Théorie des Langages

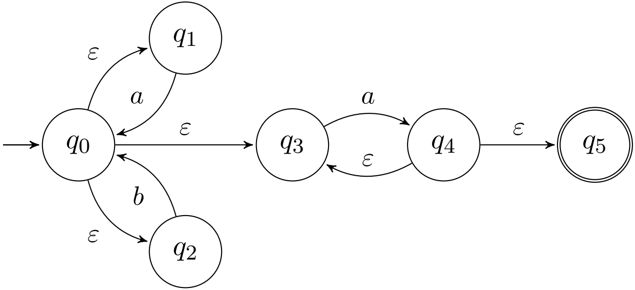
Cours 4 : Transformation d'automates

Lionel Rieg

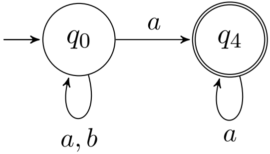
Grenoble INP - Ensimag, 1re année

Plusieurs automates pour le même langage

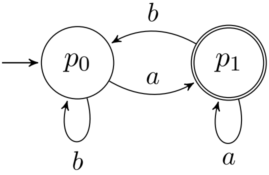
- AFND :



- AFND$-\varepsilon$:



- AFD :



Transformations d'automates

Question

Peut-on toujours effectuer les transformations

$$AF(ND) \Longleftrightarrow \quad AF(ND) - \varepsilon \Longleftrightarrow \quad \text{AFD} \quad ?$$

Sens $\Leftarrow$ trivial : ce sont des cas particuliers

Dans la suite : techniques de transformation, preuves plus tard

Algorithme de suppression des ε-transitions

Deux étapes :

1. Déterminer les états atteignables en n'utilisant que des ε-transitions
2. S'en servir pour construire un automate équivalent sans ε-transition

Étape 1

Définition (États accessibles)

Etant donné un automate $A = \langle Q, V, \delta, I, F \rangle$, on définit par induction pour tout $p \in Q$ l'ensemble $\mathrm{Acc}_\varepsilon(p)$ des états accessibles depuis p par ε-transitions :

- $p \in \mathrm{Acc}_\varepsilon(p)$
- si $q \in \mathrm{Acc}_\varepsilon(p)$ et $(q, \varepsilon, r) \in \delta$ alors $r \in \mathrm{Acc}_\varepsilon(p)$

Calcul de $\mathrm{Acc}_\varepsilon(p)$ par itération (cf cours 2)

Lien avec le théorème du point fixe : $\mathrm{Acc}_\varepsilon(p) = \mu(f)$

avec $f(X) = \{p\} \cup \{q' \mid \exists q,\, q \in X \wedge (q, \varepsilon, q') \in \delta\}$

Rappel sur le calcul par itération pour $\mathrm{Acc}_\varepsilon(p)$

Idée : Calculer les états de $\mathrm{Acc}_\varepsilon(p)$ accessibles en au plus n pas et faire croître n.

$\mathrm{Acc}_\varepsilon(p) = \bigcup_{n \geqslant 0} A_n$, où la suite (A_n) est définie par :

$$\begin{aligned} A_0 &\stackrel{\text{def}}{=} \{p\} \\ A_{n+1} &\stackrel{\text{def}}{=} A_n \cup \{r \in Q \mid \exists q \in A_n \text{ et } (q, \varepsilon, r) \in \delta\} \end{aligned}$$

(comme pour $\mu(f)$ avec $f(X) = \{p\} \cup \{r \in Q \mid \exists q \in A_n. (q, \varepsilon, r) \in \delta\}$)

algorithme $\mathrm{Acc}_\varepsilon(p) =$
  $n \leftarrow 0$, $A_0 \leftarrow \{p\}$
  répéter
    $A_{n+1} \leftarrow A_n \cup \{\kappa_i(e_1, \ldots, e_{k_i}) \mid \kappa_i \in K, e_1, \ldots, e_{k_i} \in A_n\}$
    $n \leftarrow n + 1$
  jusqu'à $A_{n+1} = A_n$
  renvoyer A_n

Question : Est-ce que ça termine toujours ?

Au besoin, on fait un tableau des A_n jusqu'à stabiliser.

Algorithme de suppression des ε-transitions (suite)

Étape 2

Définition

Etant donné un automate $A = \langle Q, V, \delta, I, F \rangle$, on définit l'automate $B = \langle Q, V, \delta', I, F' \rangle$ de la façon suivante :

- $(p, a, q) \in \delta'$ ssi $a \neq \varepsilon$ et $\exists r \in \mathrm{Acc}_\varepsilon(p)$ tel que $(r, a, q) \in \delta$

$$p \underset{\boxed{B}}{\xrightarrow{\quad a \quad}} q \iff \exists\, r,\ p \underset{\boxed{A}}{\overset{\varepsilon}{\rightsquigarrow}} r \underset{\boxed{A}}{\xrightarrow{\quad a \quad}} q$$

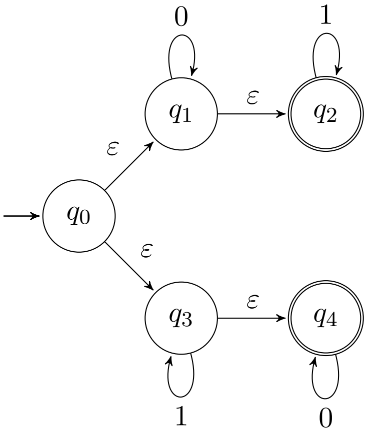
- $F' = \{p \in Q \mid \mathrm{Acc}_\varepsilon(p) \cap F \neq \emptyset\}$

Propositions

- B est un automate sans ε-transition.
- B est équivalent à A. preuve plus tard

Exercice

Construire B pour l'automate A suivant :

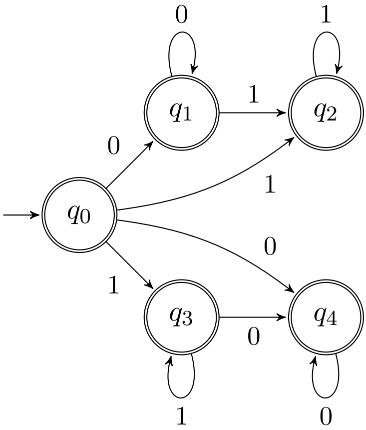


p	$\mathrm{Acc}_\varepsilon(p)$
q_0	q_0, q_1, q_2, q_3, q_4
q_1	q_1, q_2
q_2	q_2
q_3	q_3, q_4
q_4	q_4

δ'	0	1
q_0	q_1, q_4	q_2, q_3
q_1	q_1	q_2
q_2	–	q_2
q_3	q_4	q_3
q_4	q_4	–

$F' = \{q_0, q_1, q_2, q_3, q_4\}$

Exercice (solution)



Transformations d'automates

Question

Peut-on toujours effectuer les transformations

$$AF(ND) \overset{\text{OK}}{\Longleftrightarrow} AF(ND)-\varepsilon \Longleftrightarrow \text{AFD} \quad ?$$

Rappels sur les AFD complets

Définition (Automate déterministe complet)

Un AF $\langle Q, V, \delta, I, F \rangle$ est dit déterministe complet si

1. $\text{Card}(I) = 1$ (exactement un état intial)
2. $\not\exists\, (q, \varepsilon, p) \in \delta$
3. $\forall (q, a) \in Q \times V, \exists! p \in Q, (q, a, p) \in \delta$.

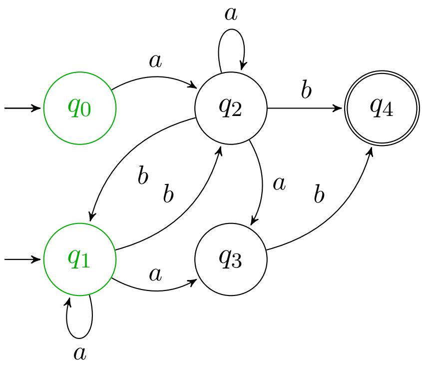
Conséquences

- L'automate a un seul état initial
- δ est une fonction totale : $Q \times V \to Q$
- δ s'étend aux mots : $\delta^* : Q \times V^* \to Q$
- Donne directement un « programme » reconnaisseur

Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

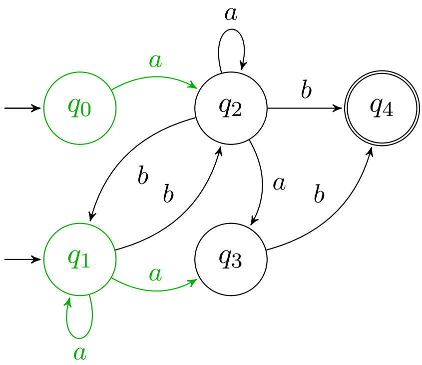
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

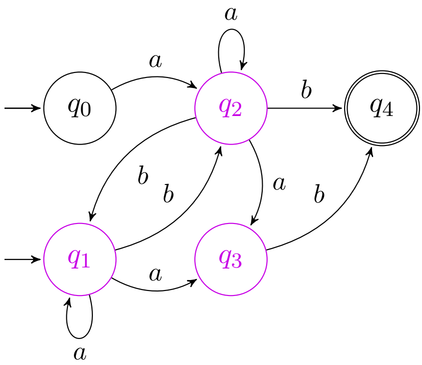
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

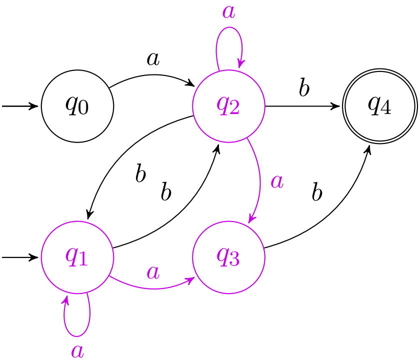
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

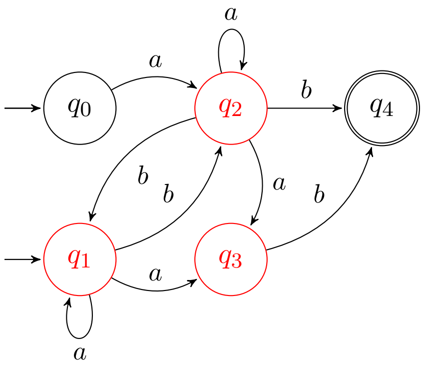
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

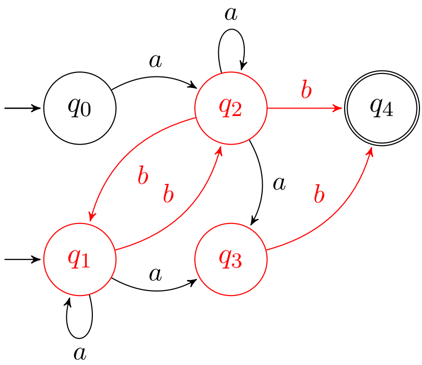
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

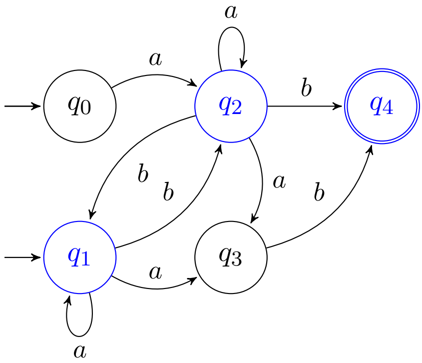
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

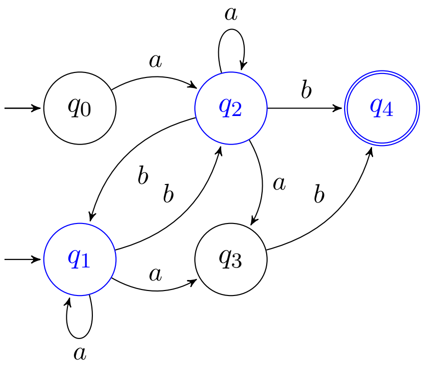


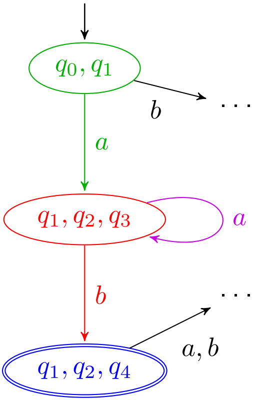
OK : $aab \in \mathcal{L}(A)$

Principe de la déterminisation

Idée : suivre tous les chemins possibles en parallèle

Exemple : L'automate suivant reconnaît-il aab ?





Définition

Définition (Automate des parties)

Etant donné un automate $A = \langle Q, V, \delta_A, I, F_A \rangle$ sans ε-transition, on construit l'automate $B = \langle \mathcal{P}(Q), V, \delta_B, \{I\}, F_B \rangle$, où :

- δ_B est défini par

$$\forall P \subseteq Q, \forall a \in V,\ \delta_B(P, a) = \{q \in Q \mid \exists p \in P : (p, a, q) \in \delta_A\}$$

- $F_B = \{P \subseteq Q \mid P \cap F_A \neq \emptyset\}$

Remarques

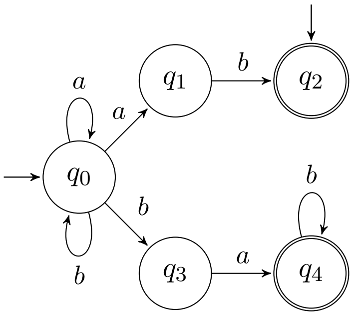
- $P \subseteq Q \iff P \in \mathcal{P}(Q)$ *et* $\emptyset \subseteq Q$ *: un état puits de* B
- *Certains* $P \subseteq Q$ *peuvent ne pas être accessibles depuis* I *donc on construit* B *de proche en proche à partir de* I.

Proposition

L'automate B *est un automate fini déterministe complet équivalent à* A.

Exemple

Construire un AFD (complet) équivalent à :

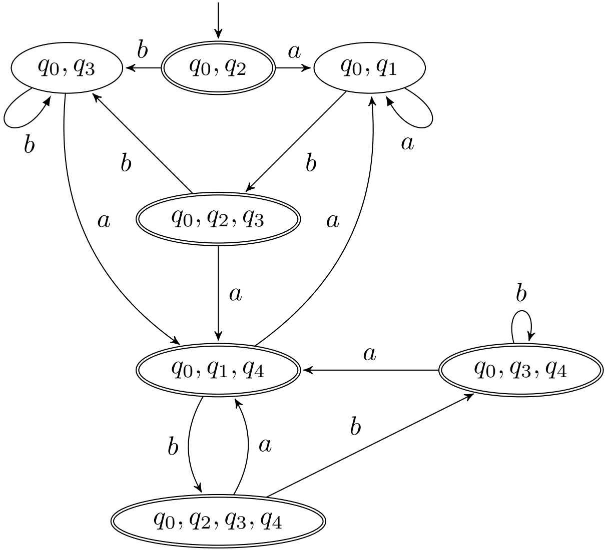


	$\mathcal{P}(Q)$	a	b
$\mathcal{I}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3\}$
	$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3, q_4\}$
	$\{q_0, q_2, q_3, q_4\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3, q_4\}$
	$\{q_0, q_3, q_4\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3, q_4\}$

Pas d'autre état accessible

7 états accessibles (sur 32 potentiels)
5 états acceptants accessibles (sur 24 potentiels)

Solution



Transformations d'automates

Question

Peut-on toujours effectuer les transformations

$$AF(ND) \overset{\text{OK}}{\Longleftrightarrow} AF(ND) - \varepsilon \overset{\text{OK}}{\Longleftrightarrow} \text{AFD} \quad ?$$

En résumé, on peut toujours se ramener à un automate déterministe (complet). On peut donc choisir le type d'automates qui nous arrange.

Question

Est-ce que cet AFD complet caratérise le langage ? (est-il unique ?)

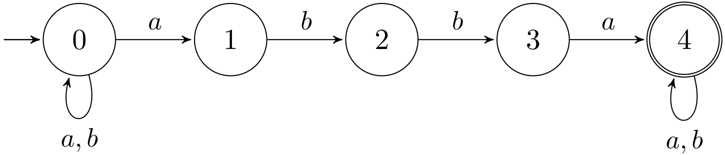
Si oui, on peut tester l'égalité des langages reconnus !

De la déterminisation à la minimisation

On considère le langage

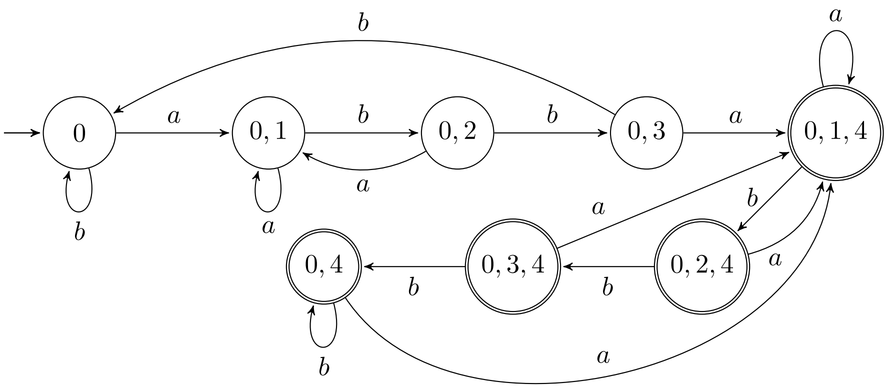
$$L = \{w \in V^* \mid w \text{ contient le motif } abba\}$$

Voici un automate non-déterministe qui le reconnaît :



Regardons l'AFD (complet) équivalent construit par la déterminisation.

AFD complet équivalent



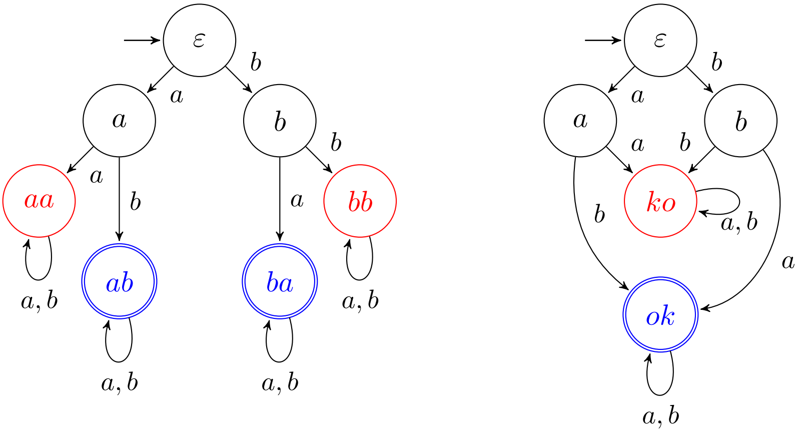
On voit qu'une fois dans un état acceptant, on ne reste que dans des états acceptants. Du coup, on pourrait fusionner tous les états acceptants.

C'est l'objectif de la minimisation d'automate.

Principe de construction

On peut facilement « fusionner » certains états.

Exemple : $L = \{ab, ba\}\{a, b\}^*$



Généralisation

Définition (Minimalité)

Un AFD complet A est minimal si tout AFD complet équivalent à A a au moins autant d'états que A.
Cet automate minimal est unique au renommage des états près.

Définition (Équivalence de Nerode)

Soit $A = \langle Q, V, \delta, \{q_0\}, F \rangle$ un AFD complet.
Deux états $p, q \in Q$ sont équivalents dans A si et seulement si

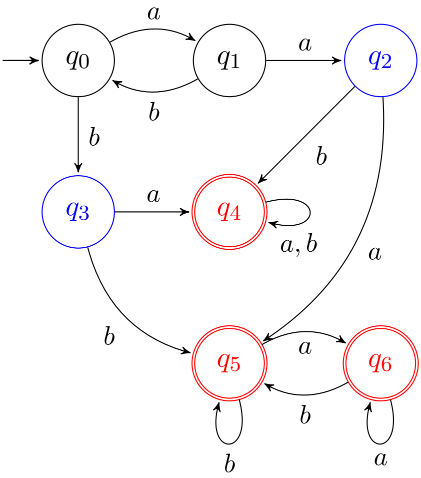
$$\forall w \in V^*,\ (\delta^*(p, w) \in F \iff \delta^*(q, w) \in F)$$

On note alors $p \equiv_A q$, ou simplement $p \equiv q$.

Proposition

Posons $A_p \stackrel{def}{=} \langle Q, V, \delta, \{p\}, F \rangle$ *et* $A_q \stackrel{def}{=} \langle Q, V, \delta, \{q\}, F \rangle$.
On a alors : $p \equiv_A q$ *si et seulement si* A_p *et* A_q *sont équivalents.*

Exemple



$$q_4 \equiv_A q_5 \equiv_A q_6$$

$$q_2 \equiv_A q_3$$

Définition de l'automate minimal

Proposition

1. *$\equiv_A$ est une relation d'équivalence.*
   On note $[p]$ (ou $[p]_A$) la classe d'équivalence de p.
2. *Si $p \equiv_A q$, alors $\forall w \in V^*, \delta^*(p, w) \equiv_A \delta^*(q, w)$.*
   Si $[p]_A = [q]_A$ alors $\forall w \in V^*, [\delta^*(p, w)]_A = [\delta^*(q, w)]_A$.

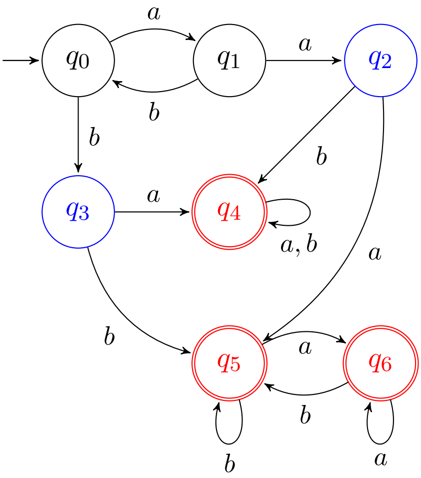
Preuve : exercice.

Définition

Soit $A = \langle Q, V, \delta, \{q_0\}, F\rangle$ un AFD complet et initialement connecté.
On définit $\mu(A) = \langle Q_\mu, V, \delta_\mu, \{[q_0]\}, F_\mu\rangle$, où :

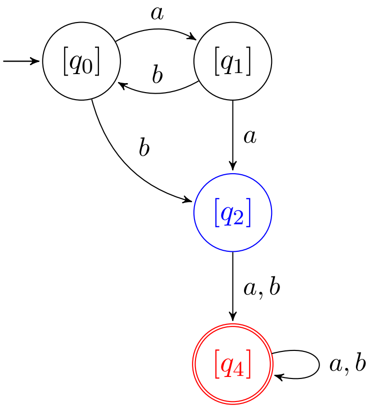
- Q_μ est l'ensemble des classes d'équivalence des états de Q ;
- F_μ est l'ensemble des classes d'équivalence des états de F ;
- $\forall [p] \in Q_\mu, \forall a \in V, \delta_\mu([p], a) = [\delta(p, a)]$.

Exemple



$q_4 \equiv_A q_5 \equiv_A q_6$

$q_2 \equiv_A q_3$



Construction d'un automate minimal

Soit un AFD complet $A = \langle Q, V, \delta, \{q_0\}, F\rangle$ qu'on souhaite minimiser.

1. Supprimer les états inaccessibles de A
2. Déterminer efficacement la relation $\equiv$
   Par approximations successives (cf. $\mathrm{Acc}_\varepsilon(p)$)
3. Construire l'automate minimal : $\langle Q_\mu, V, \delta_\mu, \{[q_0]\}, F_\mu\rangle$

Définition

Pour $k \geqslant 0$, on définit la relation $\equiv_k$ sur Q par : $p \equiv_k q$ si et seulement si

p et q sont équivalents pour tous les mots de longueur au plus k.

Formellement :

$$\forall w \in V^*,\ \text{si } |w| \leqslant k,\ \text{alors } (\delta^*(p, w) \in F \iff \delta^*(q, w) \in F)$$

Si $p \equiv_k q$, alors les automates A_p et A_q reconnaissent exactement les mêmes mots de longueur au plus k.

Calcul de $\equiv$

Proposition

On a les propriétés suivantes :

$$\equiv_{k+1} \quad \subseteq \quad \equiv_k \qquad\qquad \equiv \; = \; \bigcap_{k \geqslant 0} \equiv_k$$

Proposition (Stabilisation de la suite $\equiv_k$)

Si A est un AFD *à n états, alors*
il existe $k \leqslant n$ tel que les relations $\equiv_k$, $\equiv_{k+1}$ et $\equiv$ sont identiques.

Donc, si on sait calculer les relations $\equiv_k$ efficacement,
on saura en déduire la relation $\equiv$.

Calcul de $\equiv$ (suite)

Proposition

On a les propriétés suivantes :

1. $p \equiv_0 q$ *si et seulement si* $p \in F \iff q \in F$
2. $\forall k \geqslant 0$, $p \equiv_{k+1} q$ *si et seulement si*

$$p \equiv_k q \text{ et } \forall a \in V,\ \delta(p,a) \equiv_k \delta(q,a)$$

Preuve : exercice

Conséquences

- $\equiv_0$ contient (en général) deux classes : F et $Q \setminus F$
- Si $p \equiv_k q$ et $\exists a \in V$ tel que $\delta(p,a) \not\equiv_k \delta(q,a)$, alors $p \not\equiv_{k+1} q$
- Le calcul des $\equiv_k$ s'apparente au calcul des A_n pour $\mathrm{Acc}_\varepsilon(p)$ mais à l'envers ! (on fait réduire des classes et non grossir des ensembles)

Version par le théorème du point fixe

Cela ressemble beaucoup au calcul pour un point fixe :

- Calcul approché d'une limite monotone
- Stabilisation au bout d'un certain nombre d'itérations
- Mais on part « à l'envers »

Version originale	Version renversée
inclusion $\subseteq$	$\supseteq$
minimum $\emptyset$	maximum E
$(A_i)_{i\in\mathbb{N}}$ croissante	$(A_i)_{i\in\mathbb{N}}$ décroissante
$\mu(f) = \bigcup_{i\in\mathbb{N}} f^i(\emptyset)$	plus grand point fixe $\bigcap_{i\in\mathbb{N}} f^i(E)$

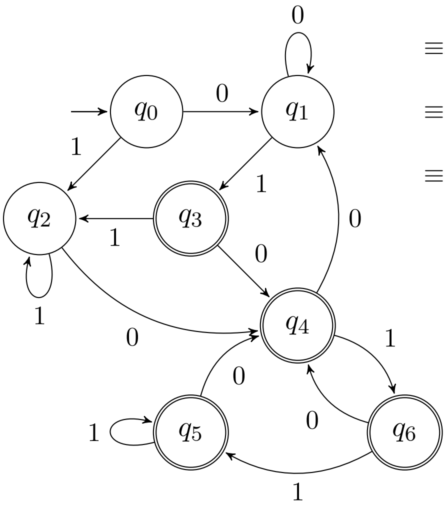
$$f(X) \stackrel{\text{def}}{=} (F \times F \cup (Q \setminus F) \times (Q \setminus F)) \\ \cap \{(q,q') \in X \mid \forall a \in V.\, (\delta(q,a), \delta(q',a)) \in X\}$$

$$f^0(Q \times Q) = Q \times Q \qquad f^1(Q \times Q) = F \times F \cup (Q \setminus F) \times (Q \setminus F)$$

$$f^2(Q \times Q) = \left\{(q,q') \in f^1(Q \times Q) \;\middle|\; \forall a \in V.\, (\delta(q,a), \delta(q',a)) \in f^1(Q \times Q)\right\}$$

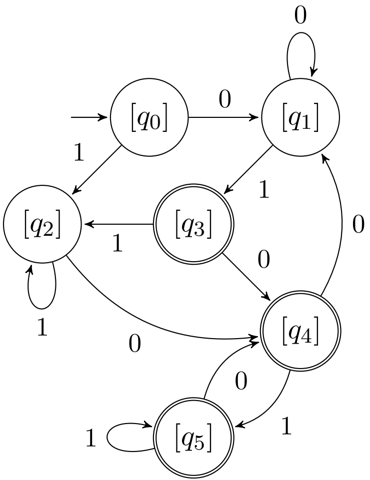
$$f^3(Q \times Q) = \ldots \qquad f^{k+1}(Q \times Q) \quad = \quad \equiv_k$$

Exemple



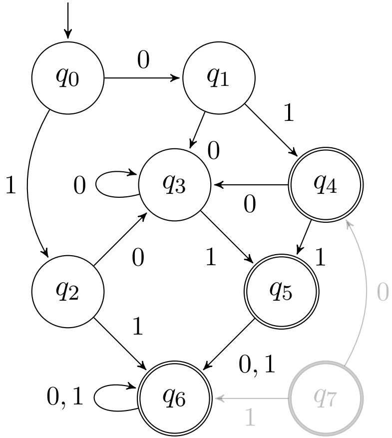
$$
\begin{array}{llllllll}
 & & A & & & B & & \\
\equiv_0 : & \{q_0, & q_1, & q_2\} & \{q_3, & q_4, & q_5, & q_6\} \\
 & AA & AB & BA & BA & AB & BB & BB \\
\equiv_1 : & \{q_0\} & \{q_1\} & \{q_2\} & \{q_3\} & \{q_4\} & \{q_5, & q_6\} \\
 & & & & & & 45 & 45 \\
\equiv_2 : & \{q_0\} & \{q_1\} & \{q_2\} & \{q_3\} & \{q_4\} & \{q_5, & q_6\}
\end{array}
$$

Exemple (solution)



Exercice

Minimiser l'automate suivant :



Suppression de q_7

$\equiv_0 : \{q_0,\ q_1,\ q_2,\ q_3\}\quad \{q_4,\ q_5,\ q_6\}$

$A \qquad B$

$AA\ \ AB\ \ AB\ \ AB \qquad AB\ \ BB\ \ BB$

$\equiv_1 : \{q_0\}\ \{q_1,\ q_2,\ q_3\}\quad \{q_4\}\ \{q_5,\ q_6\}$

$C \quad D \qquad E \quad G$

$DE\ \ DG\ \ DG \qquad GG\ \ GG$

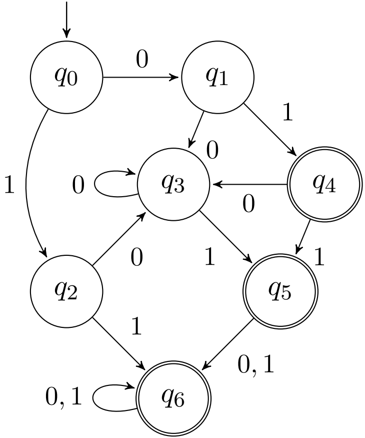
$\equiv_2 : \{q_0\}\ \{q_1\}\ \{q_2,\ q_3\}\quad \{q_4\}\ \{q_5,\ q_6\}$

$C \quad H \quad J \qquad E \quad G$

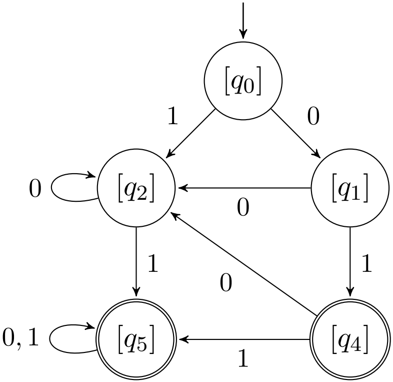
$JG\ \ JG \qquad GG\ \ GG$

$\equiv_3 : \{q_0\}\ \{q_1\}\ \{q_2,\ q_3\}\quad \{q_4\}\ \{q_5,\ q_6\}$

Exercice (suite)



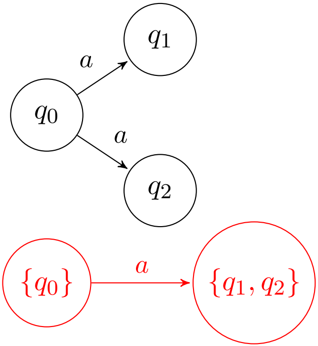
$\equiv : \{q_0\}\ \{q_1\}\ \{q_2, q_3\}\quad \{q_4\}\ \{q_5, q_6\}$



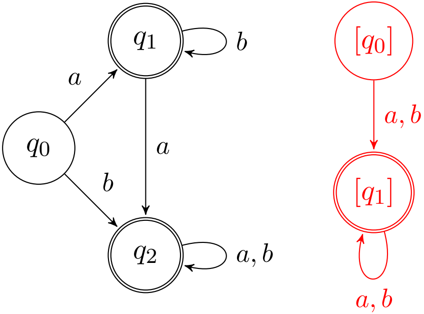
Récapitulatif

1. Suppression des ε-transitions

2. Déterminisation



3. Minimisation



Coûts des transformations

On fixe le vocabulaire et on s'intéresse au nombre d'états $n = |Q|$.

- Suppression des ε-transitions $O(n^2)$
  - Calcul des états accessibles par ε-transitions $O(n^2)$
  - Ajout des nouvelles transitions $O(n^2)$
- Déterminisation $O(2^n)$
  - Calcul des parties accessibles $O(2^n)$
  - Construction de l'automate $O(2^n)$
- Minimisation $O(n^2)$
  - Calcul des relations approchées $O(n^2)$
  - Construction de l'automate $O(n)$

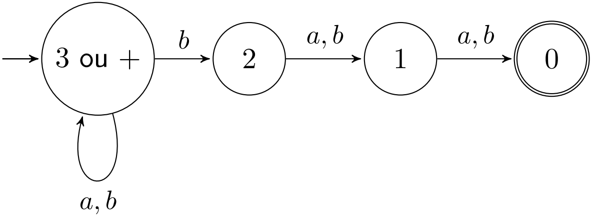
Est-ce que les bornes sont atteintes ? Oui

Déterminisation exponentielle

On considère le langage

$$L = \{w \in \{a, b\}^* \mid \text{le 3}^{\text{e}} \text{ symbole en partant de la fin est un } b\}$$

Un automate non-déterministe qui le reconnaît est :



Après déterminisation

Les 8 états correspondent à toutes les valeurs possibles des 3 derniers symboles lus. L'automate déterminisite a exponentiellement plus d'états.

(et il est minimal)