Théorie des Langages 1

Cours 3 : ε-transitions + déterminisation

L. Rieg

Grenoble INP - Ensimag, 1re année

Année 2022-2023

Une propriété

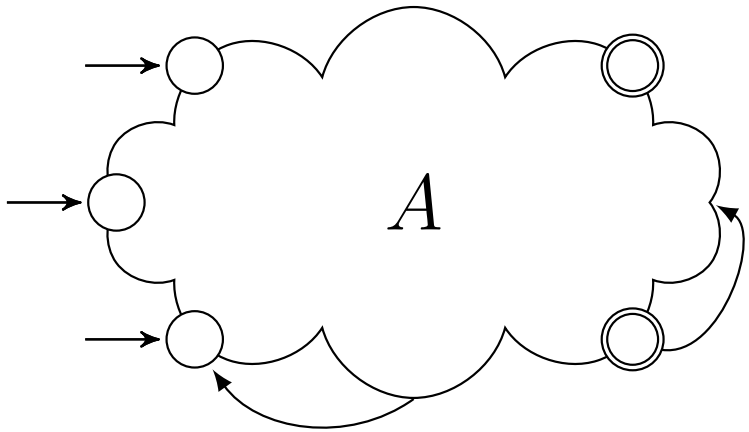
Proposition

Pour tout automate fini A, il existe un automate fini B avec :

- *un unique état initial sans transition entrante,*
- *un unique état final sans transition sortante,*

équivalent à A.

Construction :



Une propriété

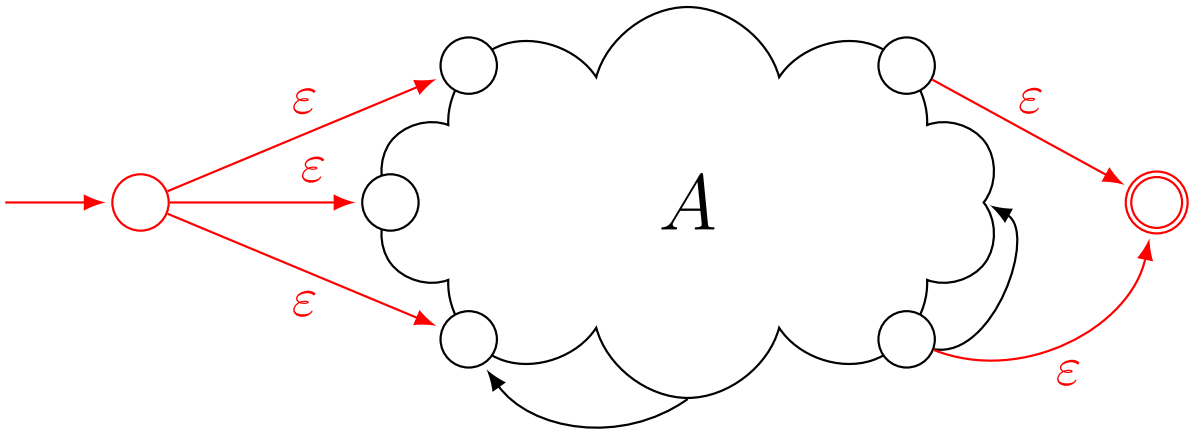
Proposition

Pour tout automate fini A, il existe un automate fini B avec :

- *un unique état initial sans transition entrante,*
- *un unique état final sans transition sortante,*

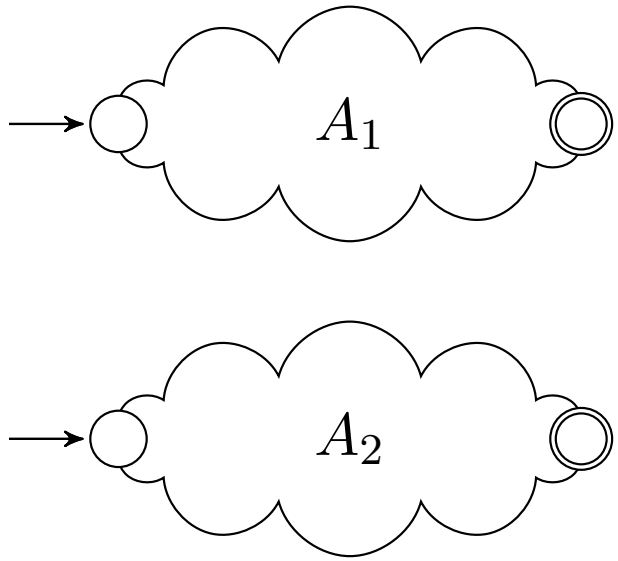
équivalent à A.

Construction :



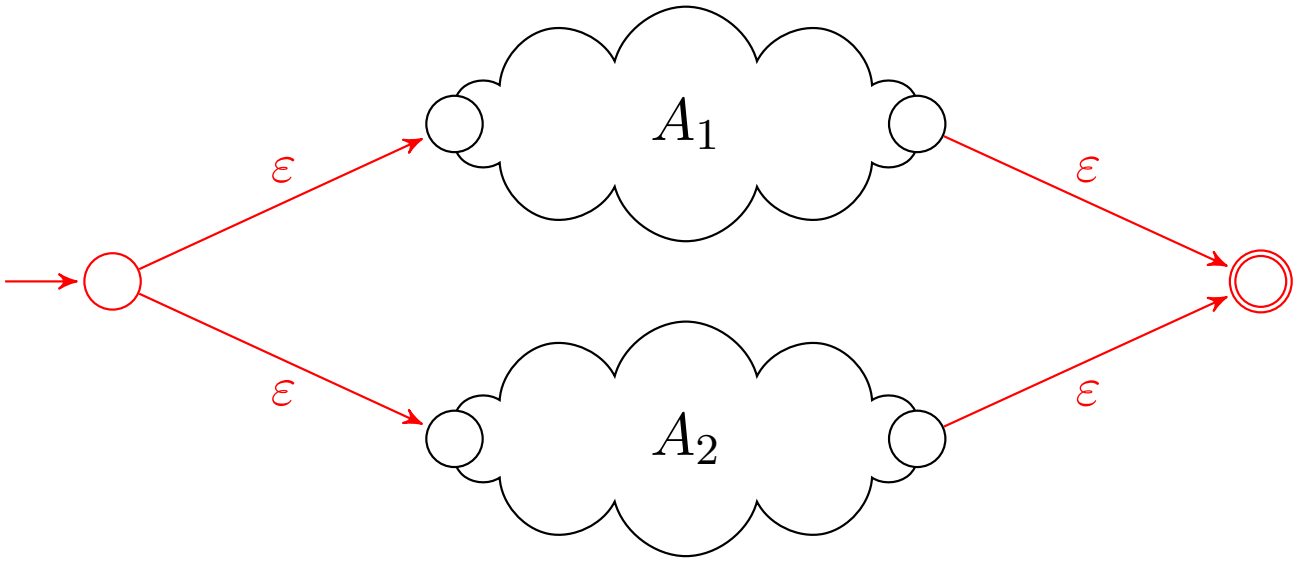
Union

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$.



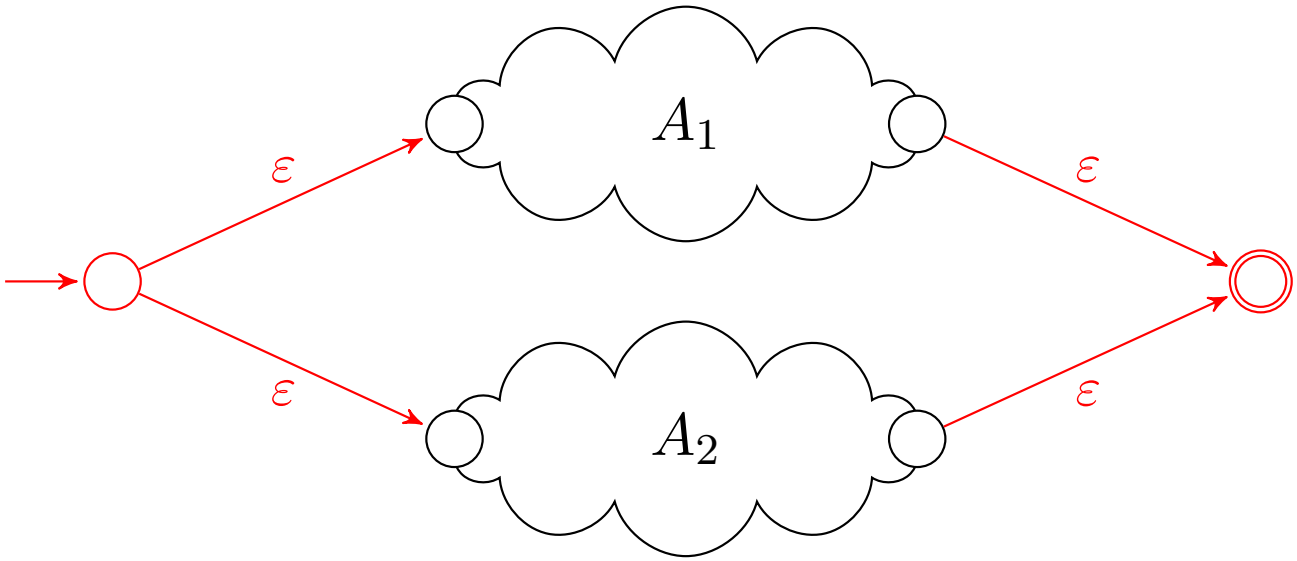
Union

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$.



Union

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$.



Formellement :
Si $A_1 = (Q_1, V, \delta_1, I_1, F_1)$ et $A_2 = (Q_2, V, \delta_2, I_2, F_2)$, on a :

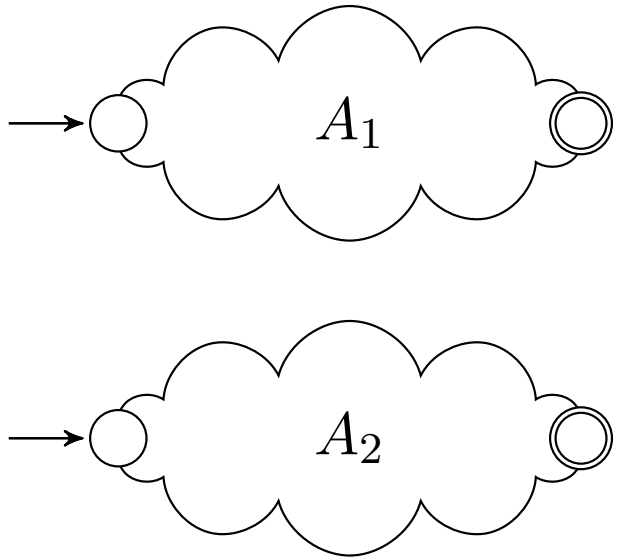
$$A_1 \cup A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2 \uplus \{i, f\}, V, \delta, \{i\}, \{f\})$$

avec

$$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(i, \varepsilon, q) \mid q \in I_1 \cup I_2\} \cup \{(q, \varepsilon, f) \mid q \in F_1 \cup F_2\}$$

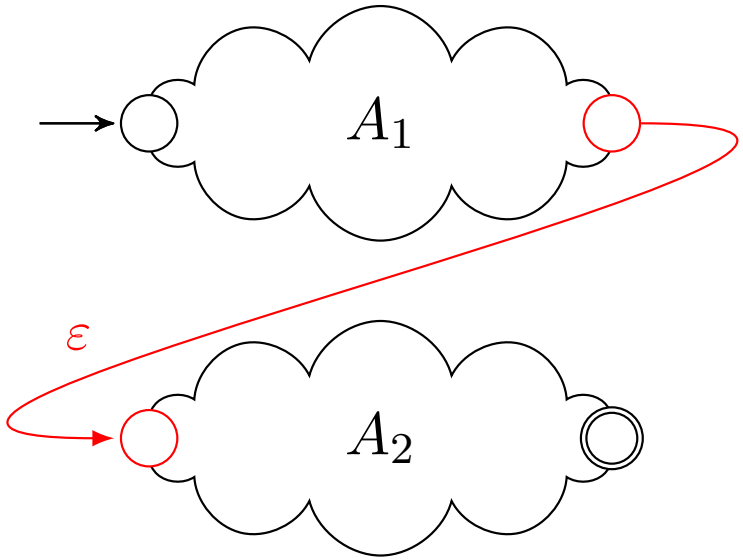
Concaténation

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1)).\mathcal{L}(A_2)$.



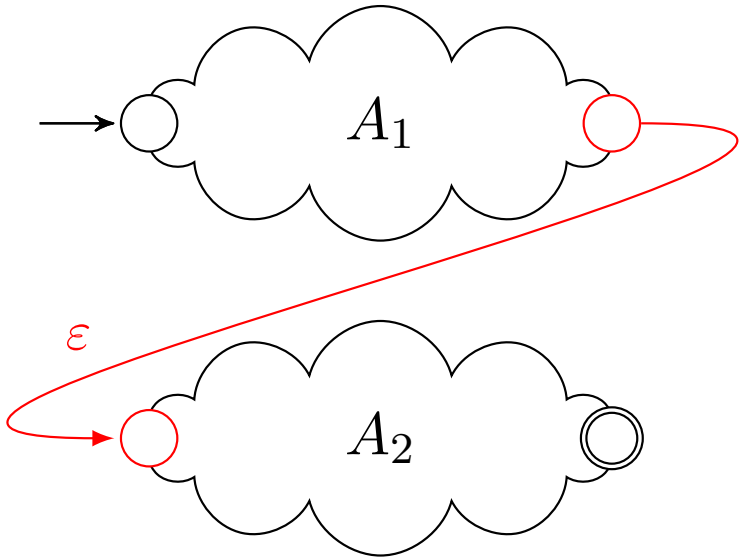
Concaténation

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1)).\mathcal{L}(A_2)$.



Concaténation

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1)).\mathcal{L}(A_2)$.

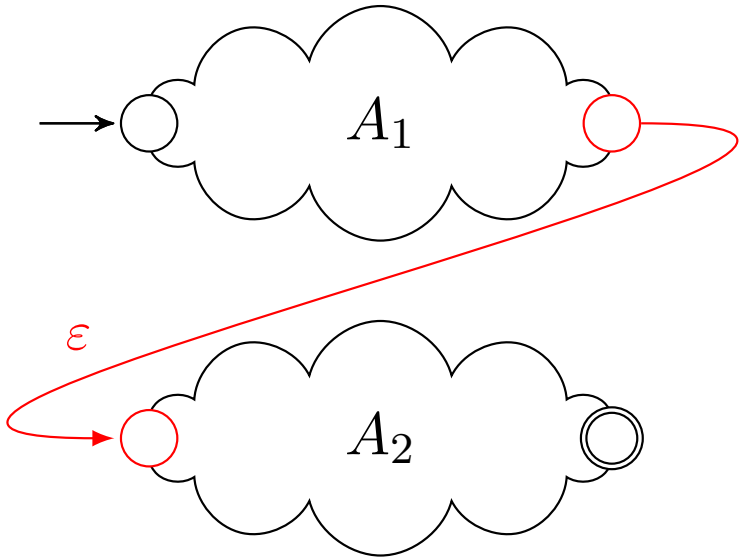


Exercice : écrire formellement cette transformation

Si $A_1 = (Q_1, V, \delta_1, I_1, F_1)$ et $A_2 = (Q_2, V, \delta_2, I_2, F_2)$, on a :

Concaténation

Etant donnés deux automates A_1 et A_2, construire un automate reconnaissant $\mathcal{L}(A_1)).\mathcal{L}(A_2)$.



Exercice : écrire formellement cette transformation

Si $A_1 = (Q_1, V, \delta_1, I_1, F_1)$ et $A_2 = (Q_2, V, \delta_2, I_2, F_2)$, on a :

$$A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$$

avec $$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$$

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$:

- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$:

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient
  $w_1 \in \mathcal{L}(A_1)$
  $w_2 \in \mathcal{L}(A_2)$

- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$:

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient

  $w_1 \in \mathcal{L}(A_1) \iff \exists \chi_1$ chemin de $i_1 \in I_1$ à $f_1 \in F_1$ de trace w_1

  $w_2 \in \mathcal{L}(A_2) \iff \exists \chi_2$ chemin de $i_2 \in I_2$ à $f_2 \in F_2$ de trace w_2

- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$:

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient
  $w_1 \in \mathcal{L}(A_1) \iff \exists \chi_1$ chemin de $i_1 \in I_1$ à $f_1 \in F_1$ de trace w_1
  $w_2 \in \mathcal{L}(A_2) \iff \exists \chi_2$ chemin de $i_2 \in I_2$ à $f_2 \in F_2$ de trace w_2
  Alors $\chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$ est un chemin de i_1 à f_2 dans $A_1.A_2$ de trace $w_1 w_2$. Ainsi, $w_1 w_2 \in \mathcal{L}(A_1.A_2)$.
- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$:

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient
  $w_1 \in \mathcal{L}(A_1) \iff \exists \chi_1$ chemin de $i_1 \in I_1$ à $f_1 \in F_1$ de trace w_1
  $w_2 \in \mathcal{L}(A_2) \iff \exists \chi_2$ chemin de $i_2 \in I_2$ à $f_2 \in F_2$ de trace w_2
  Alors $\chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$ est un chemin de i_1 à f_2 dans $A_1.A_2$ de trace $w_1 w_2$. Ainsi, $w_1 w_2 \in \mathcal{L}(A_1.A_2)$.
- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$: Soit $w \in \mathcal{L}(A_1.A_2)$, donc il existe un chemin dans $A_1.A_2$ de $i \in I_1$ à $f \in F_2$ de trace w.

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient
  $w_1 \in \mathcal{L}(A_1) \iff \exists \chi_1$ chemin de $i_1 \in I_1$ à $f_1 \in F_1$ de trace w_1
  $w_2 \in \mathcal{L}(A_2) \iff \exists \chi_2$ chemin de $i_2 \in I_2$ à $f_2 \in F_2$ de trace w_2
  Alors $\chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$ est un chemin de i_1 à f_2 dans $A_1.A_2$ de trace $w_1 w_2$. Ainsi, $w_1 w_2 \in \mathcal{L}(A_1.A_2)$.

- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$: Soit $w \in \mathcal{L}(A_1.A_2)$, donc il existe un chemin dans $A_1.A_2$ de $i \in I_1$ à $f \in F_2$ de trace w. Par contruction de $A_1.A_2$, ce chemin passe par une ε-transition entre un état $f_1 \in F_1$ et un état $i_2 \in I_2$. Ainsi, on a $\chi = \chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$.

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient
  $w_1 \in \mathcal{L}(A_1) \iff \exists \chi_1$ chemin de $i_1 \in I_1$ à $f_1 \in F_1$ de trace w_1
  $w_2 \in \mathcal{L}(A_2) \iff \exists \chi_2$ chemin de $i_2 \in I_2$ à $f_2 \in F_2$ de trace w_2
  Alors $\chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$ est un chemin de i_1 à f_2 dans $A_1.A_2$ de trace $w_1 w_2$. Ainsi, $w_1 w_2 \in \mathcal{L}(A_1.A_2)$.

- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$: Soit $w \in \mathcal{L}(A_1.A_2)$, donc il existe un chemin dans $A_1.A_2$ de $i \in I_1$ à $f \in F_2$ de trace w. Par contruction de $A_1.A_2$, ce chemin passe par une ε-transition entre un état $f_1 \in F_1$ et un état $i_2 \in I_2$. Ainsi, on a $\chi = \chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$. χ_1 et χ_2 sont acceptants dans A_1 et A_2 d'où $\text{tr}(\chi_1) \in \mathcal{L}(A_1)$ et $\text{tr}(\chi_2) \in \mathcal{L}(A_2)$.

Comment montrer que cette définition est correcte ?

On pose $A_1.A_2 \stackrel{\text{def}}{=} (Q_1 \uplus Q_2, V, \delta, I_1, F_2)$

$\delta \stackrel{\text{def}}{=} \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, i) \mid f \in F_1, i \in I_2\}$

Montrer que $\mathcal{L}(A_1.A_2) = \mathcal{L}(A_1).\mathcal{L}(A_2)$.

Par double inclusion :

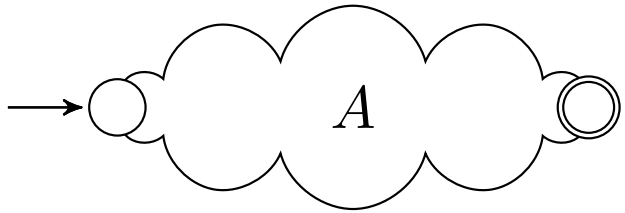
- $\mathcal{L}(A_1).\mathcal{L}(A_2) \subseteq \mathcal{L}(A_1.A_2)$: Soient
  $w_1 \in \mathcal{L}(A_1) \iff \exists \chi_1$ chemin de $i_1 \in I_1$ à $f_1 \in F_1$ de trace w_1
  $w_2 \in \mathcal{L}(A_2) \iff \exists \chi_2$ chemin de $i_2 \in I_2$ à $f_2 \in F_2$ de trace w_2
  Alors $\chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$ est un chemin de i_1 à f_2 dans $A_1.A_2$ de trace $w_1 w_2$. Ainsi, $w_1 w_2 \in \mathcal{L}(A_1.A_2)$.

- $\mathcal{L}(A_1.A_2) \subseteq \mathcal{L}(A_1).\mathcal{L}(A_2)$: Soit $w \in \mathcal{L}(A_1.A_2)$, donc il existe un chemin dans $A_1.A_2$ de $i \in I_1$ à $f \in F_2$ de trace w. Par contruction de $A_1.A_2$, ce chemin passe par une ε-transition entre un état $f_1 \in F_1$ et un état $i_2 \in I_2$. Ainsi, on a $\chi = \chi_1 \cdot (f_1, \varepsilon, i_2) \cdot \chi_2$. χ_1 et χ_2 sont acceptants dans A_1 et A_2 d'où $\text{tr}(\chi_1) \in \mathcal{L}(A_1)$ et $\text{tr}(\chi_2) \in \mathcal{L}(A_2)$.

Exercice : faire de même pour l'union $A_1 \cup A_2$.

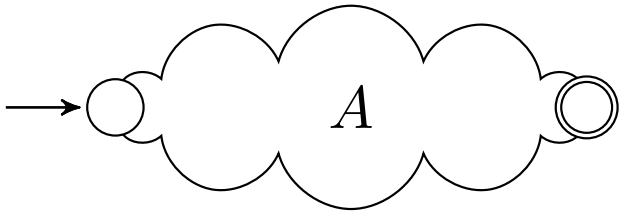
Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

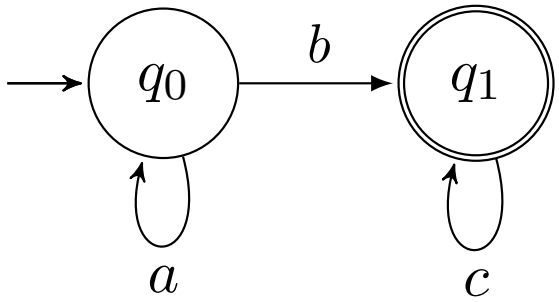


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

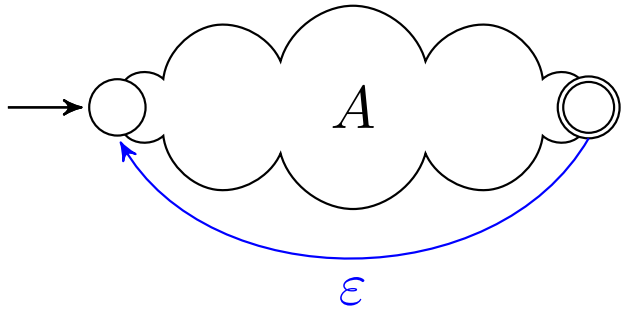


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

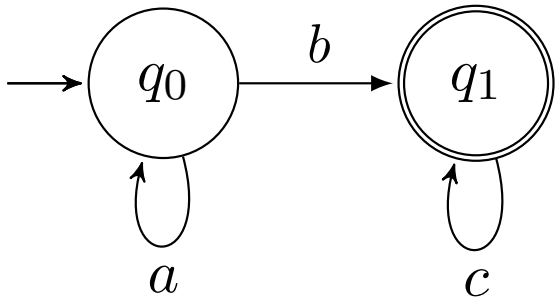


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

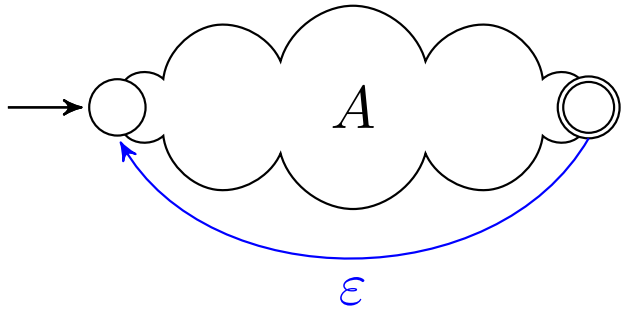


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

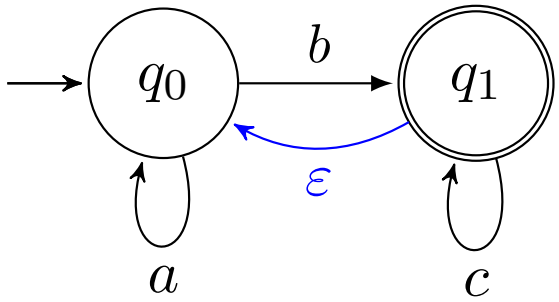


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

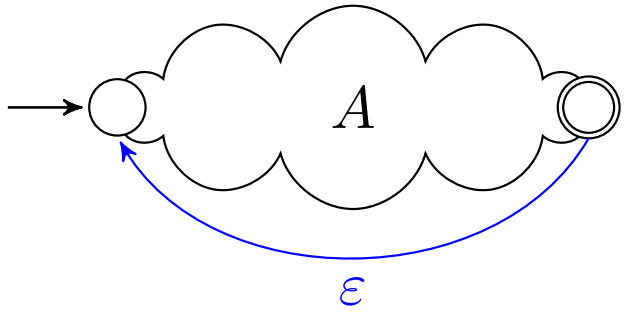


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

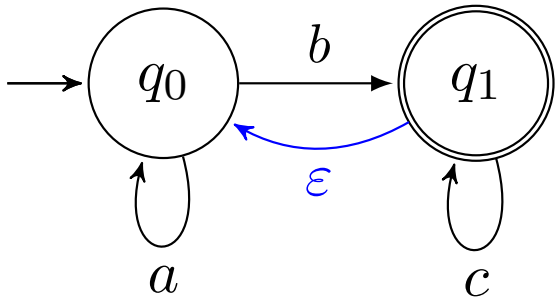


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.



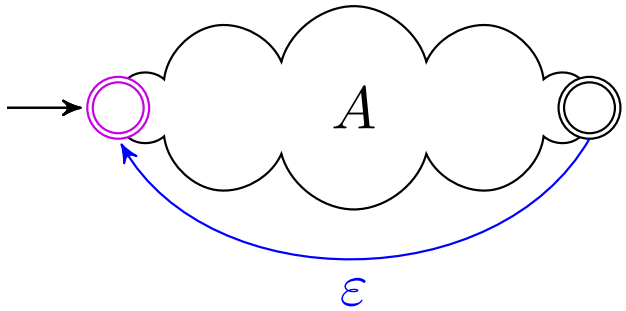
Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$



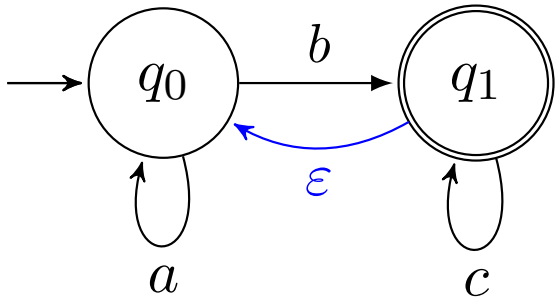
On ne reconnaît pas ε.

Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

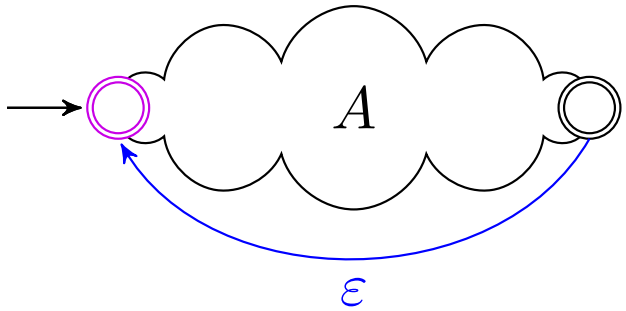


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

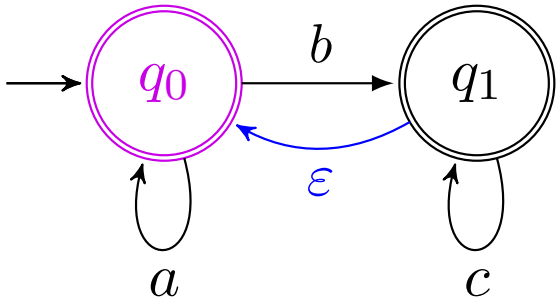


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

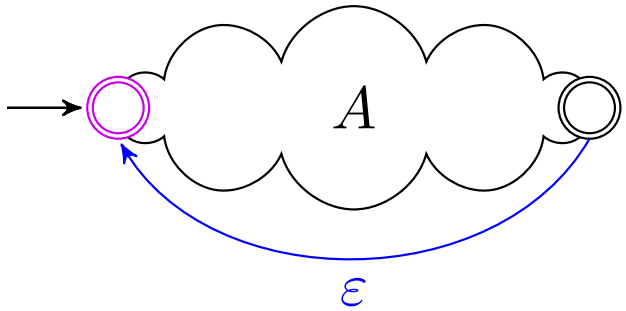


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

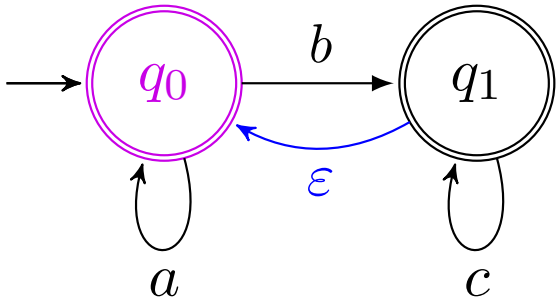


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.



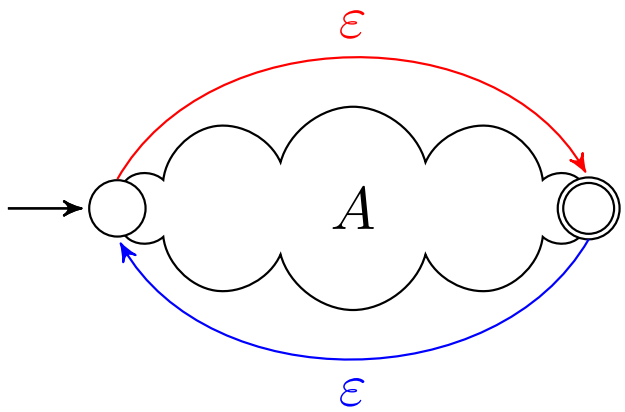
Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$



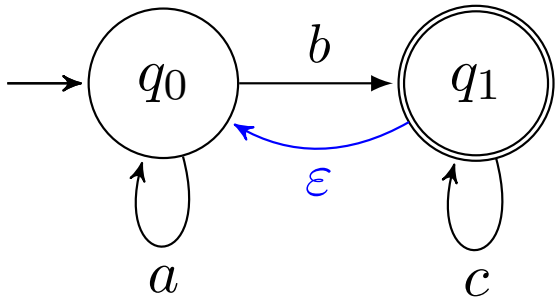
On reconnaît $\{a\}^*$ en trop.

Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

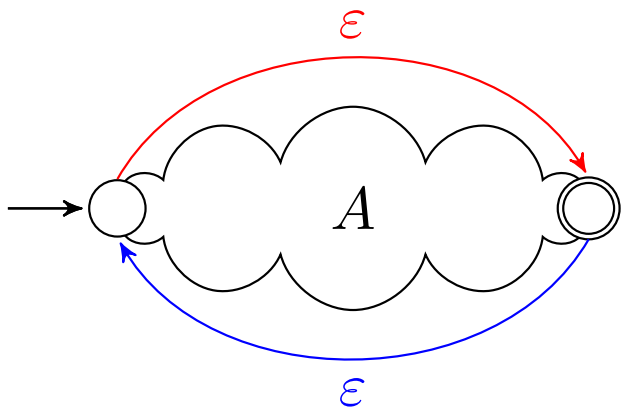


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

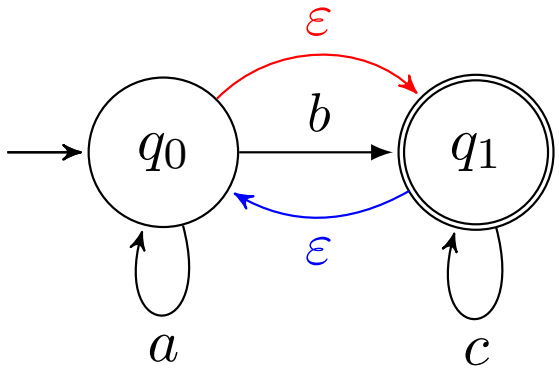


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.

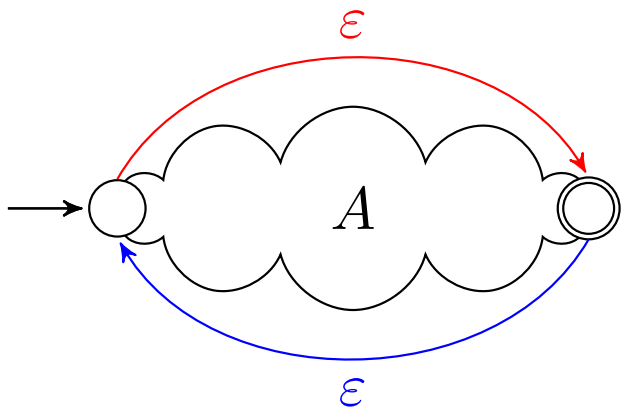


Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$

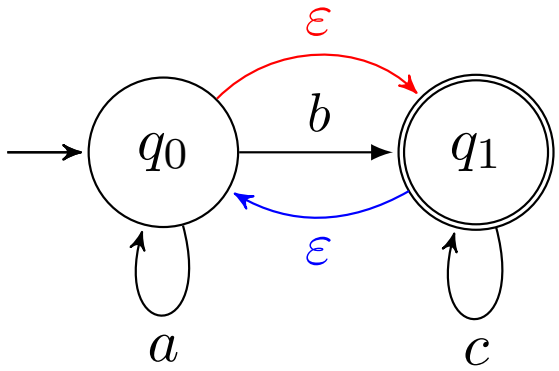


Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.



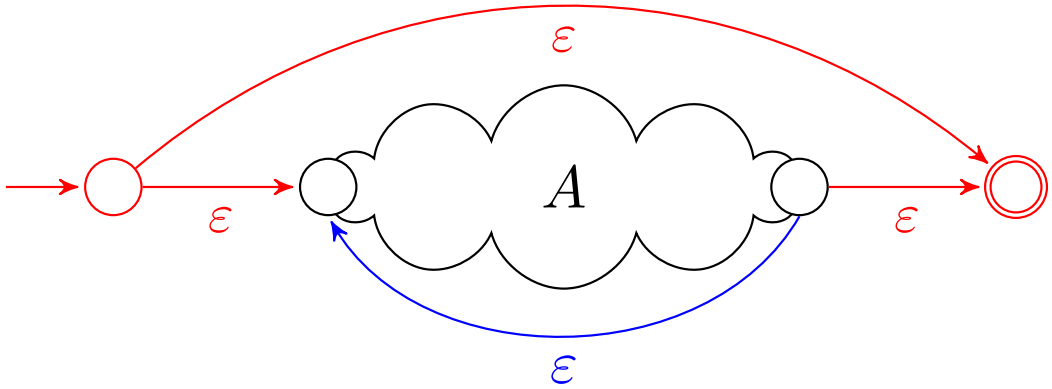
Exemple : $\mathcal{L}(A) = \{a\}^* \{b\} \{c\}^*$



On reconnaît $\{c\}^*$ en trop.

Concaténation itérée

Etant donné un automate A, construire un automate reconnaissant $\mathcal{L}(A)^*$.



Résumé

Théorème

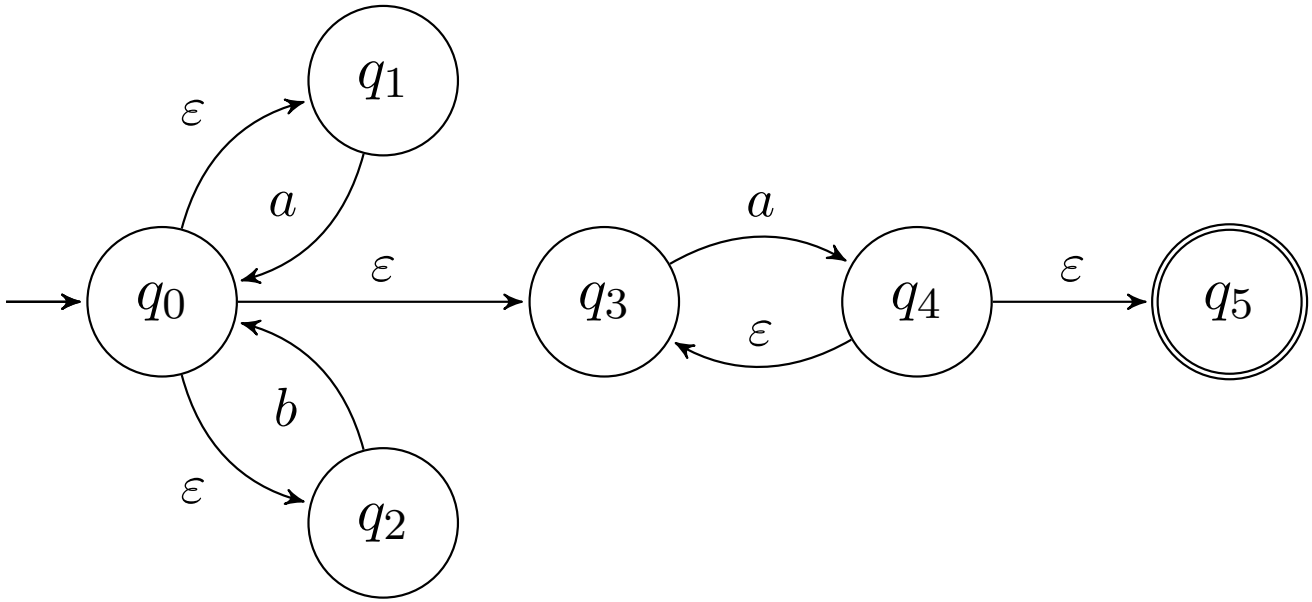
La classe des langages réguliers est fermée :

- *par union*
- *par concaténation*
- *par concaténation itérée*

Nous en verrons d'autres au cours 7.

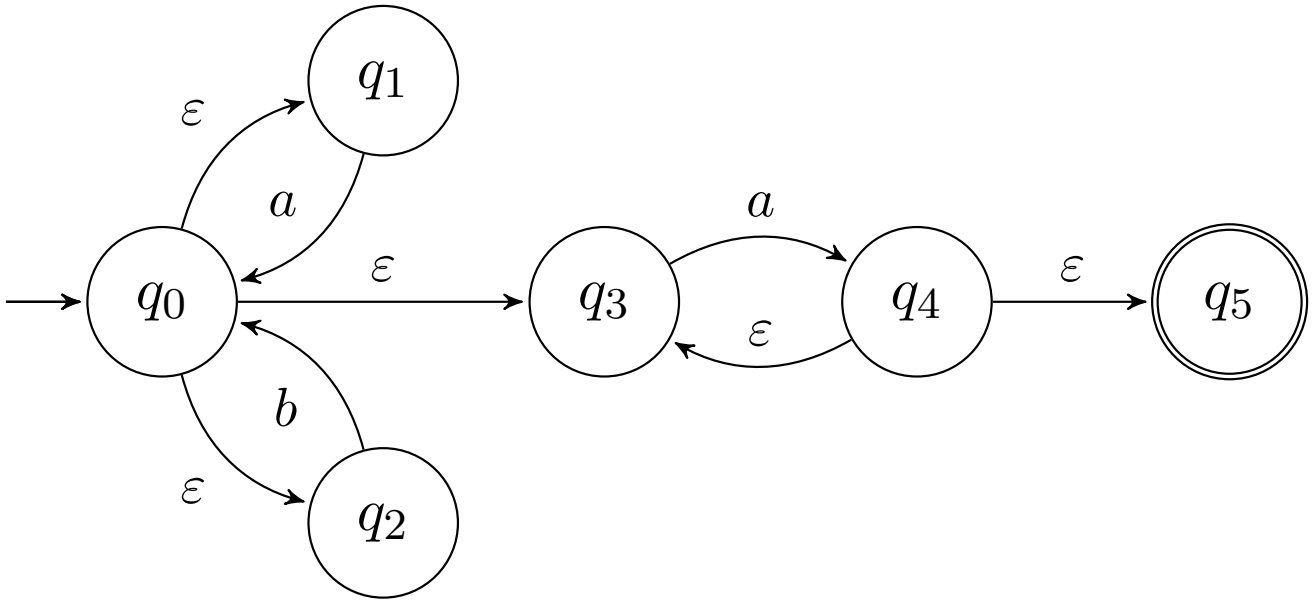
Plusieurs automates pour le même langage

- AFND :

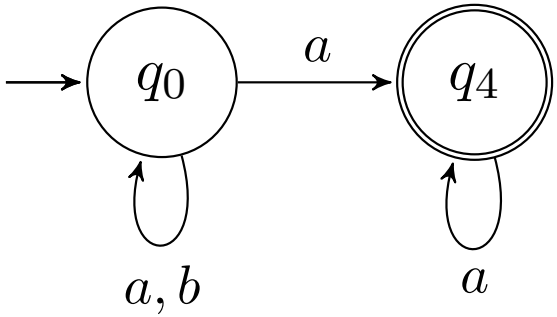


Plusieurs automates pour le même langage

- AFND :

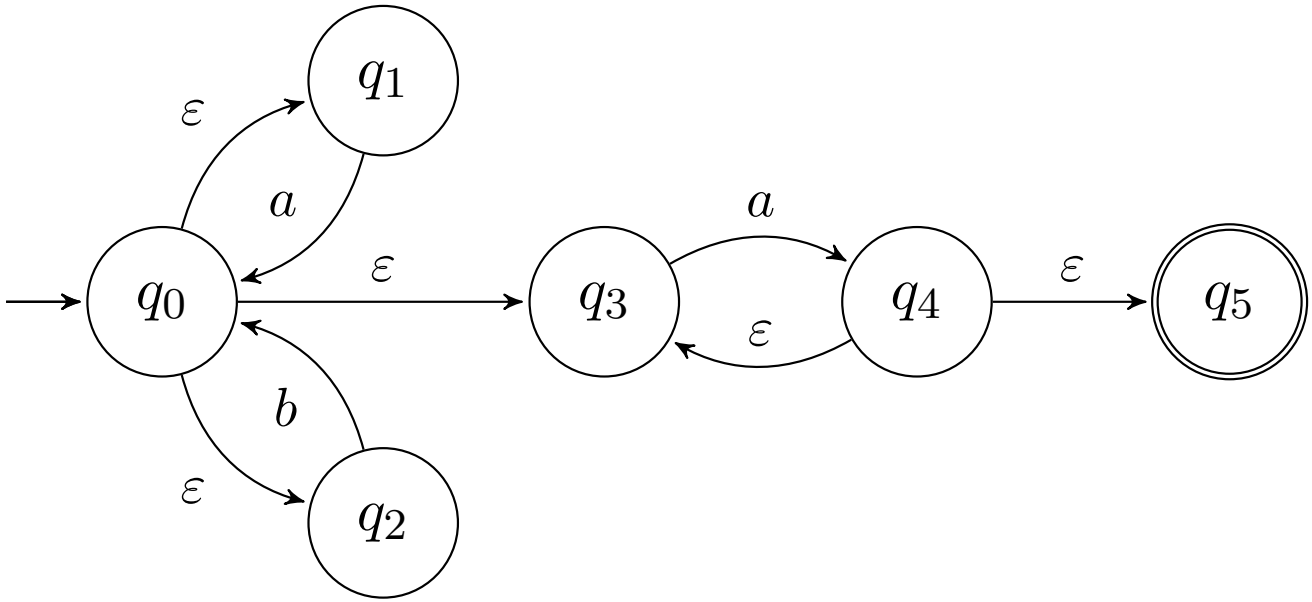


- AFND$-\varepsilon$:

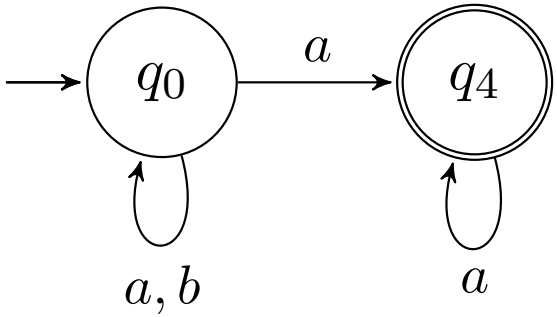


Plusieurs automates pour le même langage

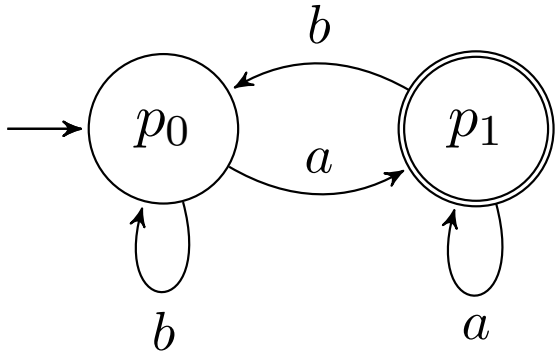
- AFND :



- AFND$-\varepsilon$:



- AFD :



Résumé

Théorème

La classe des langages réguliers est fermée :

- *par union*
- *par concaténation*
- *par concaténation itérée*

Nous en verrons d'autres au cours 7.

Question

Peut-on toujours effectuer les transformations

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

Résumé

Théorème

La classe des langages réguliers est fermée :

- *par union*
- *par concaténation*
- *par concaténation itérée*

Nous en verrons d'autres au cours 7.

Question

Peut-on toujours effectuer les transformations

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

Dans la suite : techniques de transformation, preuves plus tard

Algorithme de suppression des ε-transitions

Deux étapes :

1. Déterminer les états qu'on peut atteindre en ne se servant que d'ε-transitions
2. Se servir de cette information pour construire un automate équivalent sans ε-transition

Algorithme de suppression des ε-transitions

Deux étapes :

1. Déterminer les états qu'on peut atteindre en ne se servant que d'ε-transitions
2. Se servir de cette information pour construire un automate équivalent sans ε-transition

Etape 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 :

Algorithme de suppression des ε-transitions

Deux étapes :

1. Déterminer les états qu'on peut atteindre en ne se servant que d'ε-transitions
2. Se servir de cette information pour construire un automate équivalent sans ε-transition

Etape 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 :

- $p \in \mathrm{Acc}_\varepsilon(p)$

Algorithme de suppression des ε-transitions

Deux étapes :

1. Déterminer les états qu'on peut atteindre en ne se servant que d'ε-transitions
2. Se servir de cette information pour construire un automate équivalent sans ε-transition

Etape 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 $r \in \mathrm{Acc}_\varepsilon(p)$

Algorithme de suppression des ε-transitions

Deux étapes :

1. Déterminer les états qu'on peut atteindre en ne se servant que d'ε-transitions
2. Se servir de cette information pour construire un automate équivalent sans ε-transition

Etape 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 1)

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.

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

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.

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

$$A_0 \stackrel{\text{def}}{=} \{p\}$$

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.

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

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

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.

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

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

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

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.

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

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

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 ?

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.

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

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

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)

Etape 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 :

Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$

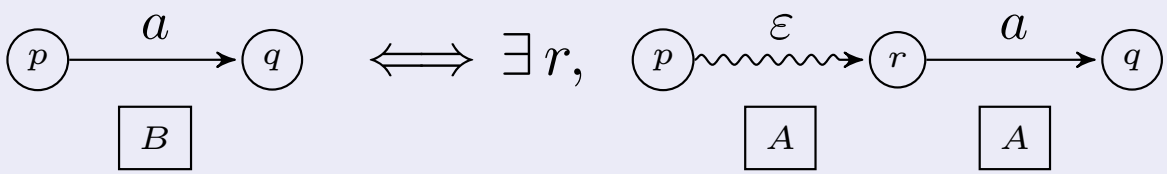
Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$



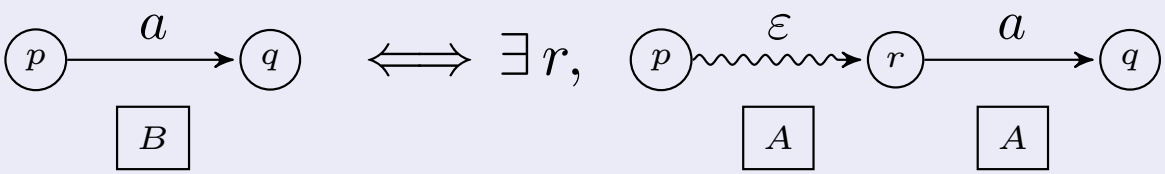
Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$



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

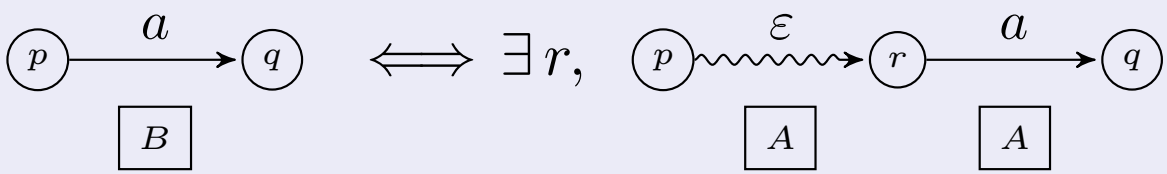
Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$



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

Propositions

- B est un automate sans ε-transition.

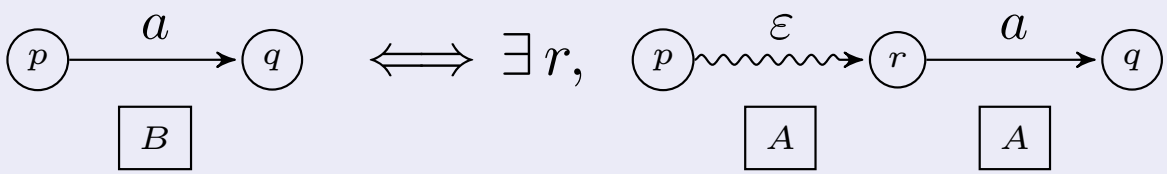
Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$



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

Propositions

- B est un automate sans ε-transition.

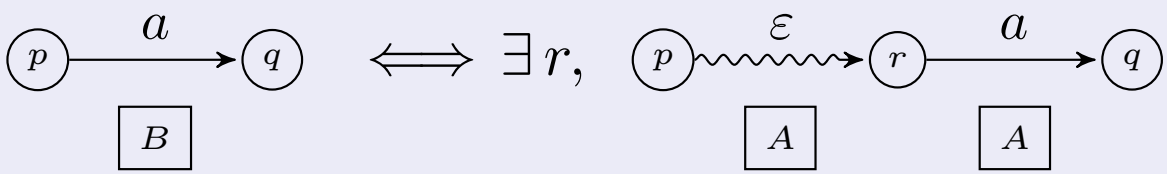
Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$



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

Propositions

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

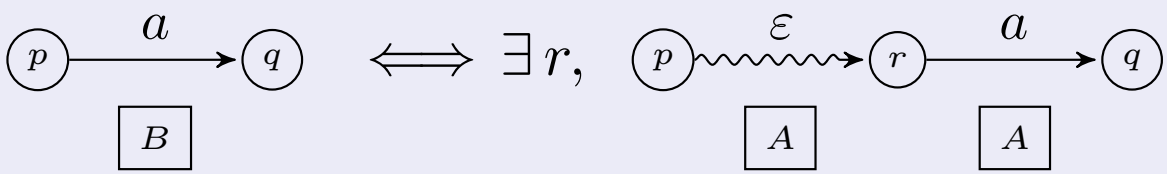
Algorithme de suppression des ε-transitions (suite)

Etape 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 $(r, a, q) \in \delta$



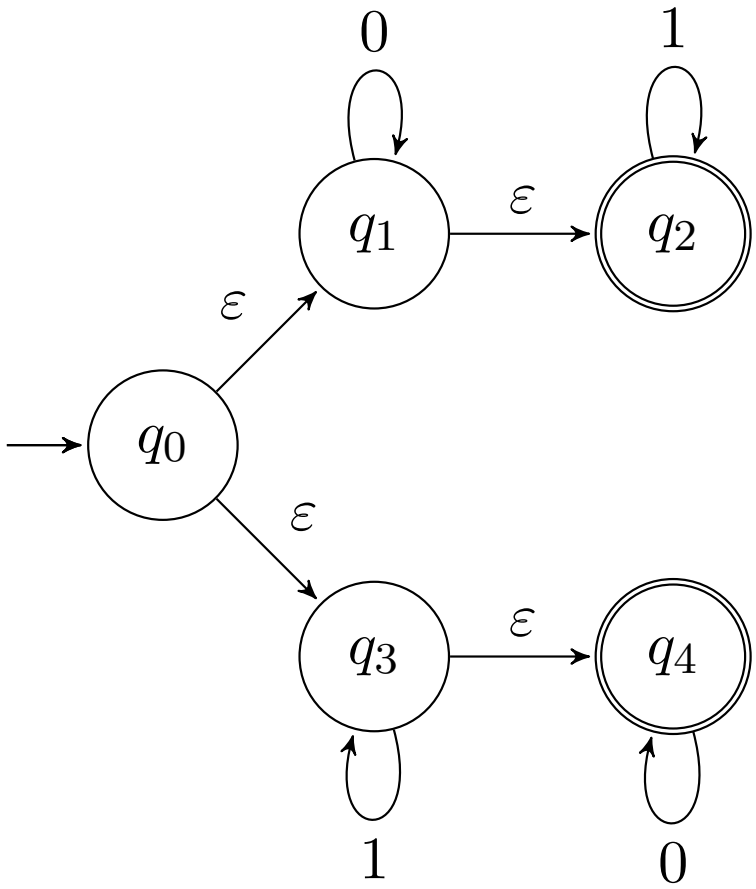
- $F' = \{p \in Q \mid \mathrm{Acc}_\varepsilon(p) \cap F \neq \emptyset\}$

Propositions

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

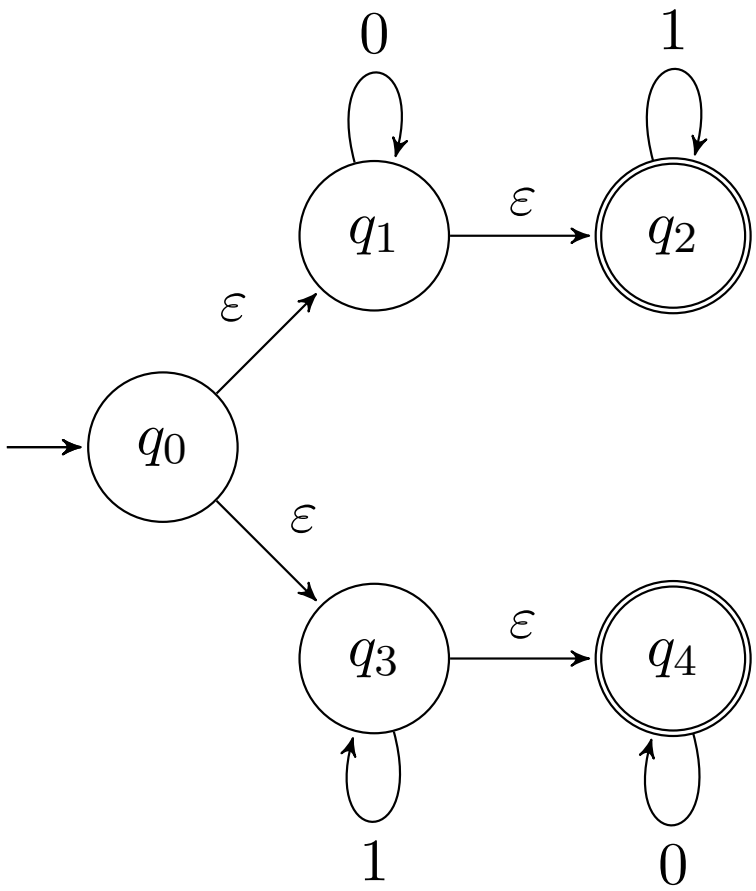
Exercice

Construire B pour l'automate A suivant :



Exercice

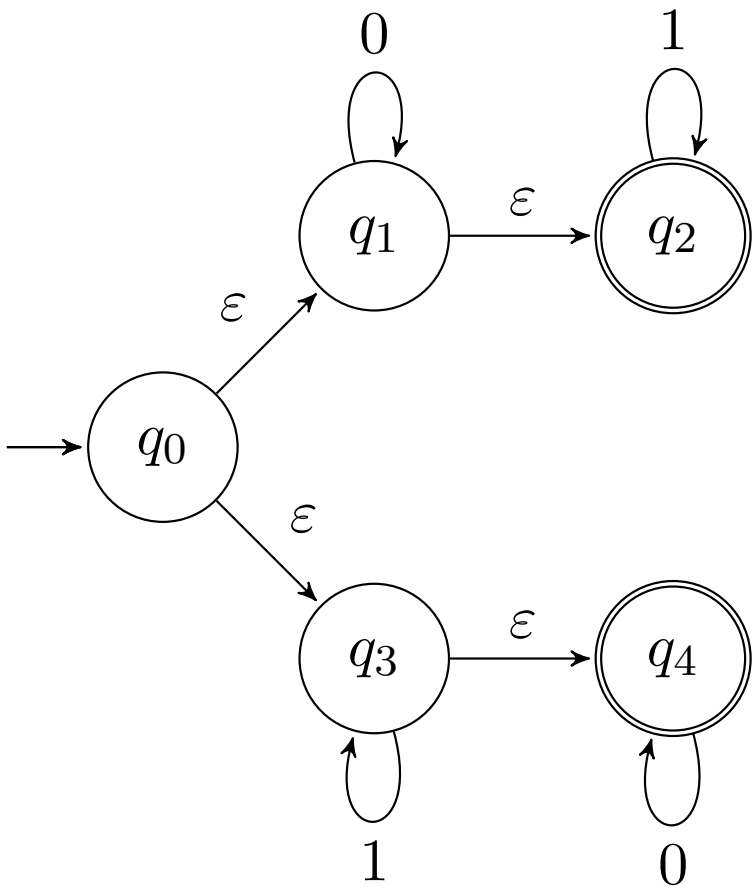
Construire B pour l'automate A suivant :



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

Exercice

Construire B pour l'automate A suivant :

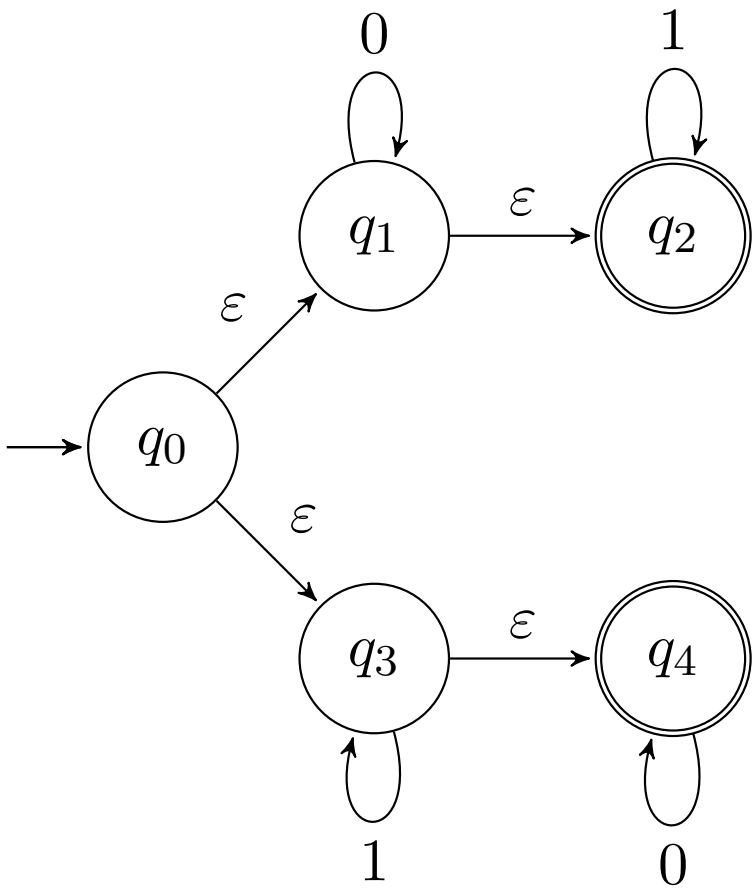


p	$\mathrm{Acc}_\varepsilon(p)$
q_0	q_0, q_1, q_2, q_3, q_4
q_1	q_1, q_2
q_2	q_2
q_3	q_3, q_4
q_4	q_4

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	–

Exercice

Construire B pour l'automate A suivant :



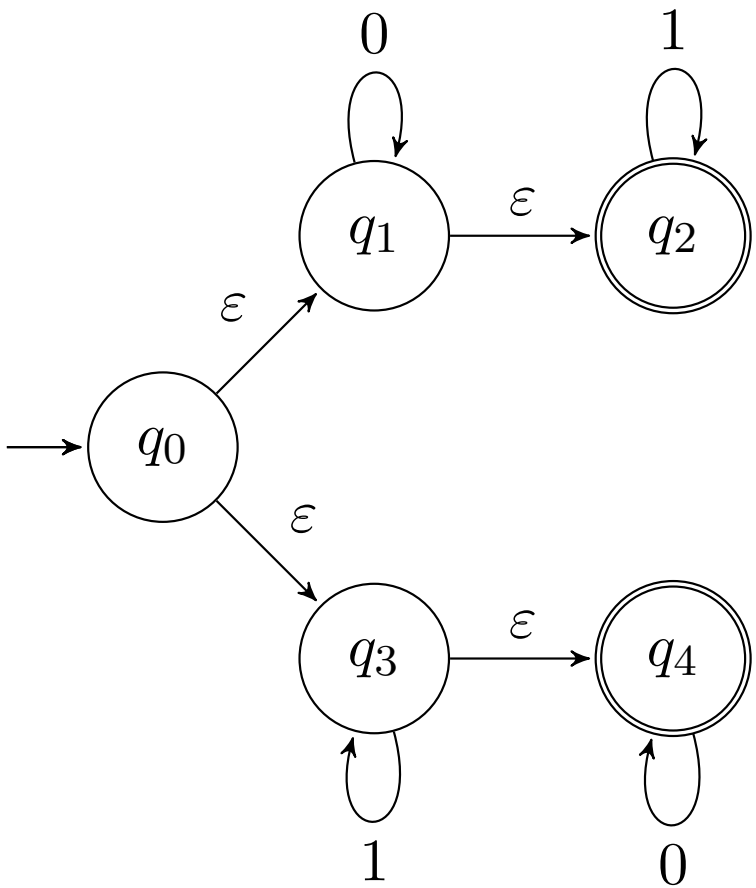
p	$\mathrm{Acc}_\varepsilon(p)$
q_0	q_0, q_1, q_2, q_3, q_4
q_1	q_1, q_2
q_2	q_2
q_3	q_3, q_4
q_4	q_4

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	–

$F' = ?$

Exercice

Construire B pour l'automate A suivant :



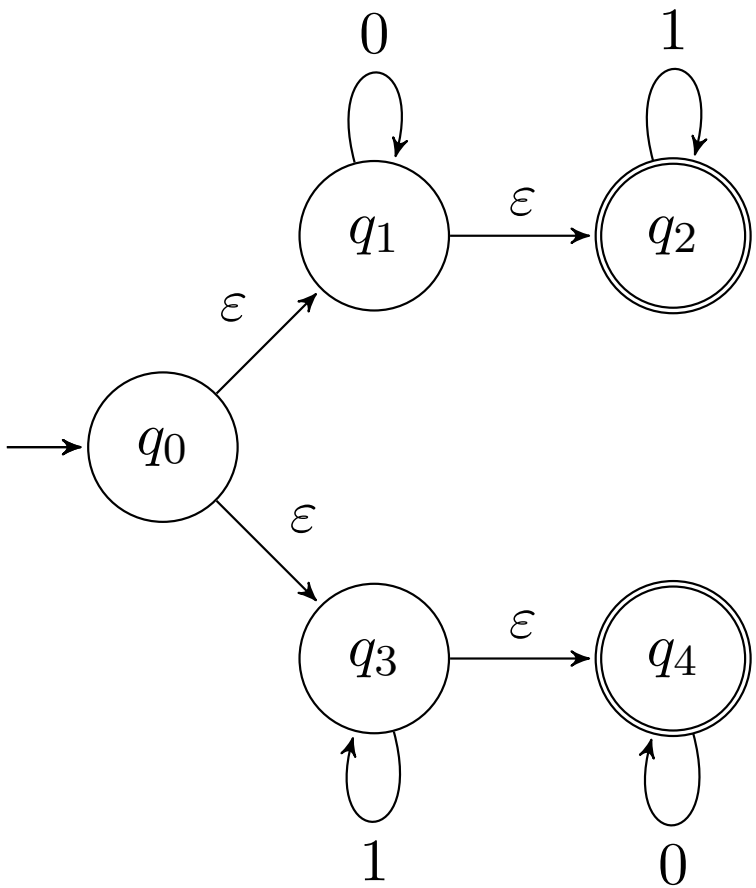
p	$\mathrm{Acc}_\varepsilon(p)$
q_0	q_0, q_1, q_2, q_3, q_4
q_1	q_1, q_2
q_2	q_2
q_3	q_3, q_4
q_4	q_4

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	–

$F' = ?$

Exercice

Construire B pour l'automate A suivant :



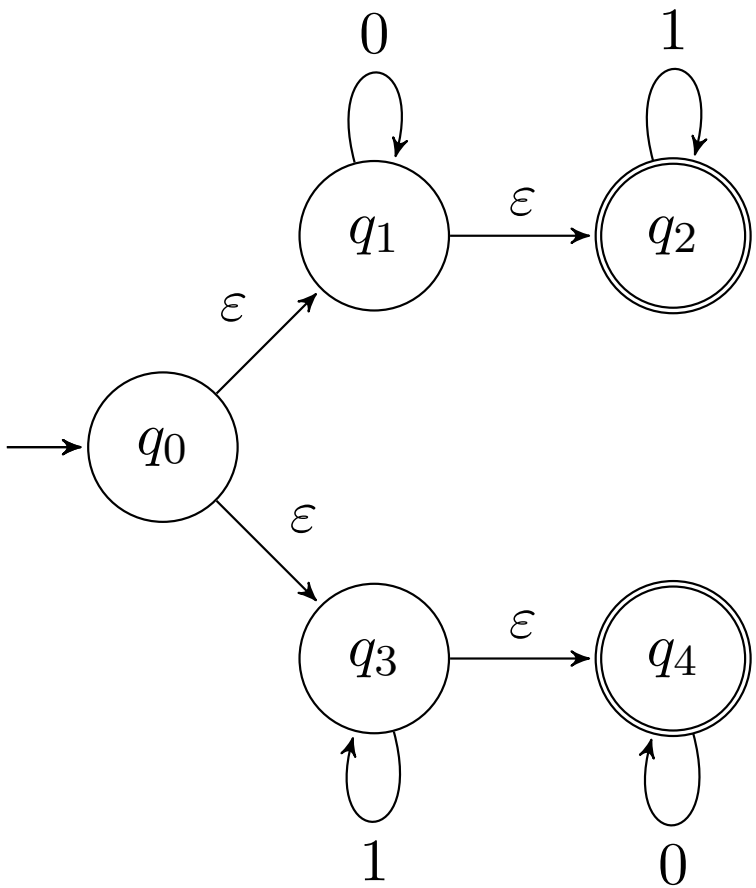
p	$\mathrm{Acc}_\varepsilon(p)$
q_0	q_0, q_1, q_2, q_3, q_4
q_1	q_1, q_2
q_2	q_2
q_3	q_3, q_4
q_4	q_4

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	–

$F' = ?$

Exercice

Construire B pour l'automate A suivant :

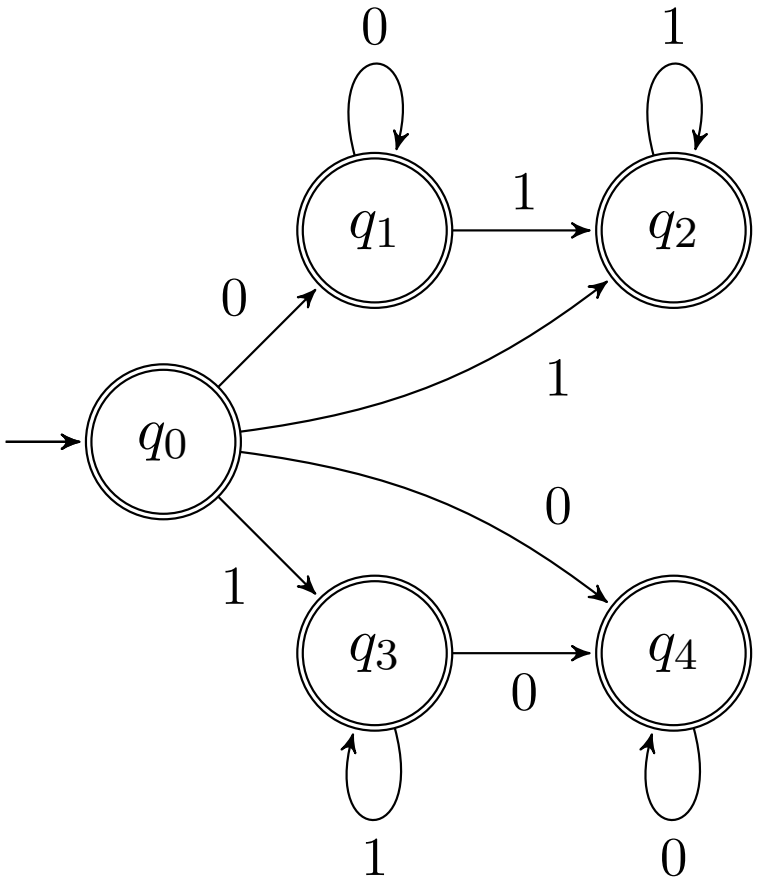


p	$\mathrm{Acc}_\varepsilon(p)$
q_0	q_0, q_1, q_2, q_3, q_4
q_1	q_1, q_2
q_2	q_2
q_3	q_3, q_4
q_4	q_4

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)



Résumé

Théorème

La classe des langages réguliers est fermée :

- *par union*
- *par concaténation*
- *par concaténation itérée*

Nous en verrons d'autres au cours 7.

Question

Peut-on toujours effectuer les transformations

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

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. $\nexists\, (q, \varepsilon, p) \in \delta$
3. $\forall (q, a) \in Q \times V, \exists! p \in Q, (q, a, p) \in \delta.$

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. $\nexists\, (q, \varepsilon, p) \in \delta$
3. $\forall (q, a) \in Q \times V, \exists! p \in Q, (q, a, p) \in \delta.$

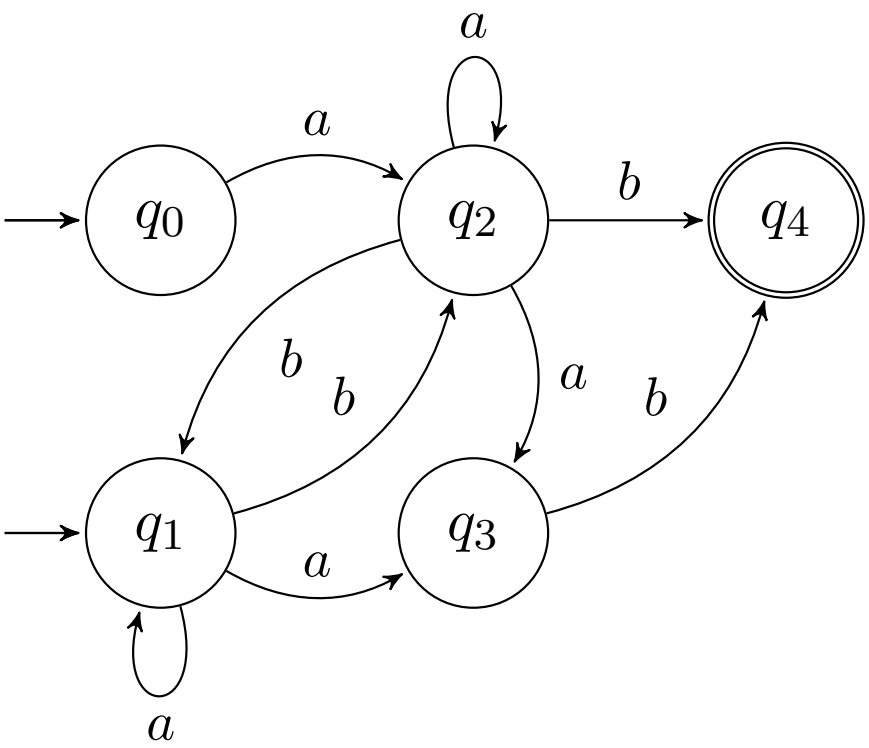
Conséquences

- 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 en parallèle

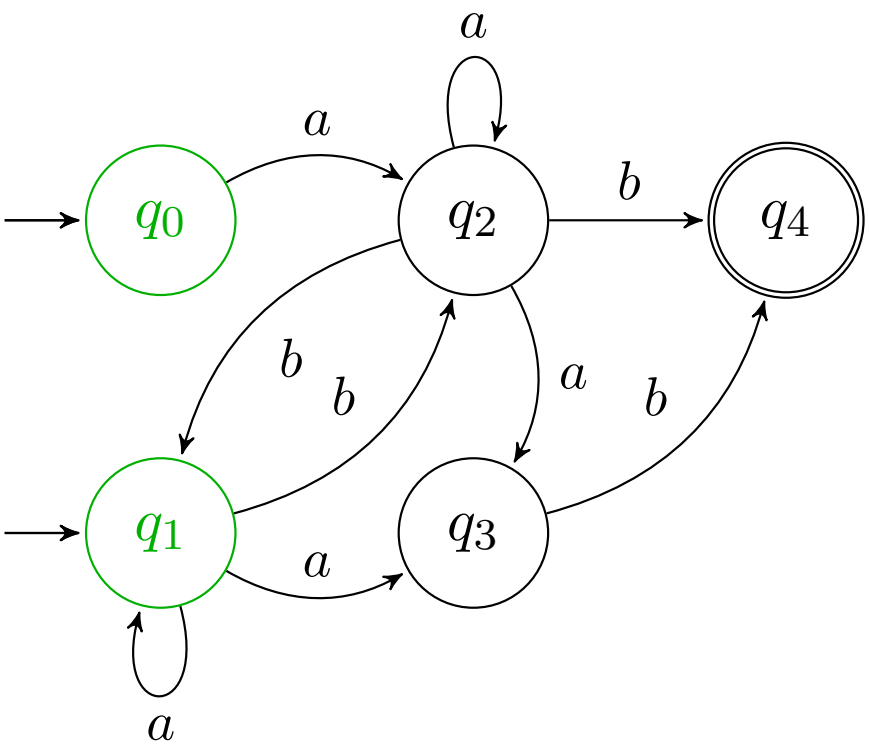
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

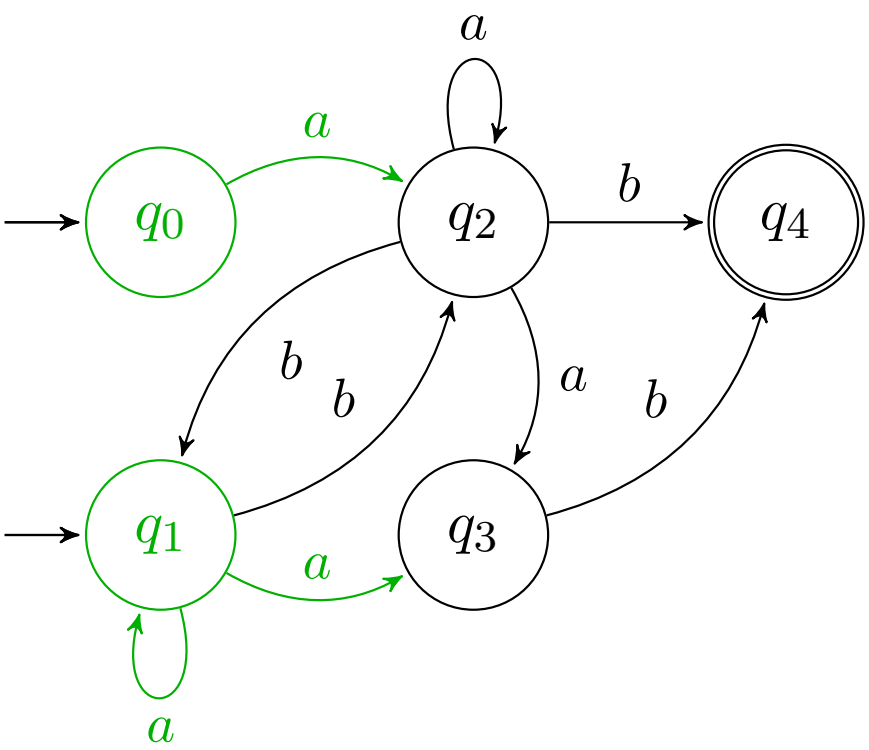
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

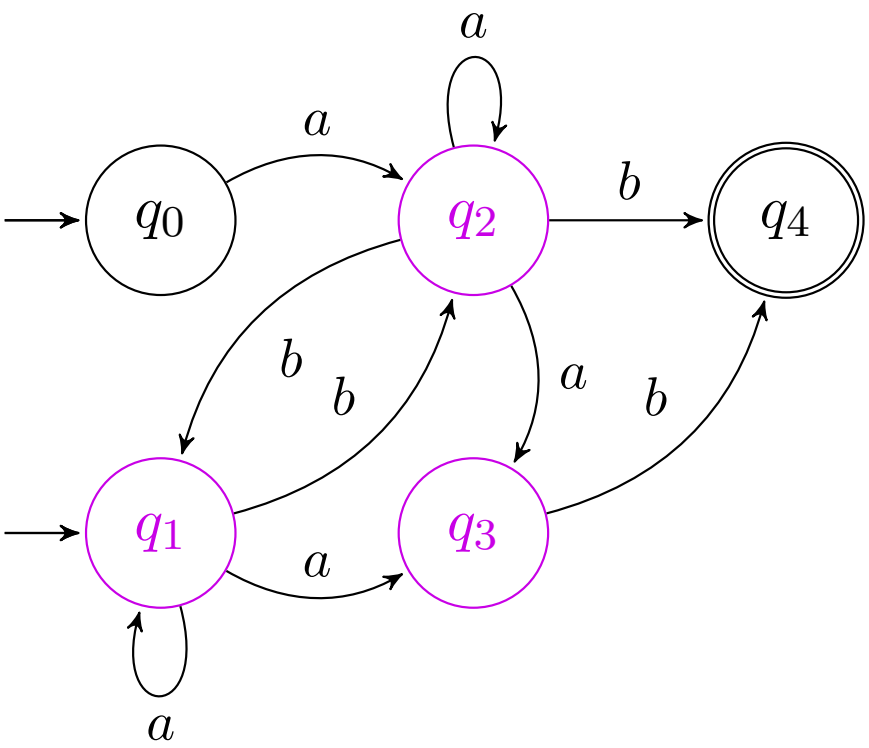
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

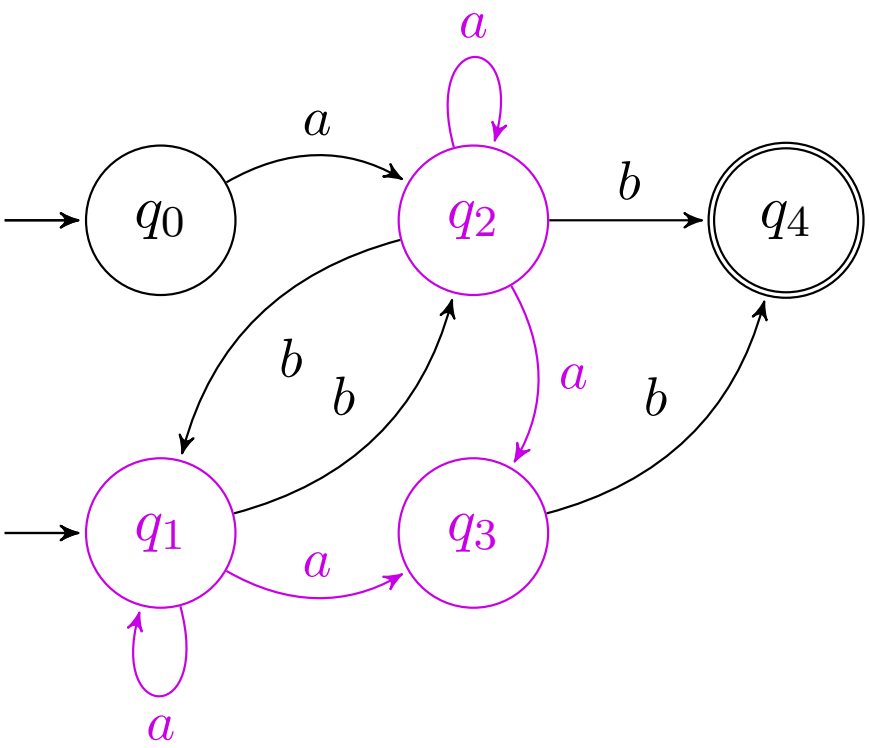
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

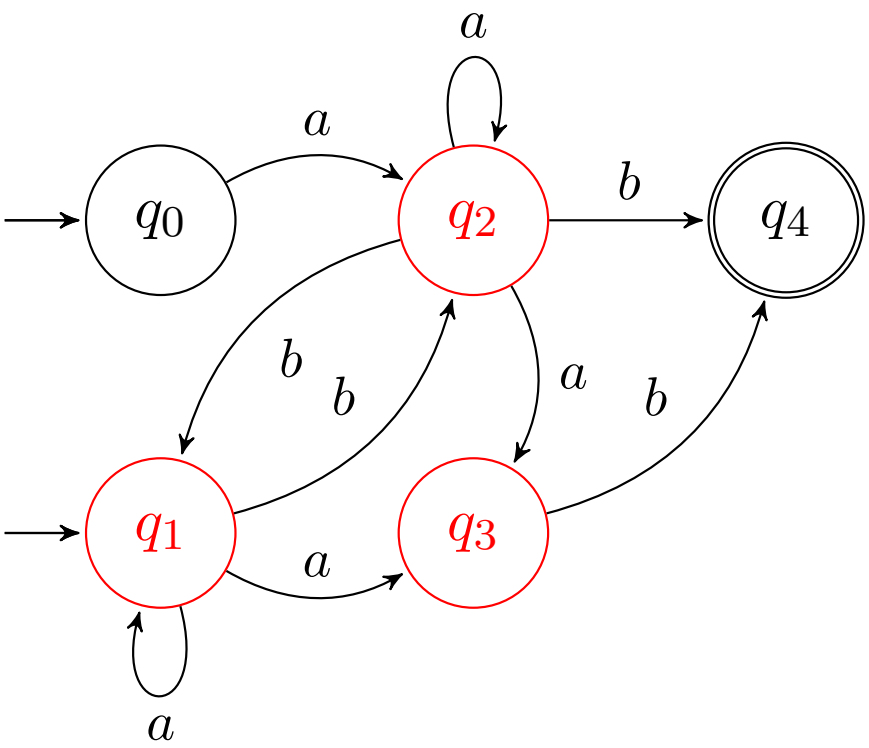
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

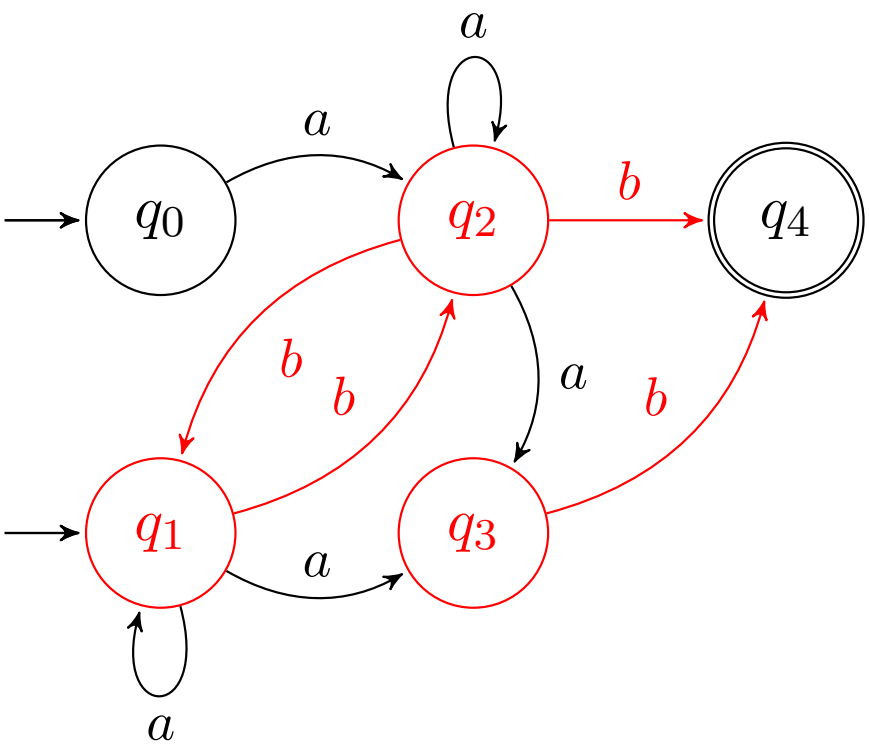
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

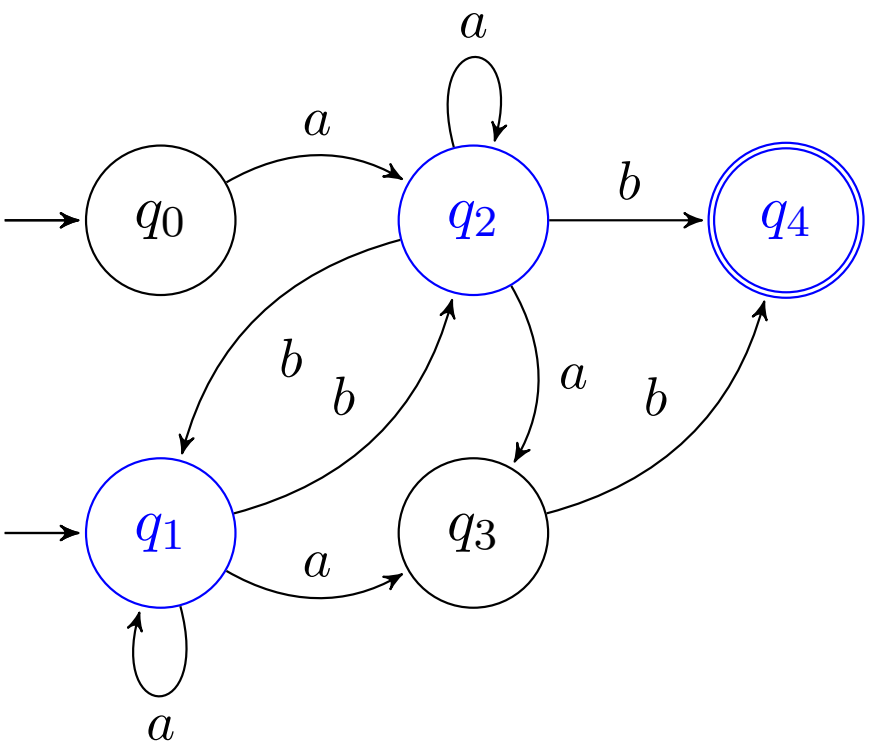
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

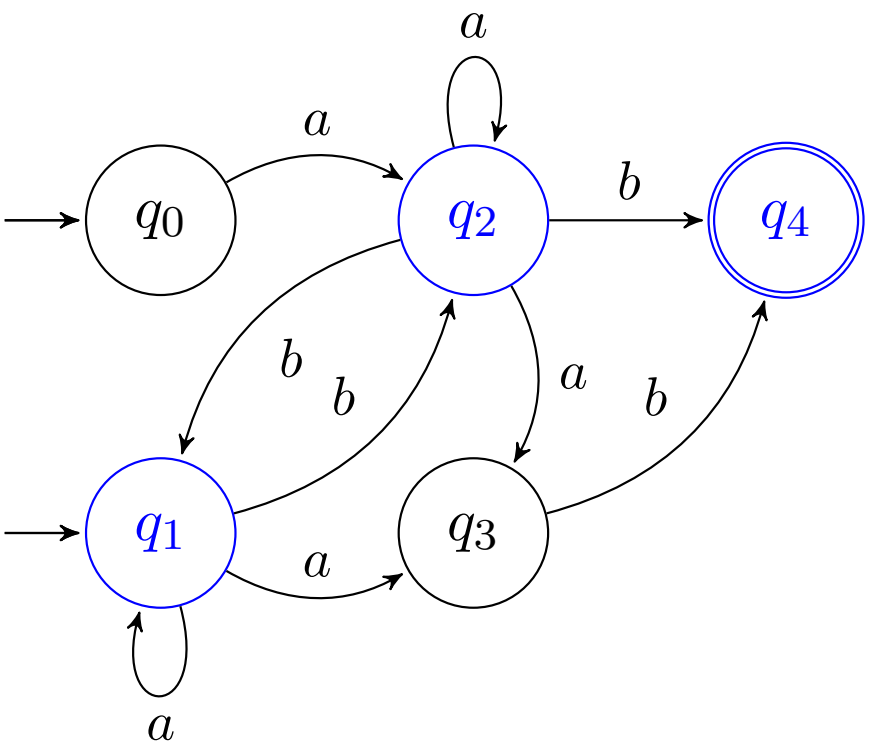
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

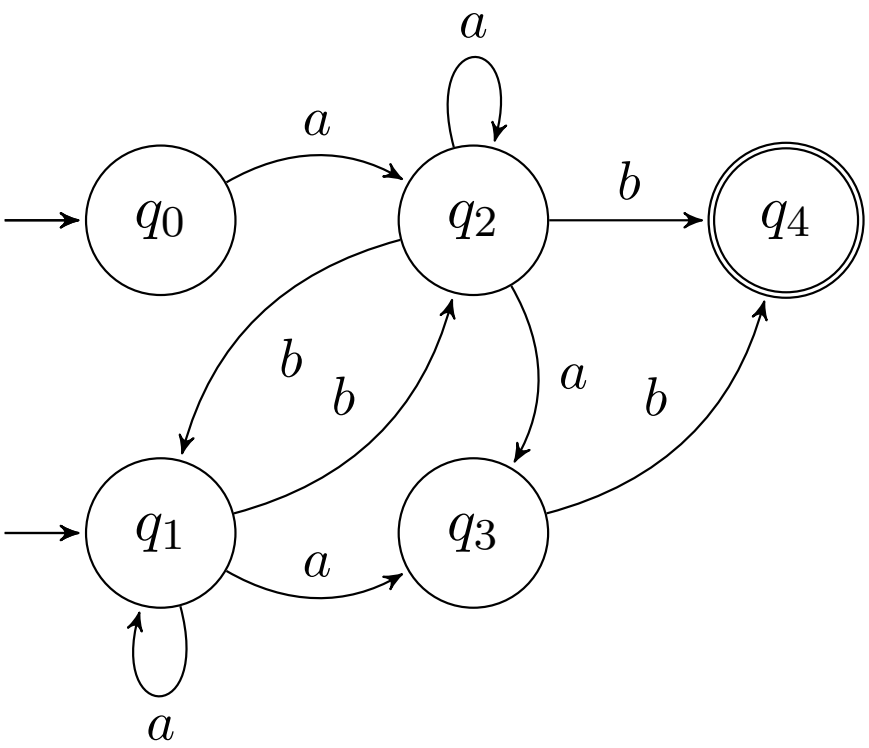


OK : $aab \in \mathcal{L}(A)$

Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

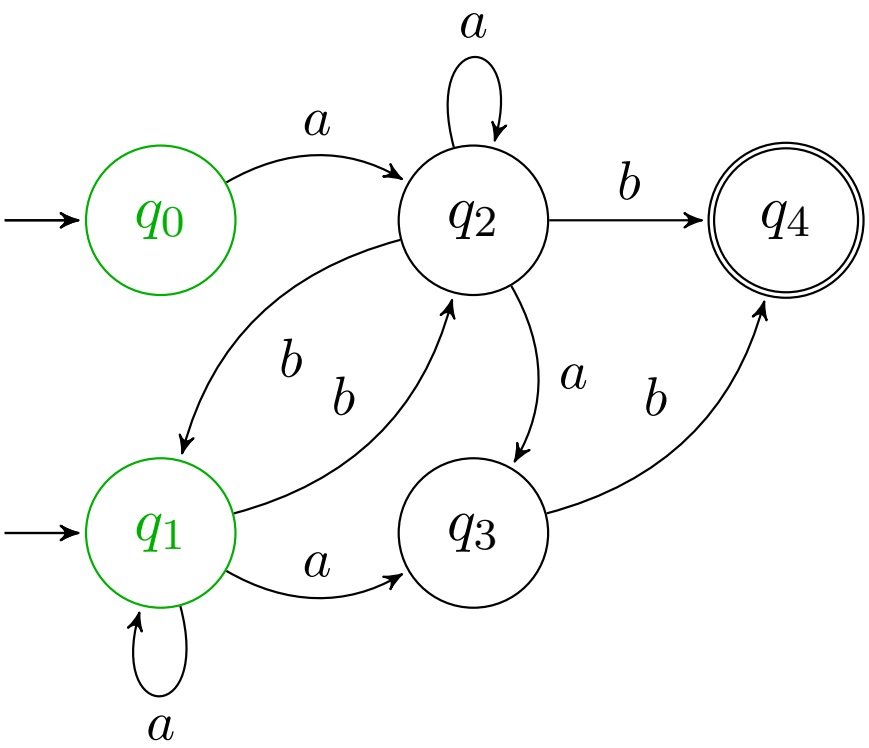
Exemple : L'automate suivant reconnaît-il aab ?



Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

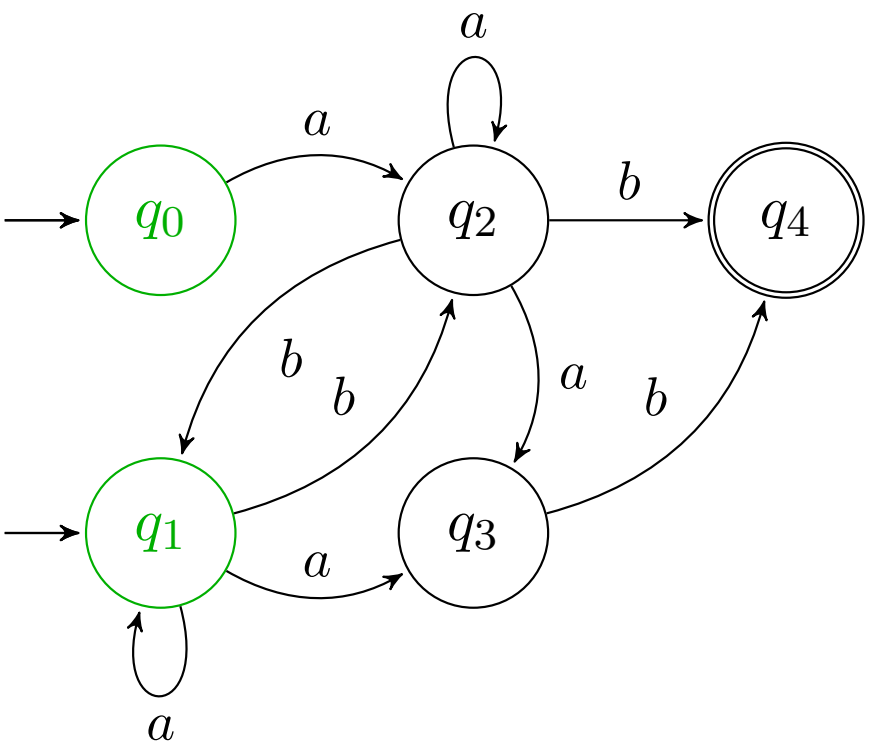
Exemple : L'automate suivant reconnaît-il aab ?

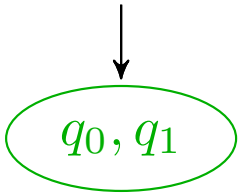


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

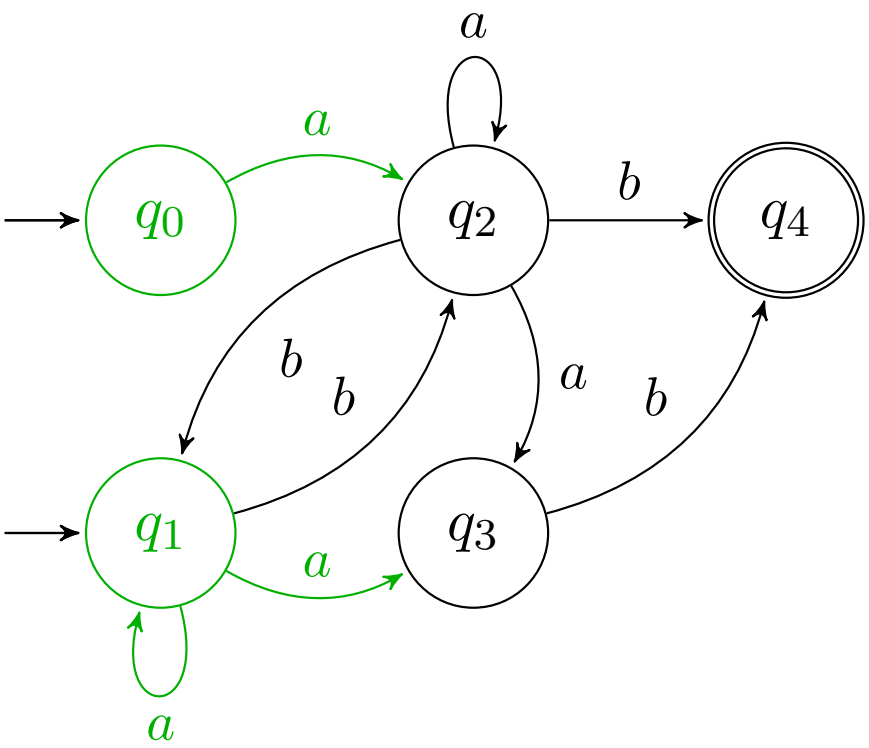


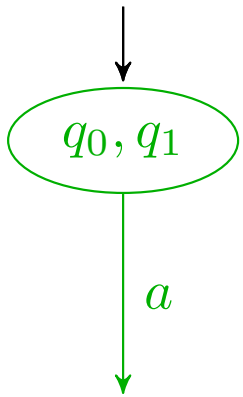


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

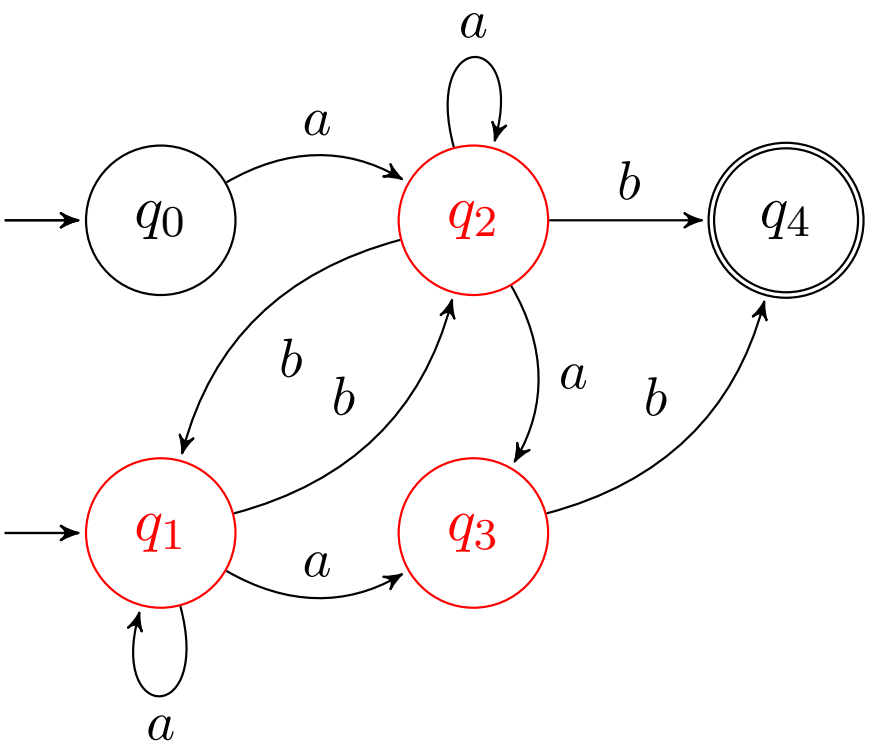


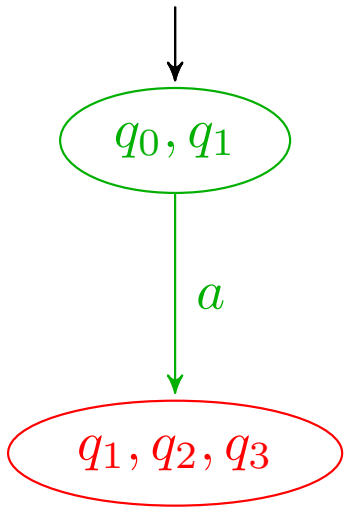


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

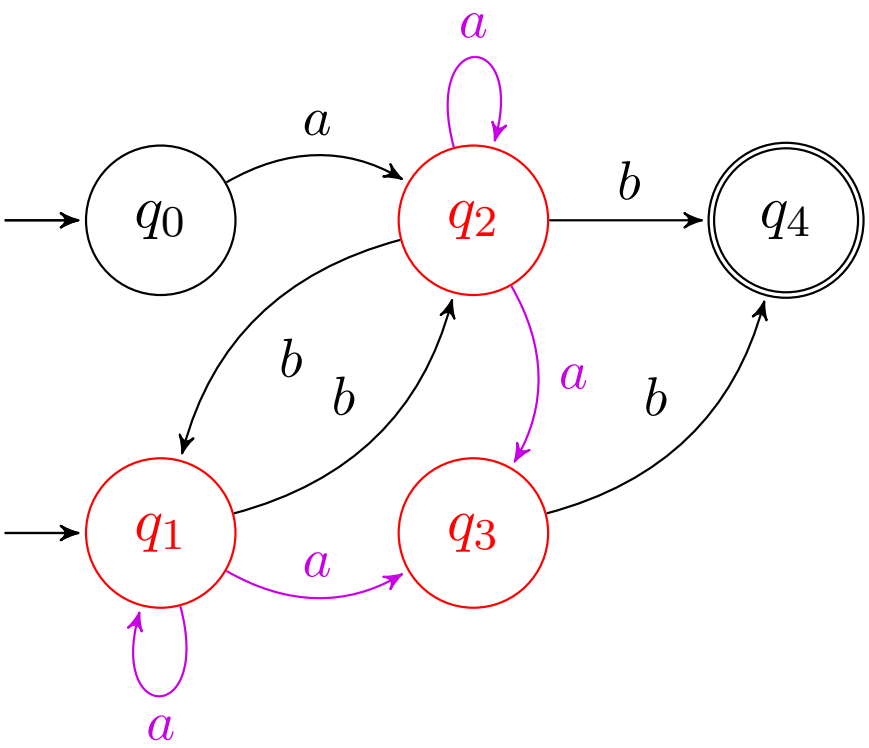


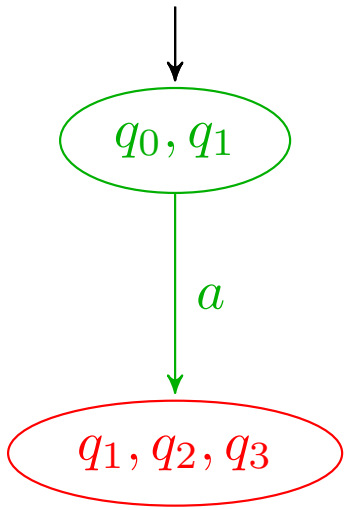


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

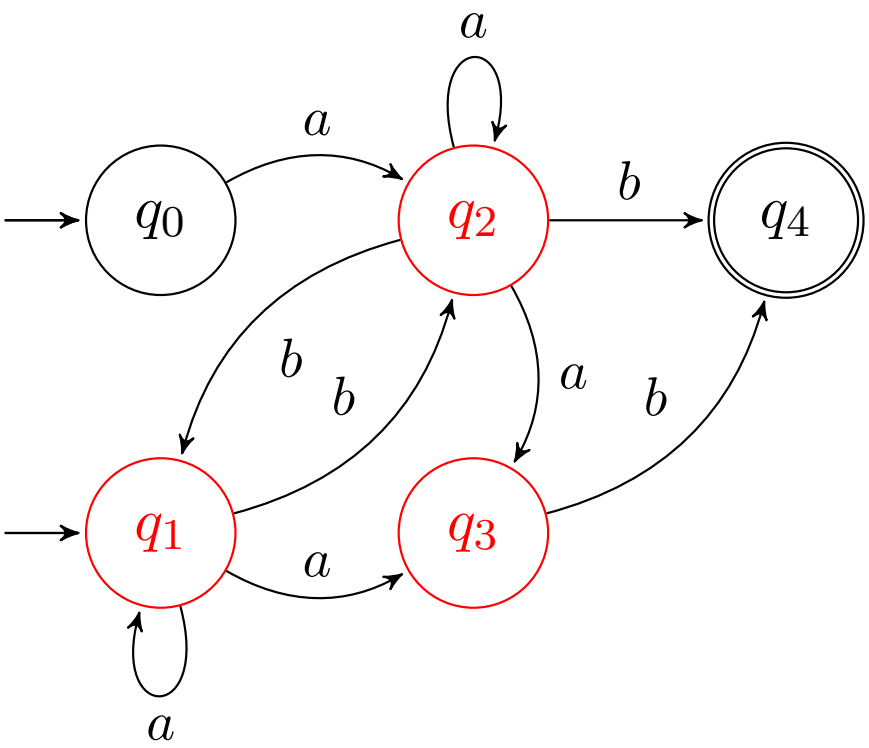


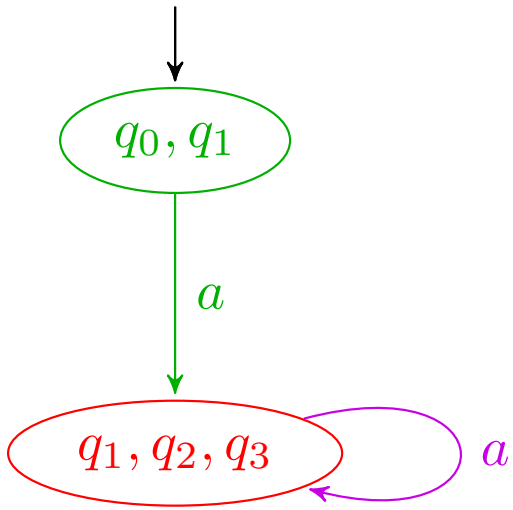


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

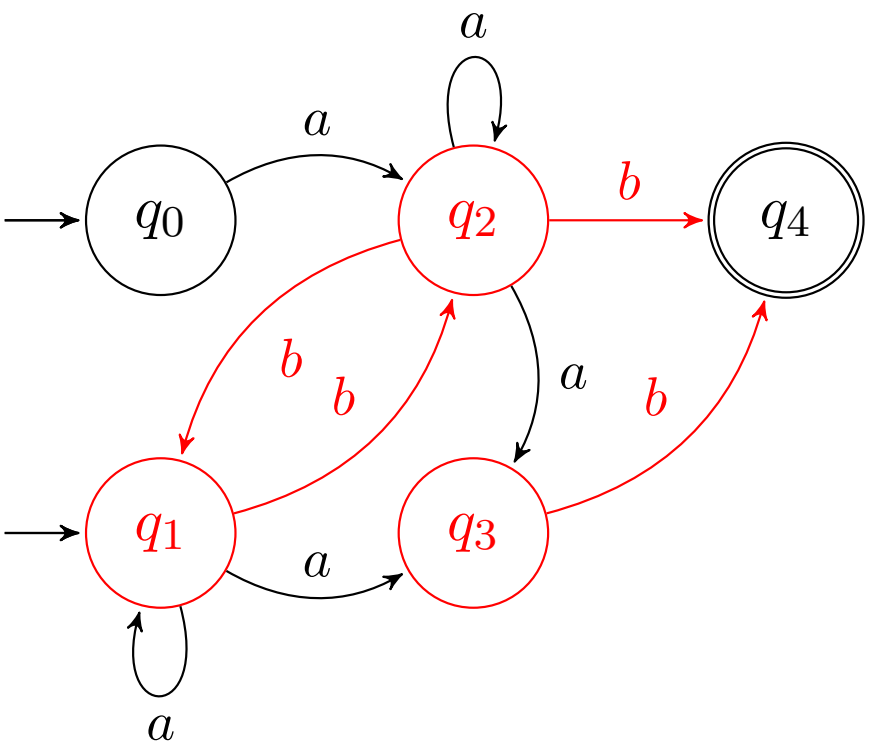


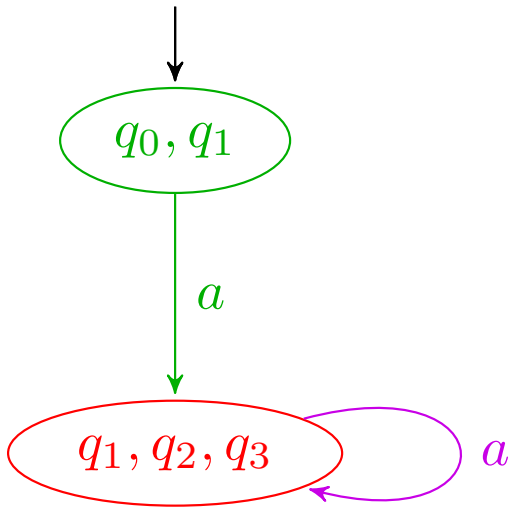


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

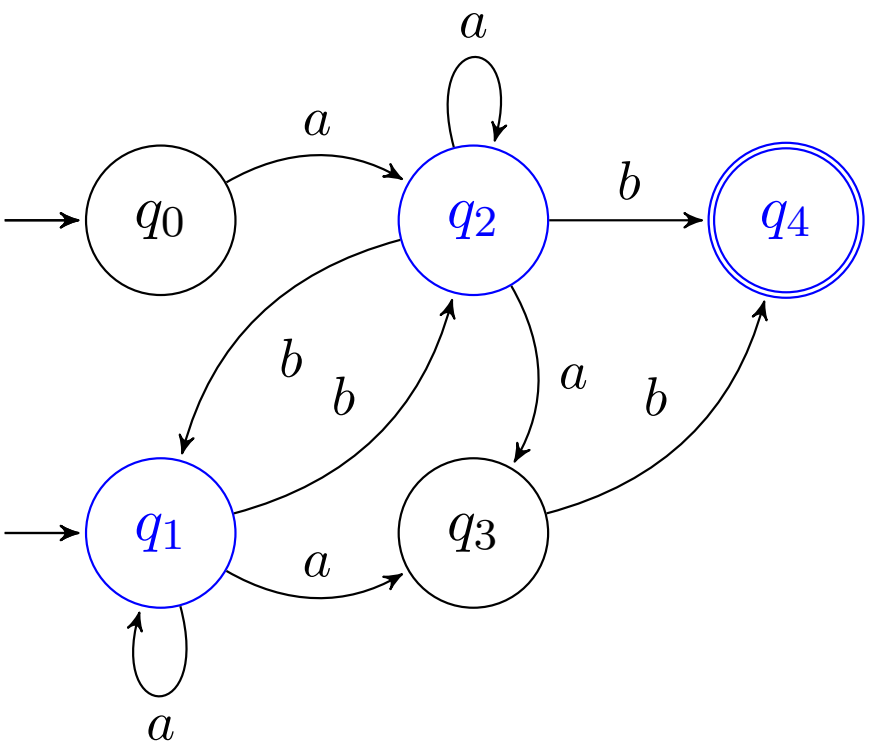


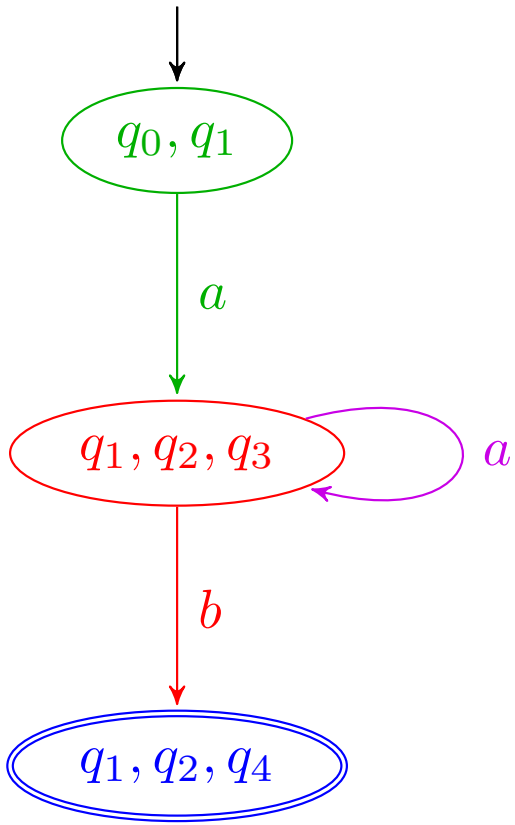


Principe de la déterminisation

Idée : suivre tous les chemins en parallèle

Exemple : L'automate suivant reconnaît-il aab ?

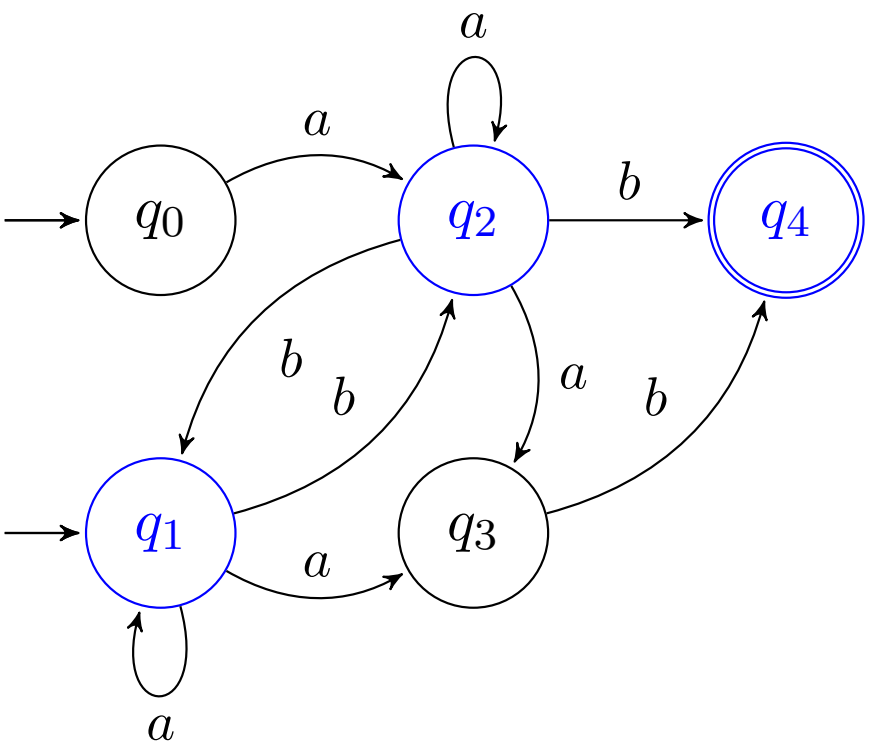


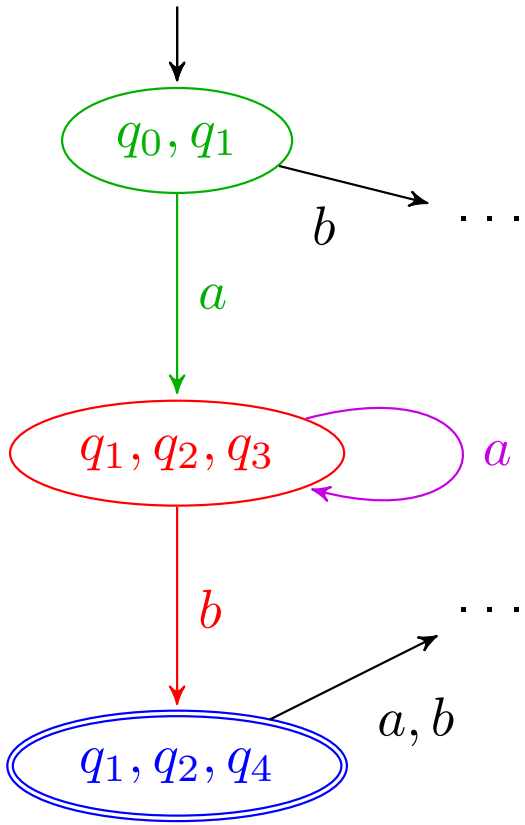


Principe de la déterminisation

Idée : suivre tous les chemins 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

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

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

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*.*

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

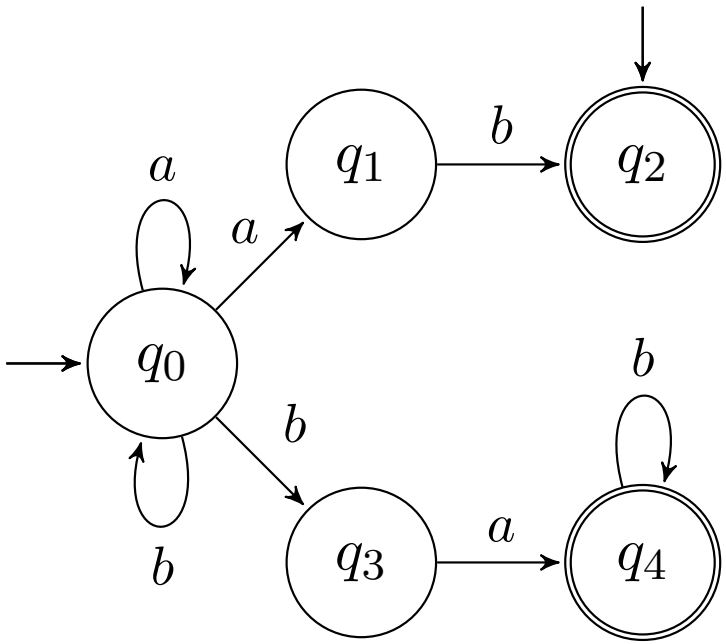
- $P \subseteq Q \iff P \in \mathcal{P}(Q)$ et $\emptyset \subseteq Q$: *un état puits de* B
- *Certains* $P \subseteq Q$ *peuvent ne pas être accessibles depuis* I *donc on construit* B *de proche en proche à partir de* I.

Proposition

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

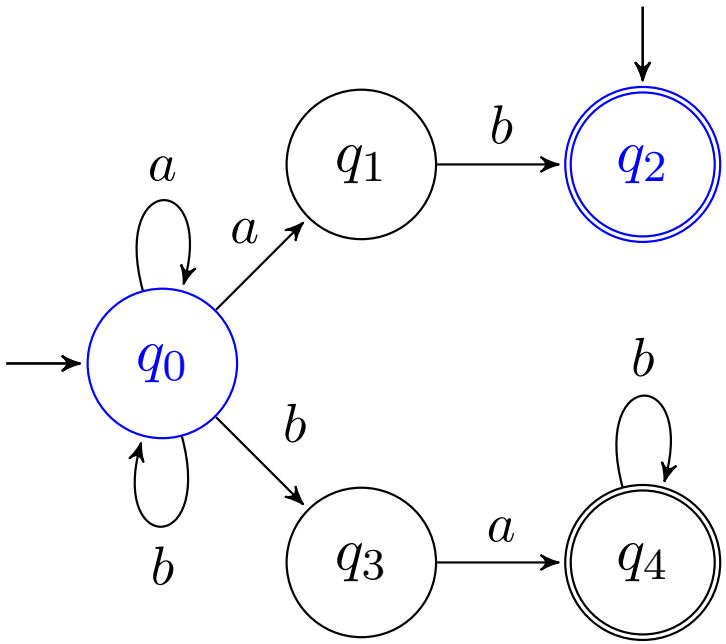
Exemple

Construire un AFD (complet) équivalent à :



Exemple

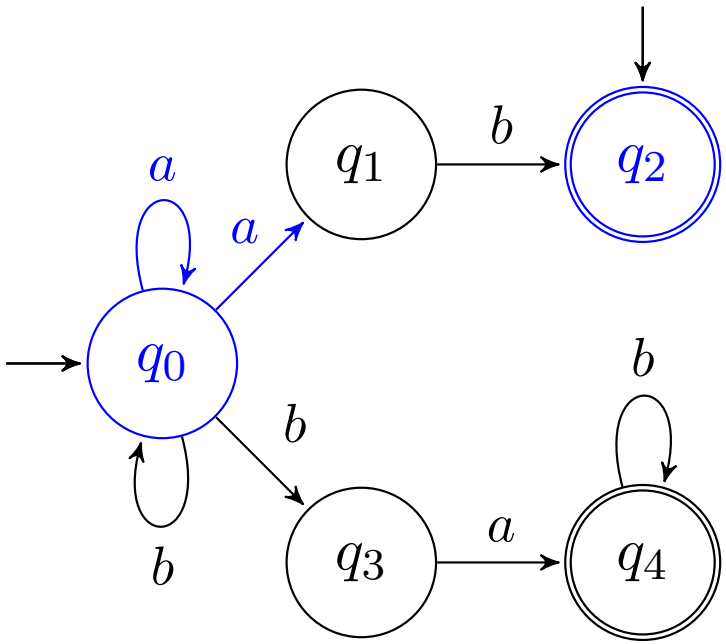
Construire un AFD (complet) équivalent à :



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

Exemple

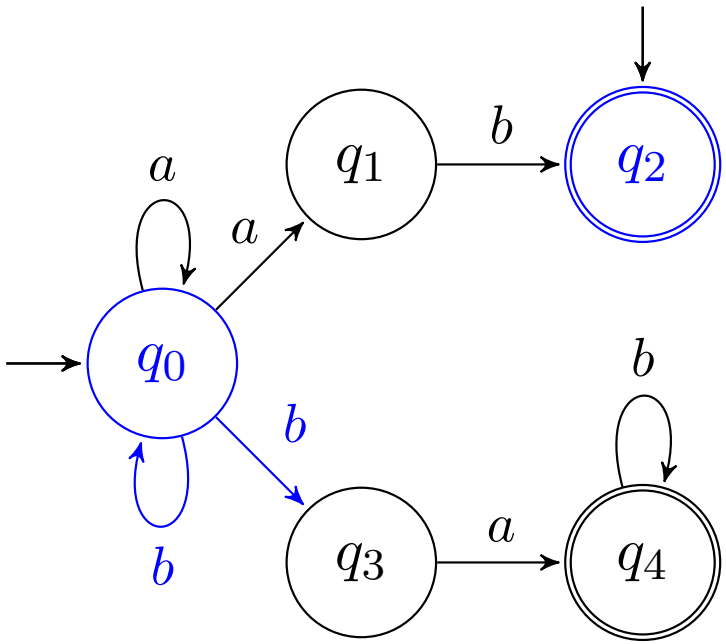
Construire un AFD (complet) équivalent à :



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

Exemple

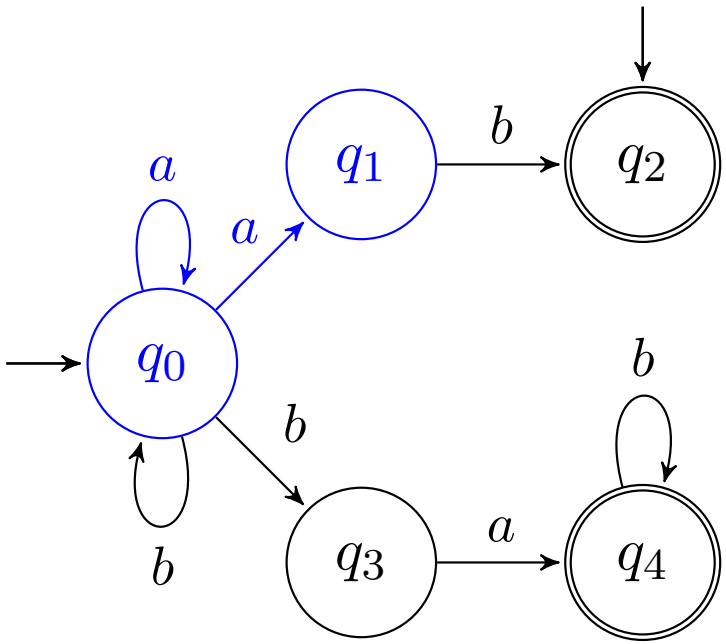
Construire un AFD (complet) équivalent à :



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

Exemple

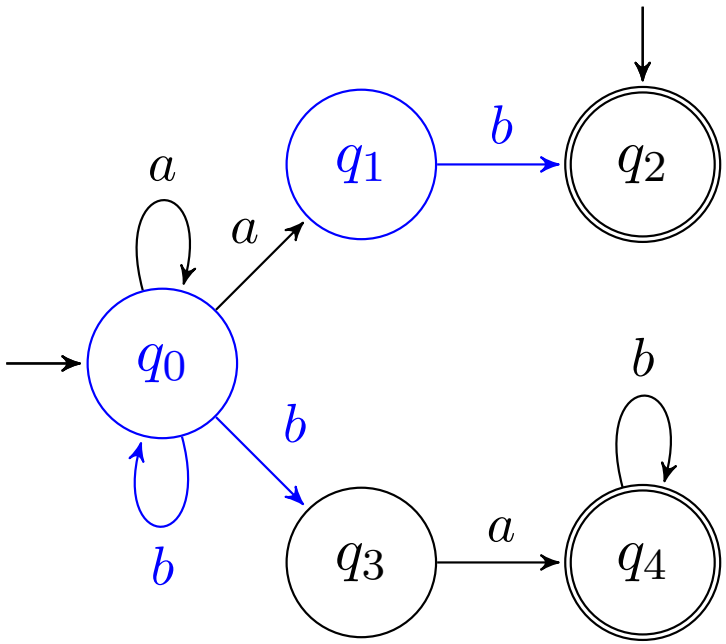
Construire un AFD (complet) équivalent à :



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

Exemple

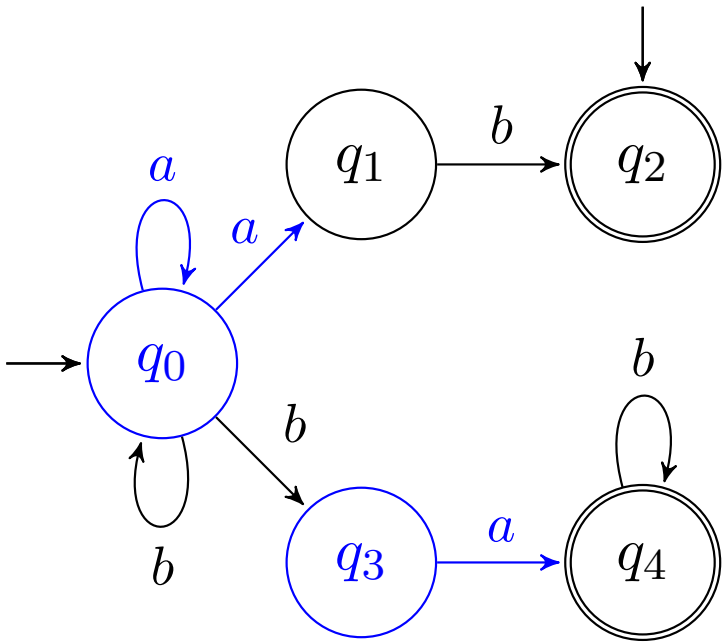
Construire un AFD (complet) équivalent à :



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

Exemple

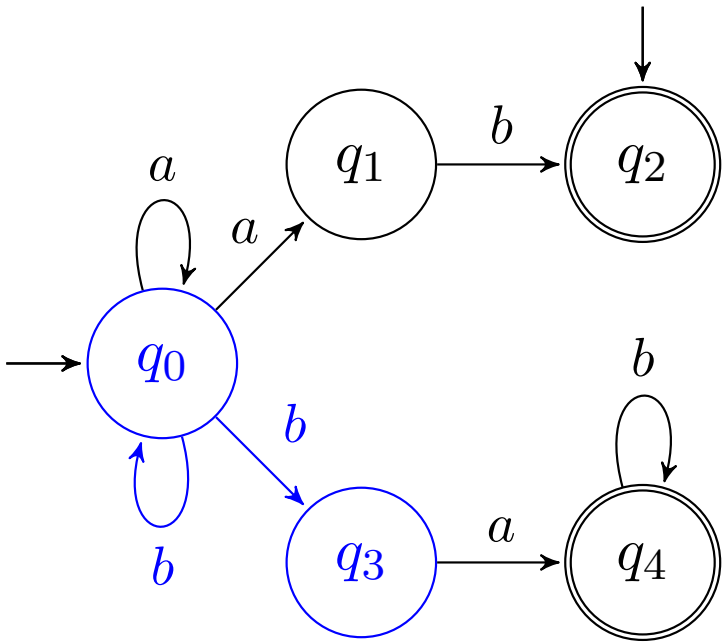
Construire un AFD (complet) équivalent à :



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

Exemple

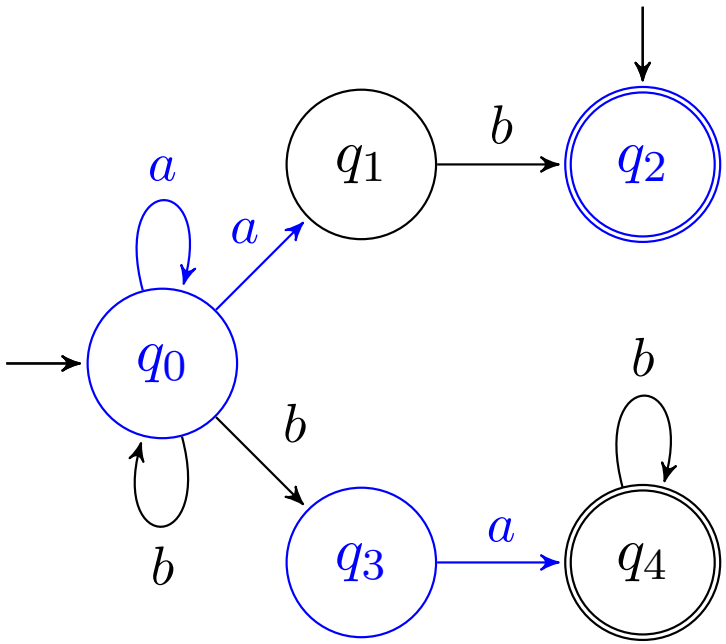
Construire un AFD (complet) équivalent à :



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

Exemple

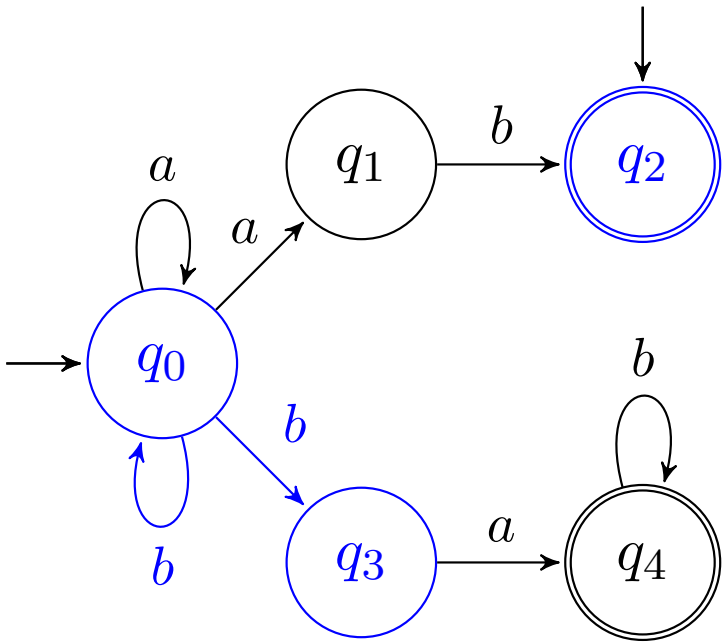
Construire un AFD (complet) équivalent à :



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

Exemple

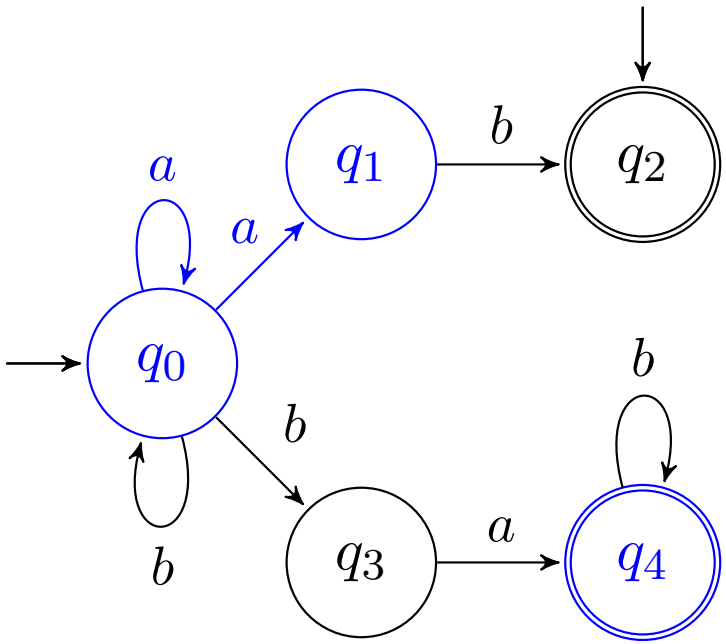
Construire un AFD (complet) équivalent à :



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

Exemple

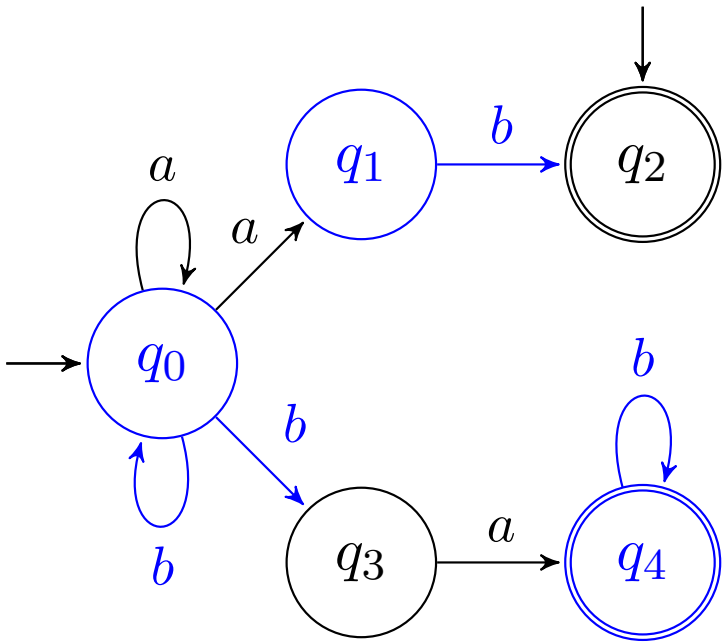
Construire un AFD (complet) équivalent à :



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

Exemple

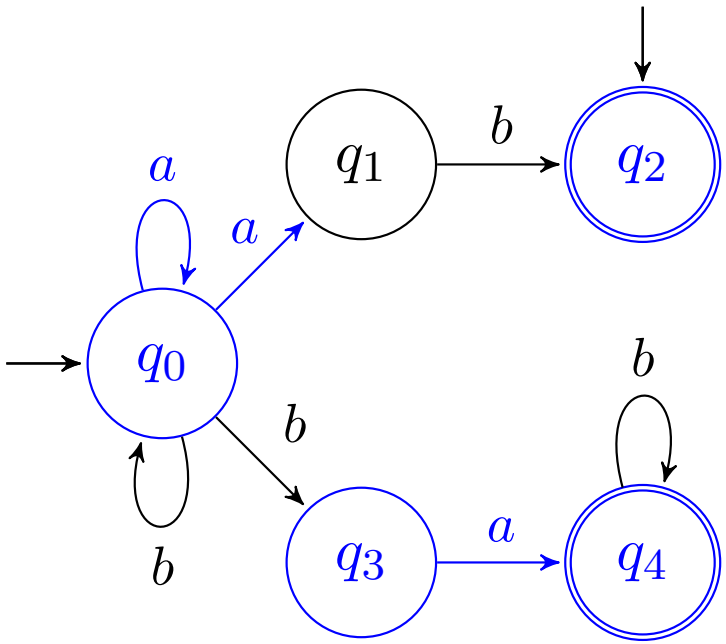
Construire un AFD (complet) équivalent à :



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

Exemple

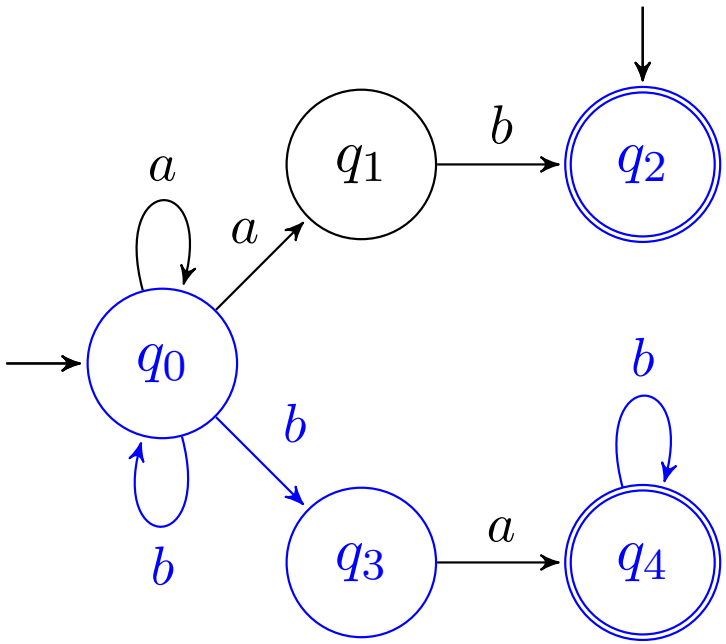
Construire un AFD (complet) équivalent à :



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

Exemple

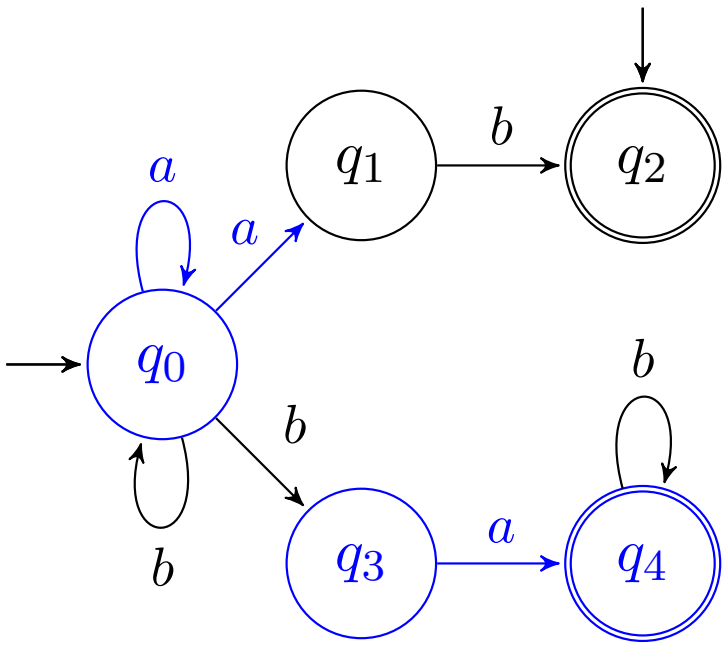
Construire un AFD (complet) équivalent à :



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

Exemple

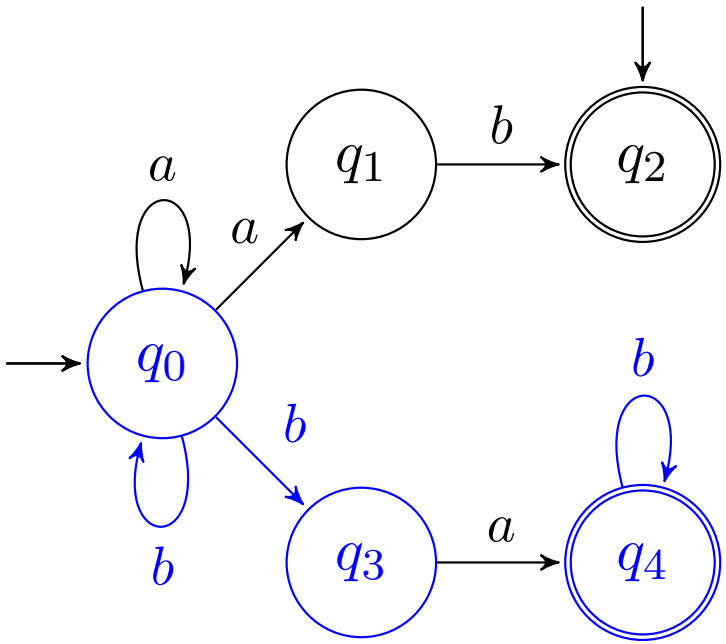
Construire un AFD (complet) équivalent à :



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

Exemple

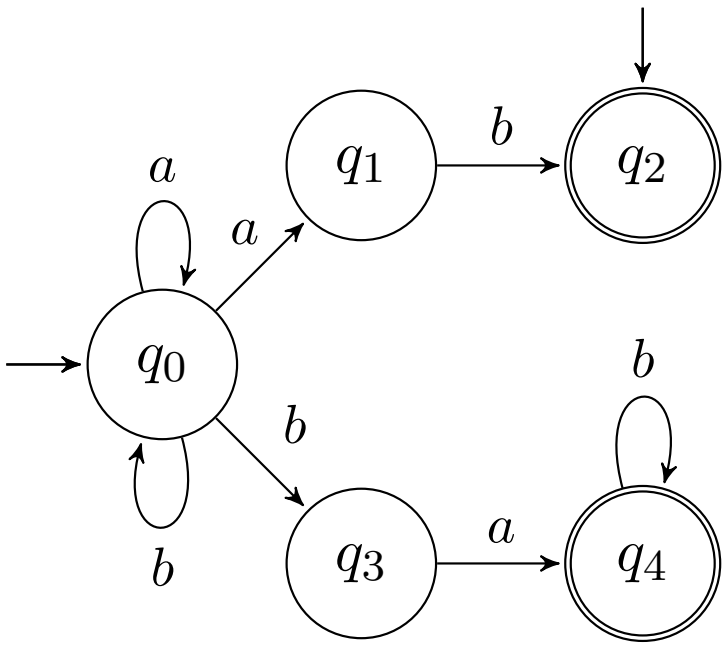
Construire un AFD (complet) équivalent à :



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

Exemple

Construire un AFD (complet) équivalent à :

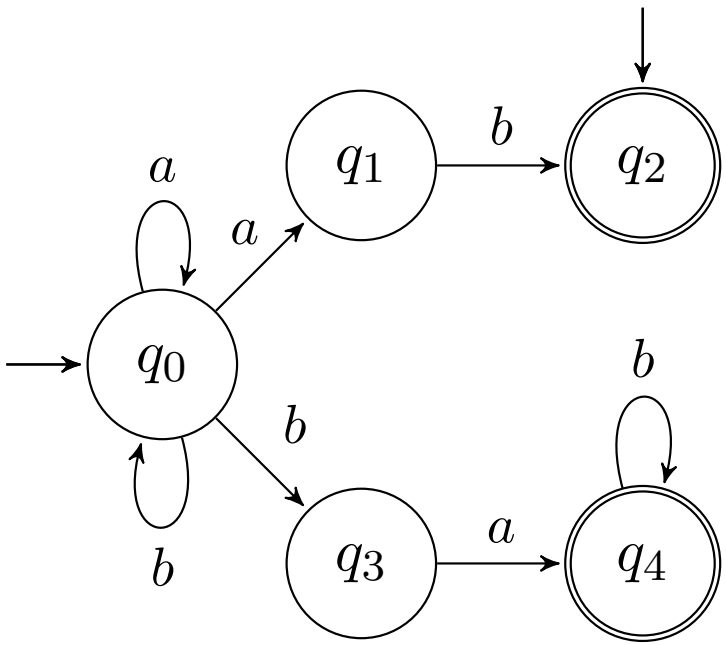


	$\mathcal{P}(Q)$	a	b
$\mathcal{I}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3\}$
	$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3, q_4\}$
	$\{q_0,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

Pas d'autre état accessible

Exemple

Construire un AFD (complet) équivalent à :



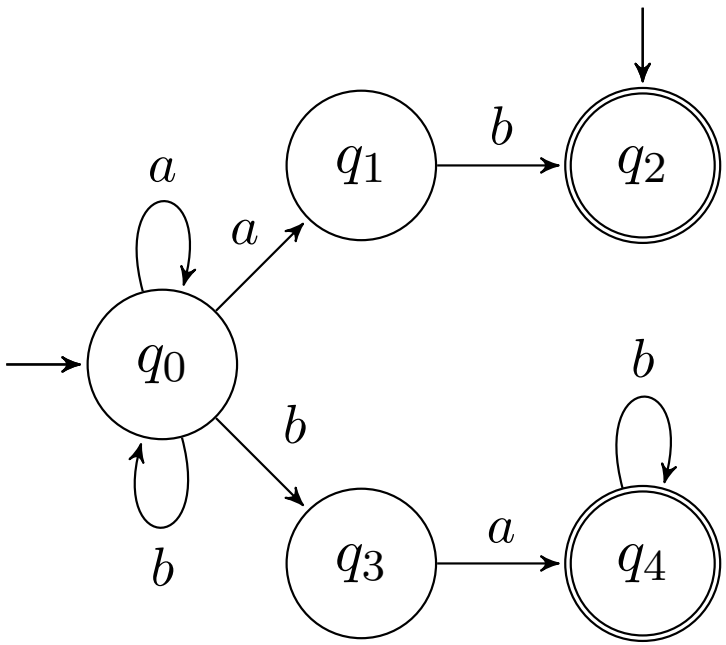
	$\mathcal{P}(Q)$	a	b
$\mathcal{I}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3\}$
	$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3, q_4\}$
	$\{q_0,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

Pas d'autre état accessible

7 états accessibles (sur 32 potentiels)

Exemple

Construire un AFD (complet) équivalent à :



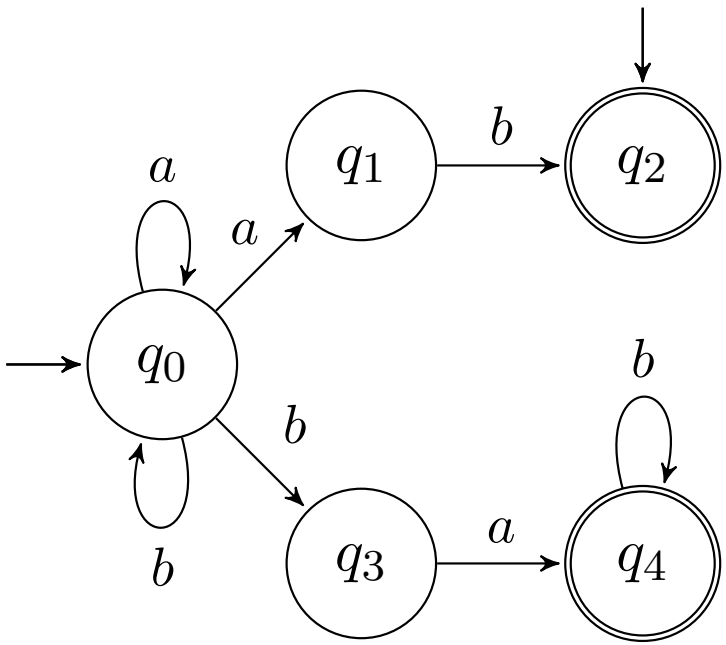
	$\mathcal{P}(Q)$	a	b
$\mathcal{I}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3\}$
	$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3, q_4\}$
	$\{q_0,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

Pas d'autre état accessible

7 états accessibles (sur 32 potentiels)

Exemple

Construire un AFD (complet) équivalent à :



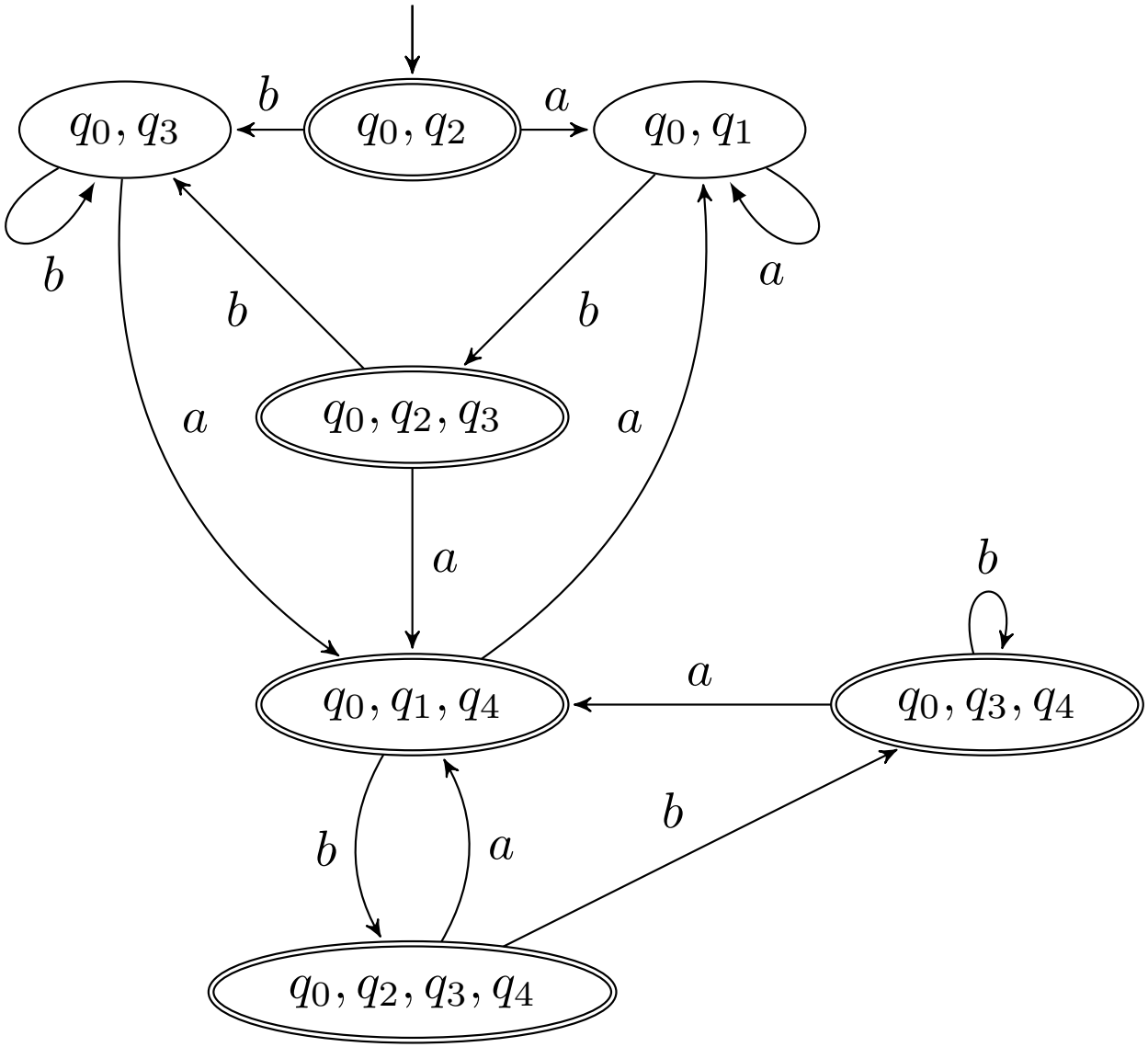
	$\mathcal{P}(Q)$	a	b
$\mathcal{I}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3\}$
	$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
	$\{q_0, q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3, q_4\}$
	$\{q_0,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

Pas d'autre état accessible

7 états accessibles (sur 32 potentiels)

5 états acceptants accessibles (sur 24 potentiels)

Solution



Résumé

Théorème

La classe des langages réguliers est fermée :

- *par union*
- *par concaténation*
- *par concaténation itérée*

Nous en verrons d'autres au cours 7.

Question

Peut-on toujours effectuer les transformations

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