiqueté par $uv^n w$ conduit à l'état acceptant q_p donc $uv^n w \in L$.

Lemme de l'étoile

Si L est un langage rationnel il existe un entier k tel que tout mot $m \in L$ de longueur supérieure ou égale à k se factorise sous la forme $m = uvw$ avec : (i) $|v| \geqslant 1$ (ii) $|uv| \leqslant k$ (iii) $\forall n \in \mathbb{N},\ uv^n w \in L$.

Soit $A = (\Sigma, Q, q_0, F, \delta)$ et $k = |Q|$. Soit m un mot de L tel que $|m| \geqslant k$. Le chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_p} q_p$ reconnaissant m implique $p+1$ états donc passe nécessairement deux fois par le même état $q_i = q_j$ avec $0 \leqslant i < j \leqslant k$:

$$q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_i} q_i \cdots\cdots \xrightarrow{a_j} q_j \longrightarrow \cdots \xrightarrow{a_p} q_p$$

Posons $u = a_1 \cdots a_i$, $v = a_{i+1} \cdots a_j$ et $w = a_{j+1} \cdots a_p$. Alors pour tout $n \in \mathbb{N}$ le chemin étiqueté par $uv^n w$ conduit à l'état acceptant q_p donc $uv^n w \in L$.

Si L est un langage rationnel il existe un entier k tel que tout mot $m = uvw$ tel que $|v| \geqslant k$ se factorise sous la forme $m = u(v_1 v_2 v_3)w$ avec :

$$(i) \quad |v_2| \geqslant 1 \qquad (ii) \quad |v_1 v_2| \leqslant k \qquad (iii) \quad \forall n \in \mathbb{N},\ u(v_1 v_2^n v_3)w \in L.$$

Lemme de l'étoile

Si L est un langage rationnel il existe un entier k tel que tout mot $m \in L$ de longueur supérieure ou égale à k se factorise sous la forme $m = uvw$ avec : (i) $|v| \geqslant 1$ (ii) $|uv| \leqslant k$ (iii) $\forall n \in \mathbb{N},\ uv^n w \in L$.

Soit $A = (\Sigma, Q, q_0, F, \delta)$ et $k = |Q|$. Soit m un mot de L tel que $|m| \geqslant k$. Le chemin $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_p} q_p$ reconnaissant m implique $p+1$ états donc passe nécessairement deux fois par le même état $q_i = q_j$ avec $0 \leqslant i < j \leqslant k$:

$$q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_i} q_i \cdots\cdots \xrightarrow{a_j} q_j \longrightarrow \cdots \xrightarrow{a_p} q_p$$

Posons $u = a_1 \cdots a_i$, $v = a_{i+1} \cdots a_j$ et $w = a_{j+1} \cdots a_p$. Alors pour tout $n \in \mathbb{N}$ le chemin étiqueté par $uv^n w$ conduit à l'état acceptant q_p donc $uv^n w \in L$.

Si L est un langage rationnel il existe un entier k tel que tout mot $m = uvw$ tel que $|v| \geqslant k$ se factorise sous la forme $m = u(v_1 v_2 v_3)w$ avec :

$$(i) \quad |v_2| \geqslant 1 \qquad (ii) \quad |v_1 v_2| \leqslant k \qquad (iii) \quad \forall n \in \mathbb{N},\ u(v_1 v_2^n v_3)w \in L.$$

On applique la méthode décrite dans la preuve précédente uniquement après avoir parcouru le chemin étiqueté par u.

Lemme de l'étoile

Exemple

- Le langage $L = \left\{ a^n b^n \;\middle|\; n \in \mathbb{N} \right\}$ n'est pas rationnel.

Lemme de l'étoile

Exemple

- Le langage $L = \left\{ a^n b^n \;\middle|\; n \in \mathbb{N} \right\}$ n'est pas rationnel.

Supposons les conclusions du lemme de l'étoile vérifiées et posons

$$u = a^k, \qquad v = b^k, \qquad w = \varepsilon.$$

D'après le lemme de l'étoile v se factorise en $v = b^{k_1} b^{k_2} b^{k_3}$ avec $k_2 \geqslant 1$ et on doit avoir : $\forall n \in \mathbb{N}$, $a^k b^{k+nk_2} \in L$, ce qui est absurde.

Lemme de l'étoile

Exemple

- Le langage $L = \left\{ a^n b^n \;\middle|\; n \in \mathbb{N} \right\}$ n'est pas rationnel.

Supposons les conclusions du lemme de l'étoile vérifiées et posons

$$u = a^k, \qquad v = b^k, \qquad w = \varepsilon.$$

D'après le lemme de l'étoile v se factorise en $v = b^{k_1} b^{k_2} b^{k_3}$ avec $k_2 \geqslant 1$ et on doit avoir : $\forall n \in \mathbb{N}, a^k b^{k+nk_2} \in L$, ce qui est absurde.

- Si Σ possède au moins deux lettres le langage $L = \left\{ m^2 \;\middle|\; m \in \Sigma^* \right\}$ des carrés parfaits n'est pas rationnel.

Lemme de l'étoile

Exemple

- Le langage $L = \left\{ a^n b^n \;\middle|\; n \in \mathbb{N} \right\}$ n'est pas rationnel.

Supposons les conclusions du lemme de l'étoile vérifiées et posons

$$u = a^k, \qquad v = b^k, \qquad w = \varepsilon.$$

D'après le lemme de l'étoile v se factorise en $v = b^{k_1} b^{k_2} b^{k_3}$ avec $k_2 \geqslant 1$ et on doit avoir : $\forall n \in \mathbb{N}$, $a^k b^{k+nk_2} \in L$, ce qui est absurde.

- Si Σ possède au moins deux lettres le langage $L = \left\{ m^2 \;\middle|\; m \in \Sigma^* \right\}$ des carrés parfaits n'est pas rationnel.

Supposons les conclusions du lemme de l'étoile vérifiées et posons $m = ab^k$. Le mot m^2 se factorise en $m^2 = uvw$ avec

$$u = a, \qquad v = b^k, \qquad w = ab^k.$$

Alors il existe $j \geqslant 1$ tel que pour tout $n \in \mathbb{N}$, $ab^{k+nj}ab^k \in L$, ce qui est absurde.