Automates finis

Montaigne 2023-2024

– mpi23@arrtes.net –

Les automates finis, c'est l'infini mis à la portée des informaticiens.

Jacques Sakarovitch

Introduction

Le chapitre précédent a montré l'élégance et l'expressivité des expressions régulières.

Elles constituent un premier moyen pour tester l'appartenance d'un mot au langage dénoté par une expression.

Une autre approche fait appel à des objets calculatoires appelés **automates de mots finis**.

Introduction

Un **automate** est une **machine abstraite** qui peut prendre un nombre fini d'**états**.

L'état global de l'automate change selon des sollicitations extérieures.

Formellement, un automate admet une définition mathématique en termes ensemblistes.

Une représentation graphique en donne une description visuelle globale.

On distingue différentes formes d'automates finis : déterministes, non déterministes, complets, incomplets, émondés, etc.

Dans la suite de l'exposé, Σ désigne un alphabet fini.

Introduction

- Soit une machine à café qui accepte des pièces de 10, 20 et 50 centimes.
- Un café coûte 40 centimes.
- La machine délivre le café dès que la somme introduite est supérieure ou égale à 40 centimes.
- Elle ne rend pas la monnaie !
- La situation finale est celle pour laquelle 40 centimes au moins ont été introduits dans la machine.

Introduction

L'automate peut être décrit par un **tableau**.

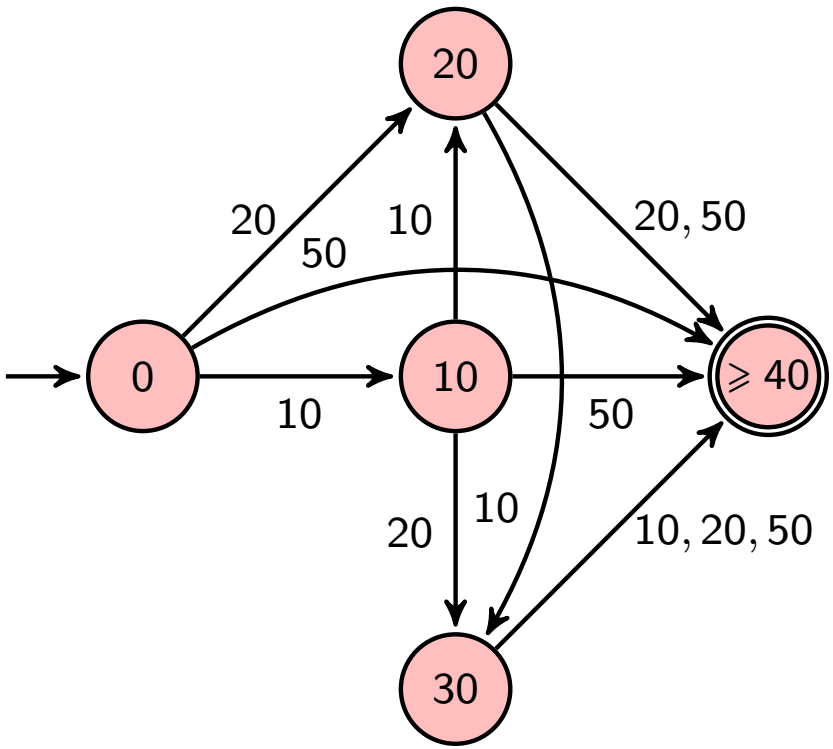
- Les lignes correspondent aux **états** (valeur numéraire possible).
- Les colonnes correspondent aux **sollicitations extérieures** (pièces).
- Aux intersections se trouvent les **états de transition**.

	10	20	50
0	10	20	$\geqslant 40$
10	20	30	$\geqslant 40$
20	30	$\geqslant 40$	$\geqslant 40$
30	$\geqslant 40$	$\geqslant 40$	$\geqslant 40$
$\geqslant 40$			

Introduction

L'automate peut aussi être représenté par un **graphe**.

- Les sommets du graphe sont appelés les **états** de l'automate. Dans l'exemple, c'est la valeur numéraire possible.
- Les arêtes du graphe sont appelées les **transitions étiquetées** par les données. Dans l'exemple, ce sont les valeurs des pièces.



Automates déterministes

DFA complet

Définition 1 (automate fini déterministe complet)

Un **automate fini déterministe complet** est un quintuplet $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ où :

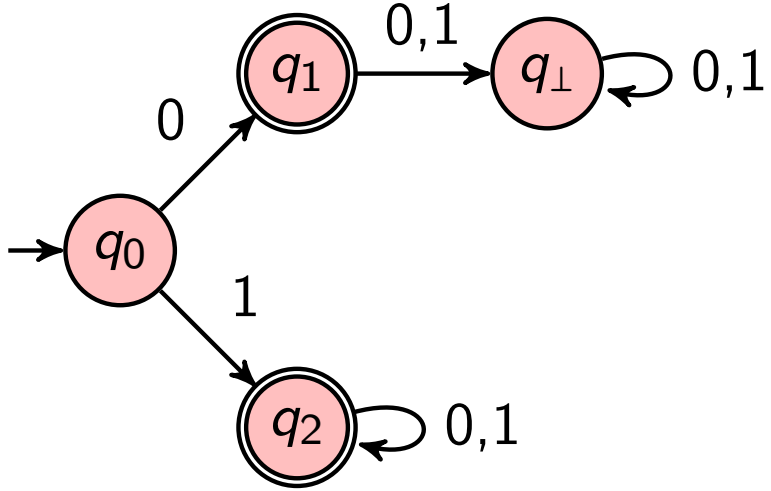
- Q est un ensemble fini d'**états** ;
- Σ est un **alphabet** ;
- $q_0 \in Q$ est l'**état initial** ;
- $F \subseteq Q$ est l'ensemble des **états acceptants** (ou finaux) ;
- $\delta : Q \times \Sigma \to Q$ est une fonction totale, appelée **fonction de transition** de l'automate.

En anglais, on le désigne par *DFA* pour **deterministic finite automaton**.

DFA complet

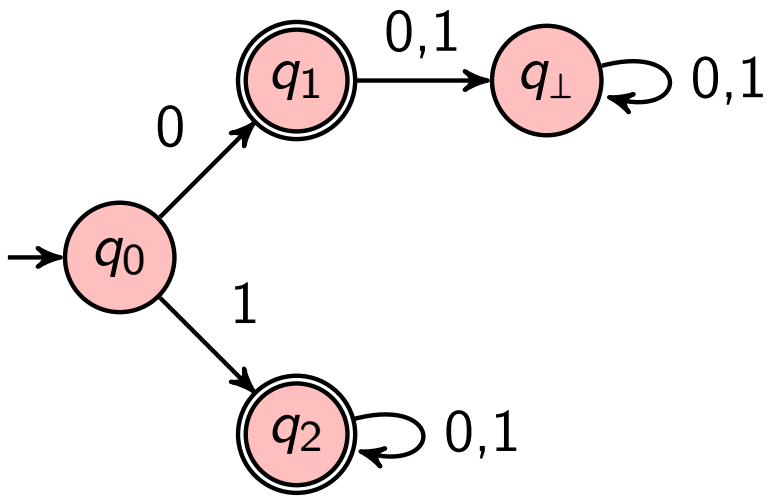
Soit l'automate $\mathcal{A}_{\text{bin}} = (\{q_0, q_1, q_2, q_\perp\}, \{0,1\}, q_0, \{q_1, q_2\}, \delta_{\text{bin}})$ où δ_{bin} est la fonction définie ci-dessous. On peut représenter visuellement cette fonction à l'aide du graphe ci-dessous.

$$\delta_{\text{bin}} : \begin{cases} (q_0, 0) \mapsto q_1 \\ (q_0, 1) \mapsto q_2 \\ (q_1, 0) \mapsto q_\perp \\ (q_1, 1) \mapsto q_\perp \\ (q_2, 0) \mapsto q_2 \\ (q_2, 1) \mapsto q_2 \\ (q_\perp, 0) \mapsto q_\perp \\ (q_\perp, 1) \mapsto q_\perp \end{cases}$$



Mot accepté/rejeté

Le fonctionnement d'un automate est intuitif : il lit un mot symbole par symbole, en partant de l'état initial et en se déplaçant suivant la fonction de transition. Si en fin de mot l'état courant est un état acceptant, le mot est **accepté**. Dans le cas contraire, le mot est **rejeté**.



Ce fonctionnement peut être formellement défini par la notion de chemin.

Chemin

Définition 2 (chemin dans un automate déterministe)

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un automate fini déterministe et $v = a_1 \dots a_n$ un mot de Σ^*. Un **chemin** de l'état r_0 à l'état r_n dans $\mathcal{A}$ pour le mot v est une séquence $r_0, \dots, r_n$ de $n+1$ états telle que :

- $\forall i, 0 \leqslant i \leqslant n, r_i \in Q$;
- $\forall i, 1 \leqslant i \leqslant n, r_i = \delta(r_{i-1}, a_i)$.

Le chemin est dit **acceptant** si $r_0 = q_0$ et $r_n \in F$. On dit que $\mathcal{A}$ **reconnaît** (ou **accepte**) le mot v s'il existe un chemin acceptant dans $\mathcal{A}$ pour le mot v.

Chemin

On note $r_0 \xrightarrow{a_1} r_1 \xrightarrow{a_2} \dots r_{n-1} \xrightarrow{a_n} r_n$ un chemin pour expliciter les symboles lus par l'automate.

On note $r_0 \xrightarrow{v}{}^*_{\mathcal{A}} r_n$ le **chemin de r_0 à r_n** pour le mot v dans l'automate $\mathcal{A}$ et on notera simplement $r_0 \xrightarrow{v}{}^* r_n$ lorsque l'automate considéré est le seul du contexte.

Dans le cadre d'une transition $p \longrightarrow q$, on dit que p est l'**état source** et q la **destination**. On dit aussi, par analogie avec les graphes orientés que p est un **prédécesseur** de q et q est un **successeur** de p.

Langage d'un automate

Un automate permet la reconnaissance de mots et, plus largement, d'un langage, à savoir un ensemble de mots.

Définition 3 (langage d'un automate)

Soit $\mathcal{A}$ un automate. Le langage de l'automate $\mathcal{A}$, noté $\mathcal{L}(\mathcal{A})$, est l'ensemble des mots acceptés par l'automate.

Définition 4 (langage reconnaissable)

Soit $L \subseteq \Sigma^*$ un langage. L est **reconnaissable** s'il existe un automate fini déterministe $\mathcal{A}$ tel que $L = \mathcal{L}(\mathcal{A})$.

L'ensemble des langages reconnaissables est noté $\mathrm{Rec}(\Sigma)$.

Déterministe et complet

Dans un automate déterministe et complet $\mathcal{A}$, pour tout mot $w \in \Sigma^*$, il n'existe **qu'un seul** chemin dans $\mathcal{A}$ pour w.

L'**existence** du chemin vient du fait que la **fonction** est **totale** : pour chaque état et chaque lettre de Σ, la fonction de transition est définie.

L'**unicité** vient du fait que le codomaine de la fonction est Q, l'ensemble des états. Pour chaque état source et pour chaque lettre, il y a donc un seul état de destination possible.

DFA incomplet

Si elle est théoriquement souhaitable, la complétude peut ne pas être satisfaisante d'un point de vue pratique car on représente souvent un ensemble de transitions **inutiles** pour lesquelles l'automate **ne fait rien**.

C'est le cas dans l'exemple présenté plus haut, où toutes les transitions impliquant $q_\perp$ ne sont là que pour ignorer des mots.

Au prix d'une légère modification, on peut définir la notion d'**automate déterministe incomplet**.

DFA incomplet

Définition 5 (automate déterministe incomplet)

Un **automate fini déterministe incomplet** est un quintuplet $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$, où

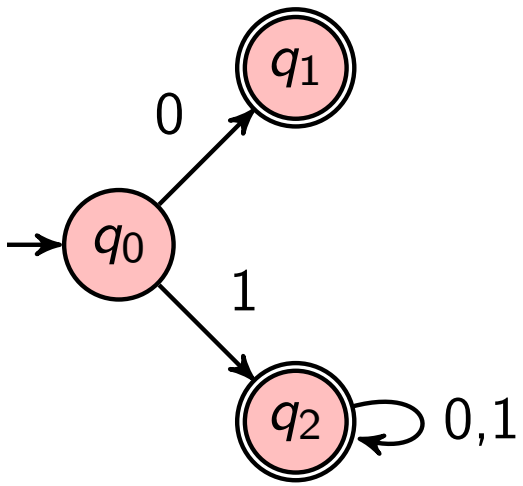
- Q est un ensemble d'états ;
- Σ est un alphabet ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des états acceptants (ou finaux) ;
- $\delta : Q \times \Sigma \to Q$ est une fonction **partielle**, appelée fonction de transition de l'automate.

La **fonction partielle** implique qu'il existe **au plus** un chemin dans $\mathcal{A}$ pour un mot donné. Les notions d'exécution et de langage d'un automate restent inchangées.

DFA incomplet

Soit l'automate : $\mathcal{A}'_{\text{bin}} = (\{q_0, q_1, q_2\}, \{0, 1\}, q_0, \{q_1, q_2\}, \delta'_{\text{bin}})$.

$$\delta'_{\text{bin}} : \begin{cases} (q_0, 0) \mapsto q_1 \\ (q_0, 1) \mapsto q_2 \\ (q_2, 0) \mapsto q_2 \\ (q_2, 1) \mapsto q_2 \end{cases}$$



Cet automate reconnaît le même langage que l'automate $\mathcal{A}_{\text{bin}}$ vu plus haut.

Complétion

Théorème 6 (complétion d'automate déterministe)

Soit $\mathcal{A}_i = (Q_i, \Sigma, q_0, F, \delta_i)$ un automate déterministe incomplet. Il existe un automate déterministe complet $\mathcal{A}_c$ tel que :

$$\mathcal{L}(\mathcal{A}_i) = \mathcal{L}(\mathcal{A}_c).$$

Complétion

Démonstration

Considérons l'automate $\mathcal{A}_c = (Q_c, \Sigma, q_0, F, \delta_c)$ où $Q_c = Q_i \cup \{q_\perp\}$, avec $q_\perp \notin Q_i$ et :

$$\delta_c : \begin{cases} Q_c \times \Sigma \to Q_c & \\ (q,a) \mapsto \delta_i(q,a) & \text{si } (q,a) \in \mathrm{dom}(\delta_i) \\ (q,a) \mapsto q_\perp & \text{si } (q,a) \notin \mathrm{dom}(\delta_i) \\ (q_\perp, a) \mapsto q_\perp & \forall a \in \Sigma \end{cases}$$

Montrons que pour tout mot $v \in \Sigma^*$:

$$v \in \mathcal{L}(\mathcal{A}_i) \to v \in \mathcal{L}(\mathcal{A}_c)$$

et :

$$v \notin \mathcal{L}(\mathcal{A}_i) \to v \notin \mathcal{L}(\mathcal{A}_c)$$

Complétion

Démonstration (suite)

Montrons d'abord $v \in \mathcal{L}(\mathcal{A}_\mathrm{i}) \to v \in \mathcal{L}(\mathcal{A}_\mathrm{c})$.

Si $v = a_1 \dots a_n$ et $v \in \mathcal{L}(\mathcal{A}_\mathrm{i})$, alors il existe un chemin acceptant :

$$r_0 \overset{v}{\longrightarrow}^*_{\mathcal{A}_\mathrm{i}} r_n$$

Ce chemin est aussi un chemin acceptant pour $\mathcal{A}_\mathrm{c}$, car :

- $r_0 = q_0$ est aussi l'état initial de $\mathcal{A}_\mathrm{c}$;
- $\forall 0 \leqslant i < n, \delta_\mathrm{c}(r_i, a_{i+1}) = \delta_\mathrm{i}(r_i, a_{i+1})$ puisque les deux fonctions coïncident sur le domaine de δ_i ;
- $r_n \in F$ et F est aussi l'ensemble des états acceptants de $\mathcal{A}_\mathrm{c}$.

Complétion

Démonstration (fin)

Montrons ensuite $v \notin \mathcal{L}(\mathcal{A}_\mathsf{i}) \to v \notin \mathcal{L}(\mathcal{A}_\mathsf{c})$.

Supposons $v \notin \mathcal{L}(\mathcal{A}_\mathsf{i})$. Par cas :

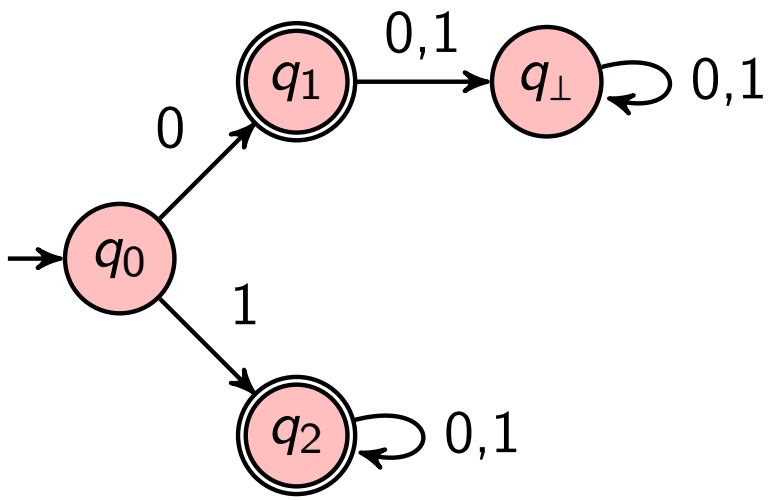
- soit il existe un chemin non-acceptant $q_0 \xrightarrow{v}{}^*_{\mathcal{A}_\mathsf{i}} q'$ arrivant dans un certain q'. Comme précédemment, ce chemin est aussi un chemin pour $\mathcal{A}_\mathsf{c}$ et il est aussi non acceptant.
- soit il n'existe pas de chemin dans $\mathcal{A}_\mathsf{i}$ pour $v = a_1 \dots a_n$. Dans ce cas, il existe un préfixe $u = a_1 \dots a_k$ de v (potentiellement vide) et un chemin $r_0, \dots, r_k$ de $\mathcal{A}_\mathsf{i}$ pour u, avec $(r_k, a_{k+1}) \notin \mathrm{dom}(\delta_\mathsf{i})$.
  Le chemin $r_0, \dots, r_k, \underbrace{q_\perp, \dots, q_\perp}_{(k-n) \text{ fois}}$ est un chemin pour v. En effet,

  $(r_k, a_{k+1}) \notin \mathrm{dom}(\delta_\mathsf{i}) \to \delta_\mathsf{c}(r_k, a_{k+1}) = q_\perp$. Et $\forall k < j \leqslant n, \delta_\mathsf{c}(q_\perp, a_j) = q_\perp$.

État puits

L'état $q_\perp$ ajouté dans la preuve du théorème est communément appelé un **état puits** car une fois que l'automate y arrive, il ne peut en sortir.

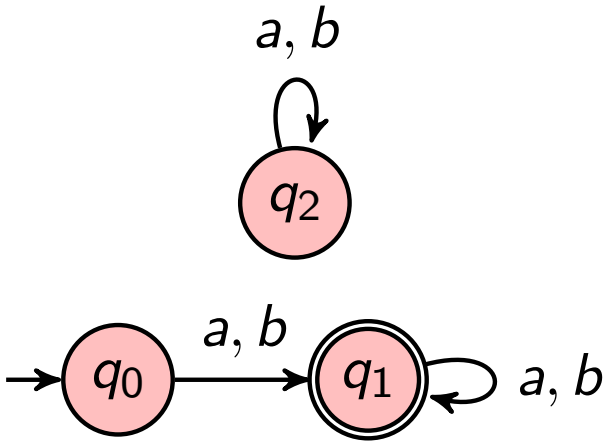
L'automate $\mathcal{A}_{bin}$ défini plus haut comporte un tel état puits.



Émondage ?

Le théorème a montré qu'il est toujours possible de passer d'un automate incomplet à un automate complet. L'opération inverse, appelée **émondage**, est-elle possible, c'est-à-dire retirer les états et transitions **inutiles** d'un automate ?

Par exemple, l'automate suivant est complet et accepte les mots d'au moins une lettre sur $\Sigma = \{a, b\}$. Mais l'état q_2 n'étant pas relié au reste de l'automate, il ne peut êtte atteint depuis q_0 et ne peut donc pas faire partie d'une exécution acceptante.



État accessible

Définition 7

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un automate déterministe. Un état $q \in Q$ est **accessible** s'il existe un mot $v \in \Sigma^*$ et un chemin $q_0 \xrightarrow{v}{}^*_{\mathcal{A}} q$.

On détermine l'ensemble des états accessibles d'un automate en effectuant un parcours (en largeur ou en profondeur) de son graphe associé depuis l'état initial q_0. L'ensemble des états visités lors du parcours est l'ensemble des états accessibles.

État co-accessible

Définition 8

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un automate déterministe. Un état $q \in Q$ est dit **co-accessible** s'il existe un mot $v \in \Sigma^*$ et un chemin $q \xrightarrow{v}{}^*_{\mathcal{A}} q'$ avec $q' \in F$.

On trouve tous les états co-accessibles d'un automate en effectuant un parcours de son graphe pour chacun des états acceptants et en considérant comme voisin d'un état q ses prédécesseurs, **i.e.** tout état p tel que $\delta(p, a) = q$ pour un certain $a \in \Sigma$.

État utile

Définition 9

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un automate déterministe. Un état $q \in Q$ est dit **utile** s'il est à la fois accessible et co-accessible.

Tous les états d'un chemin acceptant sont utiles.

Automate émondé

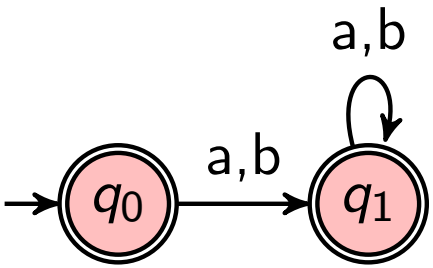
Définition 10 (automate émondé)

Un automate est dit **émondé** si tous ses états sont utiles.

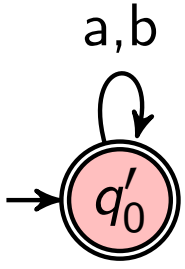
Partant d'un automate déterministe, on peut toujours construire un automate émondé sans changer le langage qu'il reconnaît car, par définition, un état qui n'est soit pas accessible soit pas co-accessible ne peut pas faire partie d'un chemin acceptant. Il faut cependant prendre garde au fait qu'un automate émondé n'est pas nécessairement le plus petit possible, au sens du nombre d'états.

Minimalité

Les deux automates $\mathcal{A}$ et $\mathcal{A}'$ suivants reconnaissent le même langage Σ^* pour $\Sigma = \{a, b\}$.



$\mathcal{A}$



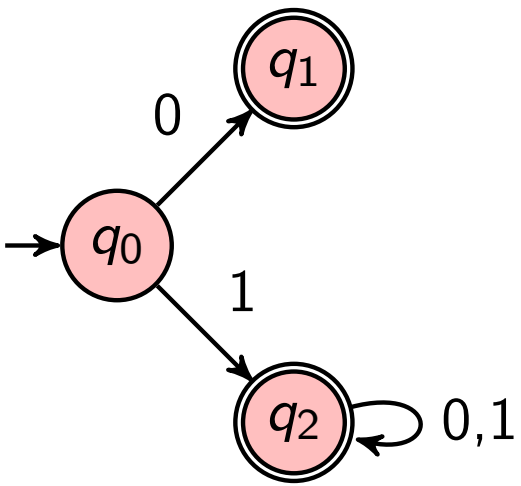
$\mathcal{A}'$

On peut aussi vérifier que, selon nos définitions, ils sont **complets** et **émondés**. Toutefois, l'automate $\mathcal{A}'$ possède cependant un état de moins.

La notion formelle de minimalité d'automate est hors programme.

Automate et programme

La présentation des automates faite jusqu'à présent est résolument calculatoire. Un automate est un programme effectuant une tâche bien particulière. Par exemple, l'automate émondé de l'automate suivant peut être simulé par le programme de la diapositive suivante.



```c
/* Déclaration des fonctions mutuellement récursives */
bool q0(void);
bool q1(void);
bool q2(void);

bool q0(void) {
  int c = getchar();
  if (c == '\n' || c == EOF) return false;
  if (c == '0') return q1();
  if (c == '1') return q2();
  // L'entrée ne doit être constituée que de 0 et de 1
  abort();
}

bool q1(void) {
  int c = getchar();
  if (c == '\n' || c == EOF) return true;
  // On n'est pas en fin de mot en q1, le mot est refusé.
  if (c == '0' || c == '1') return false;
  abort();
}

bool q2(void) {
  int c = getchar();
  if (c == '\n' || c == EOF) return true;
  if (c == '0' || c == '1') return q2();
  abort();
}
```

Automates non déterministes

NFA

Dans un automate **déterministe**, pour chaque état et chaque symbole lu, il y a au plus un état de destination. En relâchant cette contrainte, on définit la notion d'automate **non déterministe**.

En anglais, on le désigne par *NFA* pour **non-deterministic finite automaton**.

La définition suivante est celle d'un automate non possiblement incomplet. Le théorème sur la complétion vu plus haut reste valable pour les automates non déterministes.

NFA

Définition 11 (automate fini non déterministe)

Un **automate fini non déterministe** est un quintuplet $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$, où :

- Q est un ensemble d'états ;
- Σ est un alphabet ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des états acceptants (ou finaux) ;
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ est une fonction **partielle**, appelée fonction de transition de l'automate.

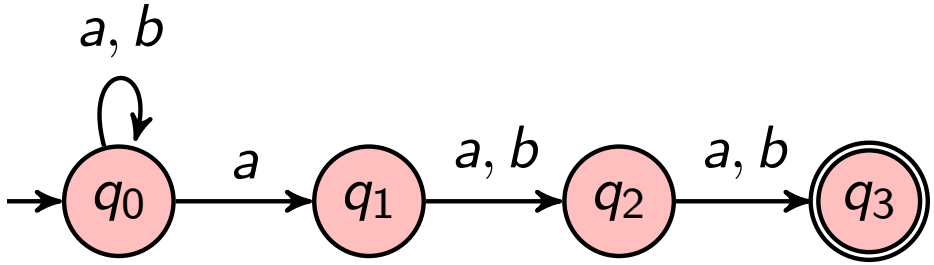
NFA

Illustrons notre propos avec l'automate :

$$\mathcal{A}_{a3} = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, q_0, \{q_3\}, \delta_{a3})$$

de graphe et de fonction de transition donnés ci-dessous.

$$\delta_{a3} : \begin{cases} (q_0, a) \mapsto \{q_0, q_1\} \\ (q_0, b) \mapsto \{q_0\} \\ (q_1, a) \mapsto \{q_2\} \\ (q_1, b) \mapsto \{q_2\} \\ (q_2, a) \mapsto \{q_3\} \\ (q_2, b) \mapsto \{q_3\} \end{cases}$$



Non déterminisme

Le non déterminisme de l'automate est observable sur l'état q_0 pour une transition liée à la lecture du symbole a : l'automate effectue un **choix non déterministe**.

Par exemple, pour lire le mot *abaaa*, l'automate peut faire le choix de transitions suivant : $q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$.

D'une certaine manière, depuis l'état q_0, sur lecture d'un a, l'automate doit **deviner** si ce a est le troisième avant la fin ou non, sans connaître le reste du mot !

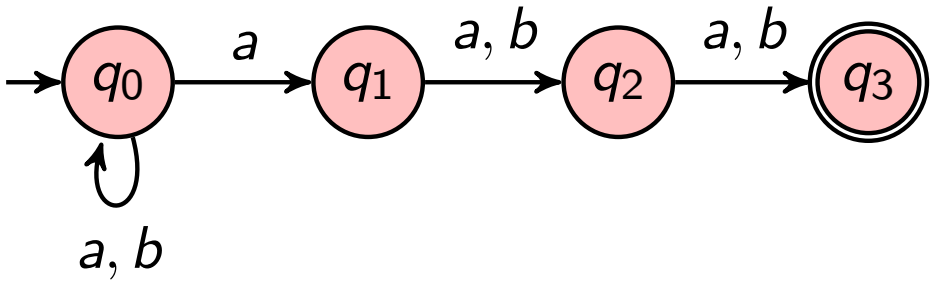
Non déterminisme

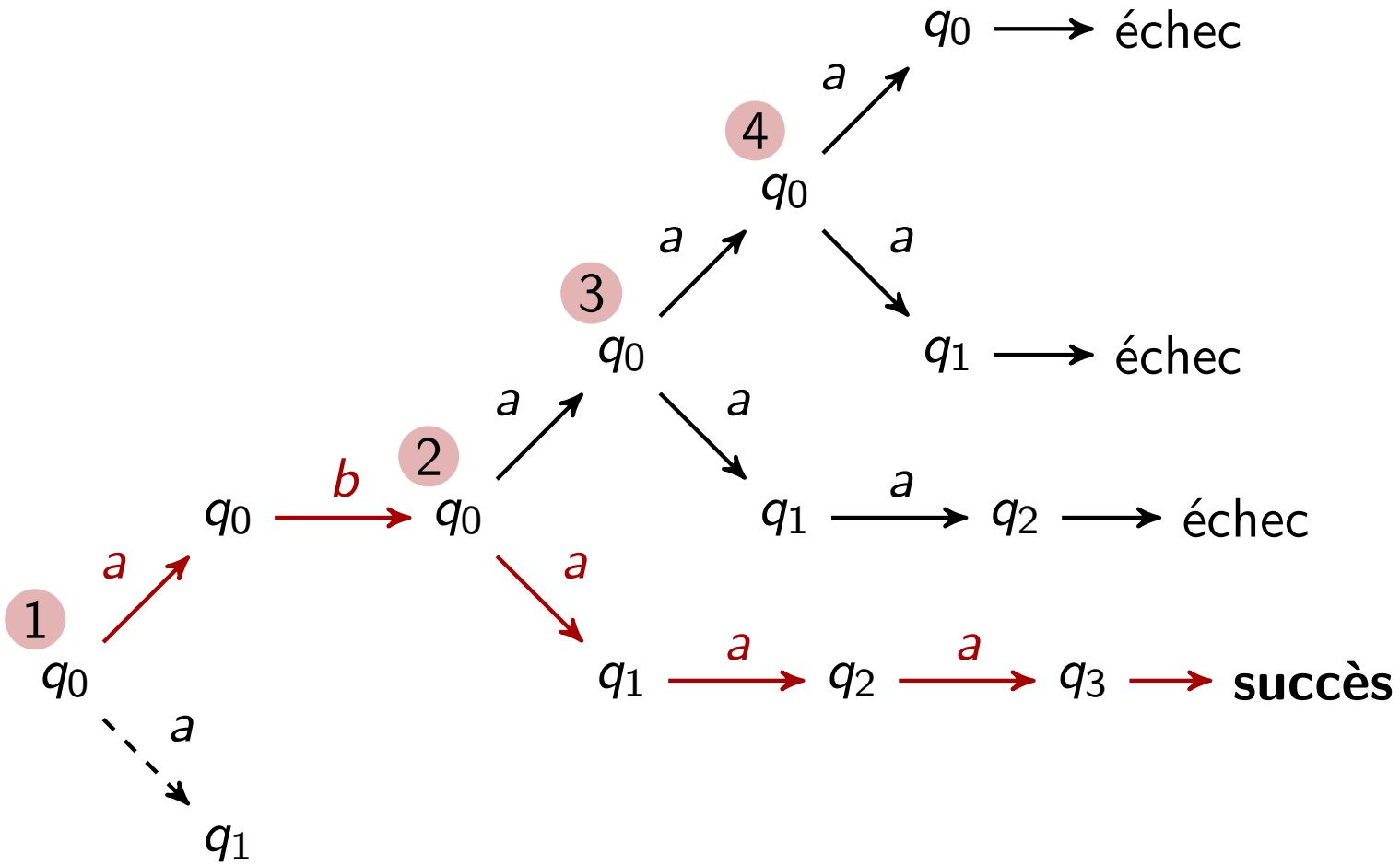
En pratique, un automate peut être associé à un programme.

Dès lors, comment construire programme qui rende compte du non déterminisme ? Ou dit dans les termes de la précédente diapositive, comment un programme rend-il compte du fait que l'automate « devine » le bon choix de transitions pour arriver dans un état acceptant ?

Une interprétation de l'implémentation est de considérer un automate non déterministe comme un programme utilisant une **stratégie de retour sur trace**. Dit autrement, l'automate recherche un chemin acceptant en essayant tous les choix possibles de façon exhaustive.

Non déterminisme





Chemin dans un automate non déterministe

Définition 12

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un NFA et $v = a_1 \dots a_n$ un mot de Σ^*. Un **chemin** dans l'automate $\mathcal{A}$ pour le mot v est une séquence $r_0, \dots, r_n$ de $n+1$ états telle que :

- $\forall i, 0 \leqslant i \leqslant n, r_i \in Q$;
- $\forall i, 1 \leqslant i \leqslant n, r_i \in \delta(r_{i-1}, a_i)$.

Le chemin est dit **acceptant** si $r_0 = q_0$ et $r_n \in F$.

La même notation $q \xrightarrow{v}{}^*_{\mathcal{A}} q'$ que pour les automates déterministes désigne un chemin de q à q' pour le mot v dans $\mathcal{A}$.

De même, un automate non deterministe $\mathcal{A}$ **accepte** ou **reconnaît** le mot v s'il existe un chemin acceptant pour v. On note $\mathcal{L}(\mathcal{A})$ le langage qu'il reconnaît.

ε-NFA

Les NFA peuvent être rendus encore moins déterministes en ajoutant un nouveau type de transition appelée **transition spontanée** ou **ε-transition**. On les appelle des *ε-NFA*.

Définition 13 (automate à transitions spontanées)

Un **automate fini non déterministe à transitions spontanées** est un quintuplet $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ où :

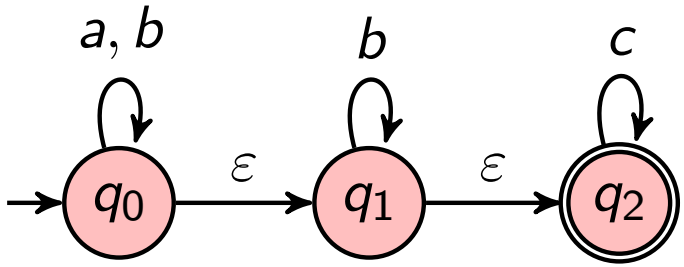
- Q est un ensemble d'états ;
- Σ est un alphabet ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des états acceptants (ou finaux) ;
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ est une fonction **partielle**, appelée fonction de transition de l'automate.

Les transitions spontanées peuvent être prises par l'automate « sans consommer » de symbole.

ε-NFA

$$\mathcal{A}_{\mathsf{abbc}} = (\{q_0, q_1, q_2\}, \{a, b, c\}, q_0, \{q_2\}, \delta_{\mathsf{abbc}})$$

$$\delta_{\mathsf{abbc}} : \begin{cases} (q_0, a) \mapsto \{q_0\} \\ (q_0, b) \mapsto \{q_0\} \\ (q_0, \varepsilon) \mapsto \{q_1\} \\ (q_1, b) \mapsto \{q_1\} \\ (q_1, \varepsilon) \mapsto \{q_2\} \\ (q_2, c) \mapsto \{q_2\} \end{cases}$$



ε-NFA

Parmi d'autres, cet automate reconnaît les mots suivants.

- ε : depuis l'état q_0, l'automate peut aller en q_1 sans lire de symbole, puis en q_2 toujours sans lire de symbole.
- ac : l'automate peut lire le symbole a en restant en q_0, puis passer en q_1 puis q_2 sans lire de symbole, puis boucler sur q_2 pour lire le c final.
- $bbbb$: l'automate peut se déplacer en q_1 sans lire de symbole, puis consommer les quatre b en restant en q_1 puis se déplacer en q_2 une fois le mot fini.

ε-NFA

Définition 14 (chemin acceptant dans un ε-NFA)

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un ε-NFA et v un mot de Σ^*. Un **chemin** dans $\mathcal{A}$ pour v est une séquence $r_0, \ldots, r_n$ de $n+1$ états telle que :

- il existe une séquence de mots $a_1 \ldots a_n$, avec $a_i \in \Sigma \cup \{\varepsilon\}$ pour $1 \leqslant i \leqslant n$, telle que $v = a_1 \ldots a_n$;
- $\forall i, 0 \leqslant i \leqslant n, r_i \in Q$;
- $\forall i, 1 \leqslant i \leqslant n, r_i \in \delta(r_{i-1}, a_i)$.

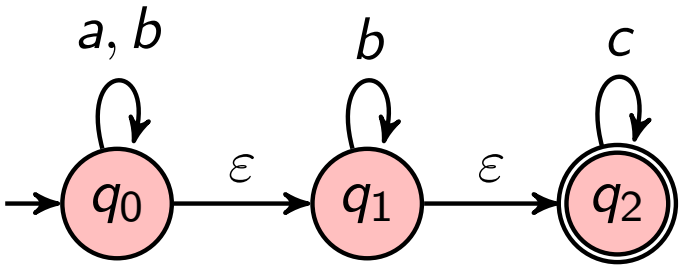
Le chemin est dit **acceptant** si $r_0 = q_0$ et $r_n \in F$.

Dans cette définition, $|v| \leqslant n$ puisque certains a_i peuvent être ε.

ε-NFA

Dans l'automate ci-dessous, le mot c est reconnu par le chemin :

$$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{c} q_2$$



Ainsi, en plus du non déterminisme qui consiste à choisir un état parmi ceux renvoyés par la fonction de transition, l'automate choisit de façon non déterministe comment découper le mot v comme la suite de ses lettres entrecoupées d'occurrences du mot vide.

Déterminisation

Trois modèles d'automates ont été décrits :

- les automates déterministes
- les automates non déterministes
- les automates à transitions spontanées.

Ces trois **modèles** sont **équivalents**.

- On peut **déterminiser** un automate non déterministe, c'est-à-dire le transformer en un automate déterministe qui reconnaît le même langage.
- On peut supprimer les transitions spontanées d'un automate non déterminiation durant la déterminisation.

Déterminisation

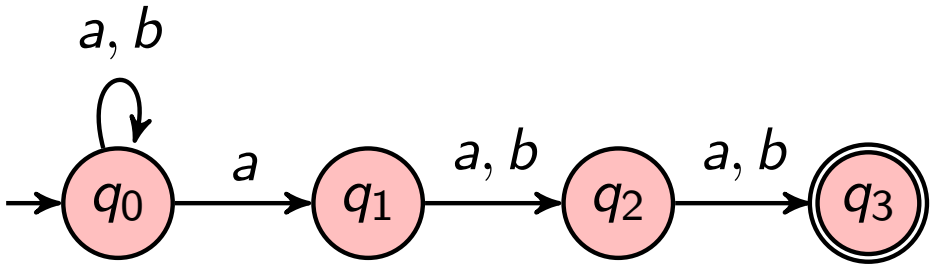
Définition 15 (construction par sous-ensembles)

Soit $\mathcal{A}_N = (Q_N, \Sigma, q_0, F_N, \delta_N)$ un NFA. Son déterminisé $\det(\mathcal{A}_N)$ est le DFA $\mathcal{A}_D = (Q_D, \Sigma, \{q_0\}, F_D, \delta_D)$ construit comme suit :

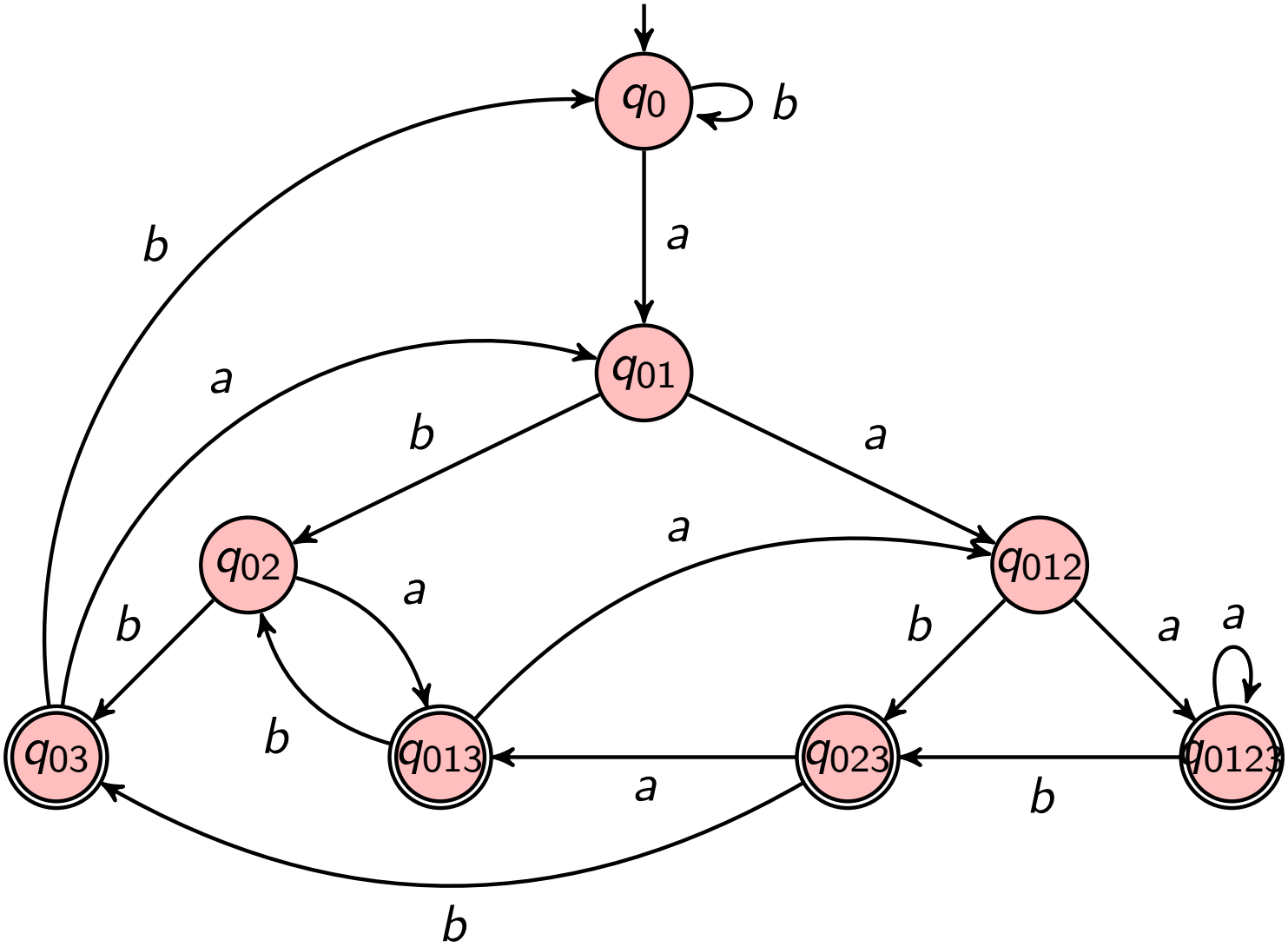
- $Q_D = \mathcal{P}(Q_N)$
- $F_D = \{S \mid S \in Q_D, S \cap F_N \neq \varnothing\}$
- $\delta_D : \begin{cases} Q_D \times \Sigma \to \mathcal{P}(Q_D) \\ (S, a) \mapsto \bigcup_{q \in S} \delta_N(q, a) \end{cases}$

Déterminisation

Déterminer l'automate suivant en appliquant la construction par sous-ensembles. Répondre sur la diapositive suivante.



Déterminisation



Déterminisation

Théorème 16 (équivalence déterministe / non déterministe)

Soit $L \subseteq \Sigma^*$ un langage. L est reconnaissable par un DFA si et seulement s'il est reconnaissable par un NFA.

Déterminisation

Démonstration

Supposons que L soit reconnu par un automate déterministe $\mathcal{A}_D = (Q_D, \Sigma, q_0, F_D, \delta_D)$.
On peut construire un automate non déterministe $\mathcal{A}_N = (Q_N, \Sigma, q_0', F_N, \delta_N)$ avec :

- $Q_N = Q_D$
- $q_0' = q_0$
- $F_N = F_D$
- $\delta_N : Q_N \times \Sigma \to \mathcal{P}(Q_N)$
  $(q, a) \mapsto \{\delta_D(q, a)\}$

En d'autres termes, on interprète l'automate déterministe comme un cas particulier d'automate non déterministe dont la fonction de transition renvoie un singleton. Il est évident que les deux automates reconnaissent le même langage.

Déterminisation

Démonstration

Dans l'autre direction, on pose $\mathcal{A}_D = \det(\mathcal{A}_N) = (Q_D, \Sigma, \{q_0\}, F_D, \delta_D)$. Il nous faut montrer que $\mathcal{L}(\mathcal{A}_N) = \mathcal{L}(\mathcal{A}_D)$.
Sans perte de généralité, supposons que $\mathcal{A}_N$ soit complet. On peut remarquer que $\mathcal{A}_D$ est complet par construction. On montre l'égalité des deux langages en montrant la propriété suivante. Pour tout mot $v \in \Sigma^*$, soit $R_0, \ldots, R_n$ l'unique chemin dans $\mathcal{A}_D$ pour v. Alors $R_n = \{q \in Q_N \mid \exists q_0 \xrightarrow{v}{}^*{}_{\mathcal{A}_N} q\}$. On montre cette propriété par récurrence sur $|v|$.

- **Cas de base** $|v| = 0$ - On a donc $v = \varepsilon$. Le chemin est réduit à $R_0 = \{q_0\}$. On a bien que $\{q_0\}$ est l'ensemble des états finissant un chemin de $\mathcal{A}_N$ pour ε.
- **Cas** $|v| = n + 1$ - On suppose la propriété vraie pour tout mot de taille n. Montrons qu'elle est vraie pour un mot v de taille $n+1$. On pose $v = ua$ avec $|u| = n$. Considérons le chemin $R_0 \xrightarrow{u}{}^* R_n \xrightarrow{a} R_{n+1}$ dans $\mathcal{A}_D$. Par hypotèse de récurrence, on sait que $R_0 \xrightarrow{u}{}^* R_n$ est un chemin pour u et $R_n = \{q \in Q_N \mid \exists q_0 \xrightarrow{u}{}^*{}_{\mathcal{A}_N} q\}$. Par construction de l'automate déterministe, $R_{n+1} = \delta_D(R_n, a) = \bigcup_{q \in R_n} \delta_N(q, a)$, qui est bien l'ensemble des états pouvant terminer un chemin pour le mot $ua = v$.

ε-NFA vers DFA

Définition 17 (ε-fermeture d'un état)

Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un ε-NFA. On appelle ε-fermeture de l'état $q \in Q$ l'ensemble :

$$E(q) = \{q' \mid q \xrightarrow{\varepsilon}{}^*_{\mathcal{A}} q'\}$$

$E(q)$ représente tous les états de l'automate accessibles depuis q en ne prenant que des transitions spontanées.

Cet ensemble peut être calculé par en parcourant l'automate vu comme un graphe, en partant de q et en ne considérant que les transitions spontanées.

$E(q)$ est utilisé pour supprimer les transitions spontanées d'un automate.

ε-NFA vers DFA

Définition 18

Soit $\mathcal{A}_S = (Q_S, \Sigma, q_0, F_S, \delta_S)$ un ε-NFA. Son déterminisé $\det^{\varepsilon}(\mathcal{A}_S)$ est le DFA $\mathcal{A}_D = (Q_D, \Sigma, E(q_0), F_D, \delta_D)$ construit comme suit :

- $Q_D = \mathcal{P}(Q_S)$
- $F_D = \{S \mid S \in Q_D, S \cap F_S \neq \varnothing\}$
- $\delta_D : \begin{cases} Q_D \times \Sigma \to \mathcal{P}(Q_D) \\ (S, a) \mapsto E\left(\bigcup_{q \in S} \delta_S(q, a)\right) \end{cases}$

ε-NFA vers DFA

Cette construction est similaire à celle par sous-ensembles utilisée pour déterminiser un automate. La seule différence est l'utilisation de l'ε-fermeture pour considérer en même temps tous les états accessibles par une lettre a et un nombre arbitraire de transitions spontanées.

Cette construction peut s'appliquer telle quelle à un automate non déterministe sans transition spontanée puisque pour un tel automate, $E(q) = \{q\}$.

ε-NFA vers DFA

Théorème 19

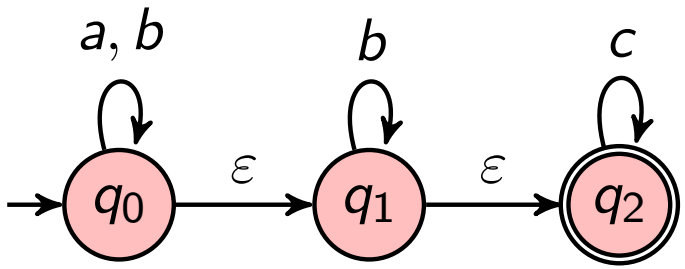
Soit $\mathcal{A}_S = (Q_S, \Sigma, q_0, F_S, \delta_S)$ un ε-NFA. Il existe un DFA $\mathcal{A}_D$ tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_D)$.

Démonstration

On construit l'automate $\mathcal{A}_D = \det^{\varepsilon}(\mathcal{A}_S) = (Q_D, q_0', F_D, \delta_D)$. La propriété à montrer est que, pour tout mot v, si $R_0 \xrightarrow{v}{}^*_{\mathcal{A}_D} R_n$ alors $R_n = E(\{q \mid \exists q_0 \xrightarrow{v}{}^*_{\mathcal{A}_S} q\})$, ce qui se montre par récurrence sur $|v|$.
L'autre direction est triviale : un automate non déterministe est un cas particulier d'automate à transitions spontanées incomplet.

Exemple

Constuisons un DFA associé au ε-NFA suivant.



Exemple

On construit le tableau des ε-fermetures des états q_0, q_1 et q_2.

$E(\{q_0\})$	$\to \{q_0, q_1, q_2\}$	$\to$ état A
$E(\{q_1\})$	$\to \{q_1, q_2\}$	$\to$ état B
$E(\{q_2\})$	$\to \{q_2\}$	$\to$ état C

On a introduit trois nouveaux états associés aux trois ε-fermetures.

Exemple

La fonction de transition δ_D est définie à partir de δ_S.

$$\begin{cases} \delta_D(A,a) & = E(\delta_S(q_0,a) \cup \delta_S(q_1,a) \cup \delta_S(q_2,a)) = E(\{q_0\}) = A \\ \delta_D(A,b) & = E(\delta_S(q_0,b) \cup \delta_S(q_1,b) \cup \delta_S(q_2,b)) = E(\{q_0,q_1\}) = A \\ \delta_D(A,c) & = E(\delta_S(q_0,c) \cup \delta_S(q_1,c) \cup \delta_S(q_2,c)) = E(\{q_2\}) = C \end{cases}$$

$$\begin{cases} \delta_D(B,a) & = E(\delta_S(q_1,a) \cup \delta_S(q_2,a)) = E(\varnothing) = \varnothing \\ \delta_D(B,b) & = E(\delta_S(q_1,b) \cup \delta_S(q_2,b)) = E(\{q_1\}) = B \\ \delta_D(B,c) & = E(\delta_S(q_1,c) \cup \delta_S(q_2,c)) = E(\{q_2\}) = C \end{cases}$$

$$\begin{cases} \delta_D(C,a) & = E(\delta_S(q_2,a)) = E(\varnothing) = \varnothing \\ \delta_D(C,b) & = E(\delta_S(q_2,b)) = E(\varnothing) = \varnothing \\ \delta_D(C,c) & = E(\delta_S(q_2,c)) = E(\{q_2\}) = C \end{cases}$$

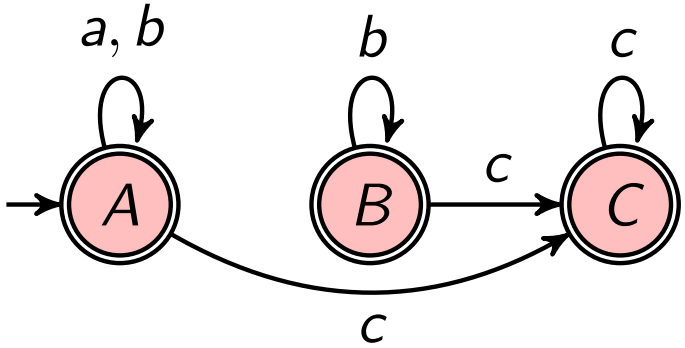
Exemple

Finalement, le DFA $\mathcal{A}_D$ recherché est défini par :

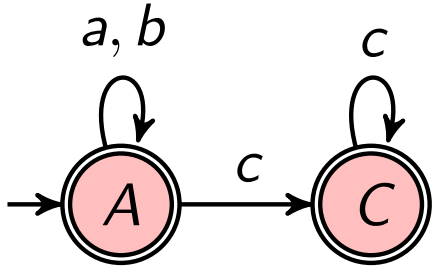
- l'ensemble de ses **états** $Q_D = \{A, B, C\}$;
- son **état initial** $A = E(q_0)$;
- l'ensemble de ses **états acceptants** $F_D = \{A, B, C\}$
- la fonction δ_D définie précédemment.

Exemple

Le DFA recherché admet la représentation graphique suivante.



L'état B est inutile. On peut émonder l'automate.



ε-NFA vers NFA

On peut supprimer les transitions spontanées pour produire un automate **non déterministe**, sans augmenter le nombre d'états. L'automate non déterministe possède donc le même nombre d'états que l'automate initial.

Certains états pouvant devenir inaccessibles (en particulier ceux qui ne sont reliés que par des transitions spontanées), on peut émonder cet automate pour éliminer de tels états.

Il suffit ensuite d'utiliser un algorithme efficace (sans retour sur trace) pour vérifier qu'un mot est reconnu par un automate non déterministe.

Ainsi, supprimer les transitions vides permet généralement de meilleures performances pour cet algorithme.

ε-NFA vers NFA

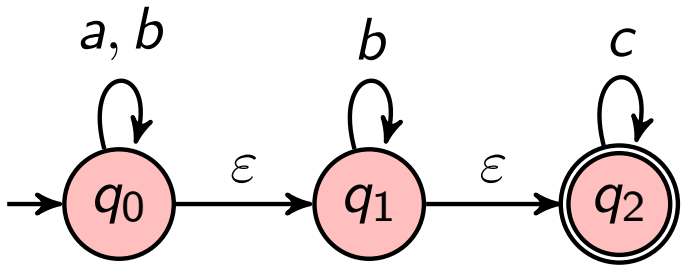
Définition 20

Soit $\mathcal{A}_S = (Q_S, \Sigma, q_0, F_S, \delta_S)$ un ε-NFA. On appelle $\mathrm{rm}^{\varepsilon}(\mathcal{A}_S)$ le NFA $\mathcal{A}_N = (Q_N, \Sigma, q_0, F_N, \delta_N)$ construit comme suit :

- $Q_N = Q_S$
- $F_N = \{q \in Q_S \mid E(q) \cap F_S \neq \varnothing\}$
- $\delta_N : \begin{cases} Q_N \times \Sigma \to \mathcal{P}(Q_N) \\ (q, a) \mapsto \bigcup\limits_{p \in E(q)} \delta_S(p, a) \end{cases}$

Exemple

Construisons un NFA associé au même ε-NFA que précédemment.



On rappelle les ε-fermetures des états q_0, q_1, q_2, utiles pour déterminer la fonction de transition δ_N.

$E(\{q_0\})$	$\to \{q_0, q_1, q_2\}$
$E(\{q_1\})$	$\to \{q_1, q_2\}$
$E(\{q_2\})$	$\to \{q_2\}$

Exemple

La fonction de transition δ_D est définie à partir de δ_S.

$$\begin{cases} \delta_N(q_0, a) & = \delta_S(q_0, a) \cup \delta_S(q_1, a) \cup \delta_S(q_2, a) = \{q_0\} \\ \delta_N(q_0, b) & = \delta_S(q_0, b) \cup \delta_S(q_1, b) \cup \delta_S(q_2, b) = \{q_0, q_1\} \\ \delta_N(q_0, c) & = \delta_S(q_0, c) \cup \delta_S(q_1, c) \cup \delta_S(q_2, c) = \{q_2\} \end{cases}$$

$$\begin{cases} \delta_N(q_1, a) & = \delta_S(q_1, a) \cup \delta_S(q_2, a) = \varnothing \\ \delta_N(q_1, b) & = \delta_S(q_1, b) \cup \delta_S(q_2, b) = \{q_1\} \\ \delta_N(q_1, c) & = \delta_S(q_1, c) \cup \delta_S(q_2, c) = \{q_2\} \end{cases}$$

$$\begin{cases} \delta_N(q_2, a) & = \delta_S(q_2, a) = \varnothing \\ \delta_N(q_2, b) & = \delta_S(q_2, b) = \varnothing \\ \delta_N(q_2, c) & = \delta_S(q_2, c) = \{q_2\} \end{cases}$$

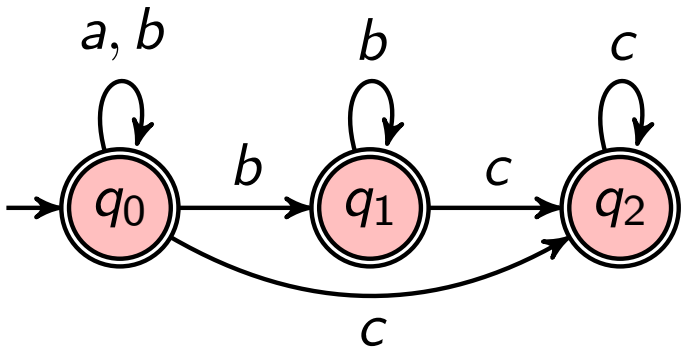
Exemple

Finalement, le NFA $\mathcal{A}_N$ recherché est défini par :

- l'ensemble de ses **états** $Q_N = \{q_0, q_1, q_2\}$;
- son **état initial** q_0 ;
- l'ensemble de ses **états acceptants** $F_N = \{q_0, q_1, q_2\}$
- la fonction δ_N définie précédemment.

Exemple

Le NFA recherché admet la représentation graphique suivante.



Fermture transitive des ε-transitions

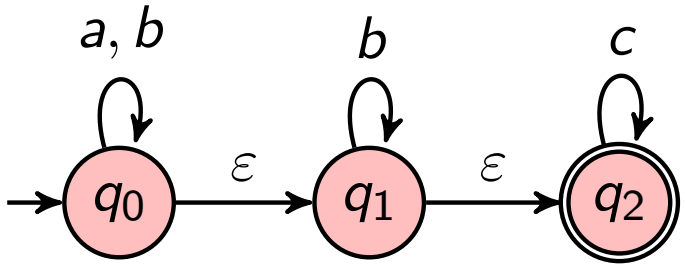
La construction des fermetures ε-transitions permet la construction d'un ensemble de ε-transitions. Cet ensemble est appelé **fermeture transitive des ε-transitions**. On parle également de **clôture transitive**.

La construction de la fermeture transitive des ε-transitions permet ensuite l'élimination des ε-transitions. On distingue deux procédures d'éliminations par **fermeture avant** ou par **fermeture arrière** qui sont illustrées dans les diapositives suivantes.

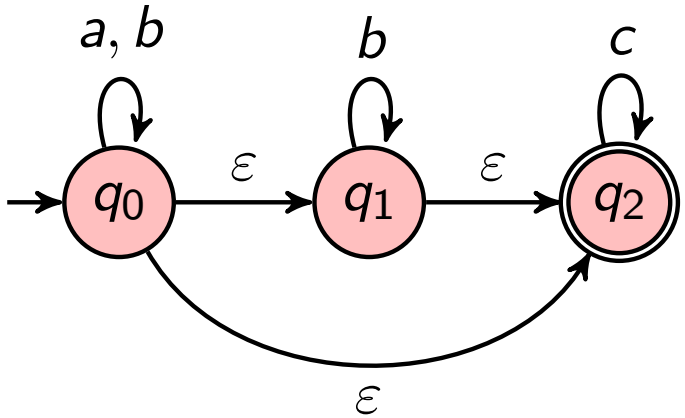
Élimination des ε-transitions par fermeture avant

- Remplacer le graphe par celui de la **fermeture transitive** de ses ε-transitions.
- Si pour une transition $p \xrightarrow{\varepsilon} q$, p est un **état initial** alors q devient un **état initial**.
- Pour toute transition $p \xrightarrow{\varepsilon} q$ et toute transition $r \xrightarrow{a} p$ où $a \in \Sigma$, $a \neq \varepsilon$, **ajouter la transition** $r \xrightarrow{a} q$ et **supprimer l'ε-transition** $p \xrightarrow{\varepsilon} q$.

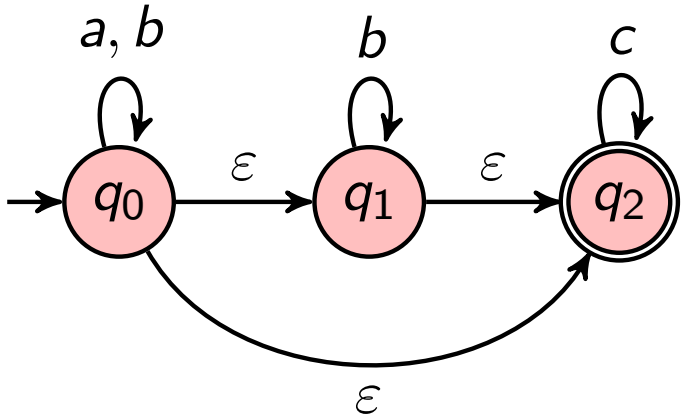
Élimination des ε-transitions par fermeture avant



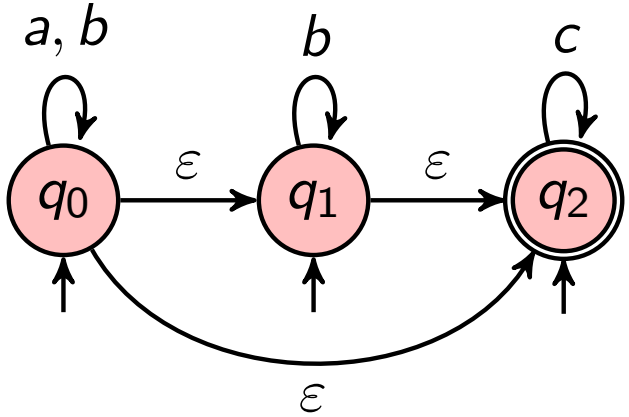
Fermeture transitive des ε-transitions.



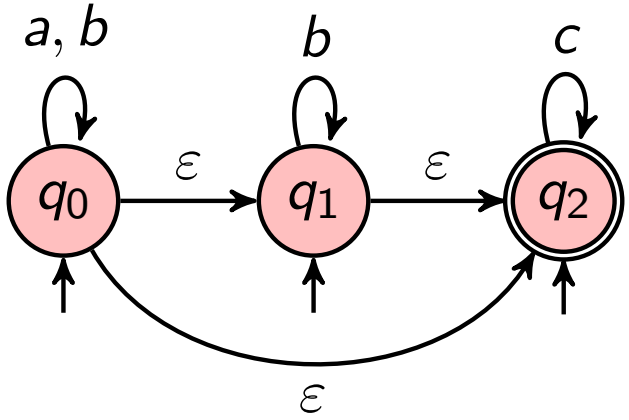
Élimination des ε-transitions par fermeture avant



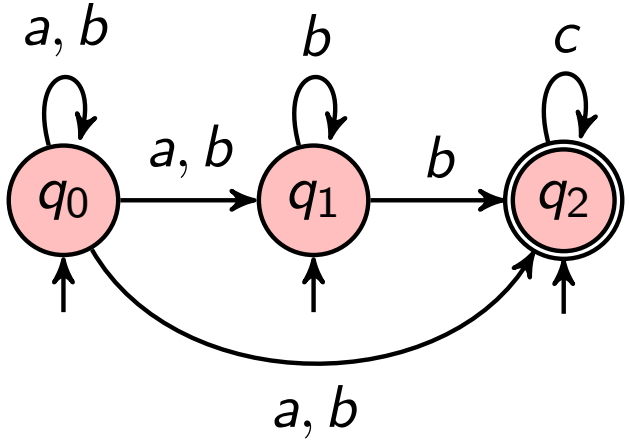
Ajout des états initiaux



Élimination des ε-transitions par fermeture avant



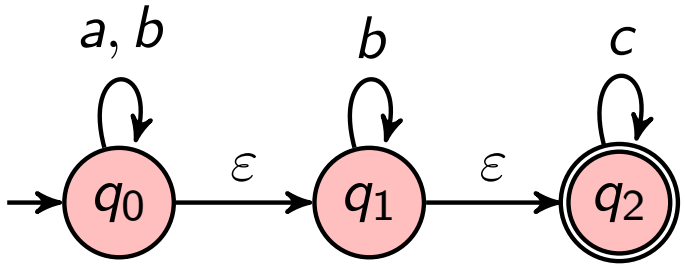
Élimination des ε-transitions et ajout des transitions étiquetées



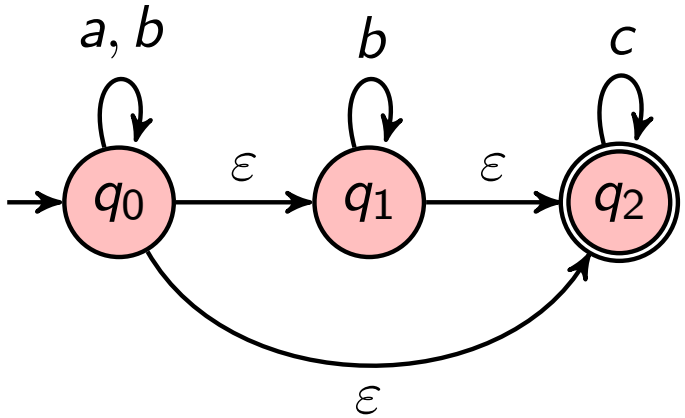
Élimination des ε-transitions par fermeture arrière

- Remplacer le graphe par celui de la **fermeture transitive** de ses ε-transitions.
- Si pour une transition $p \xrightarrow{\varepsilon} q$, q est un **état acceptant** alors p devient un **état acceptant**.
- Pour toute transition $p \xrightarrow{\varepsilon} q$ et toute transition $q \xrightarrow{a} r$ où $a \in \Sigma$, $a \neq \varepsilon$, **ajouter la transition** $p \xrightarrow{a} r$ et **supprimer l'ε-transition** $p \xrightarrow{\varepsilon} q$.

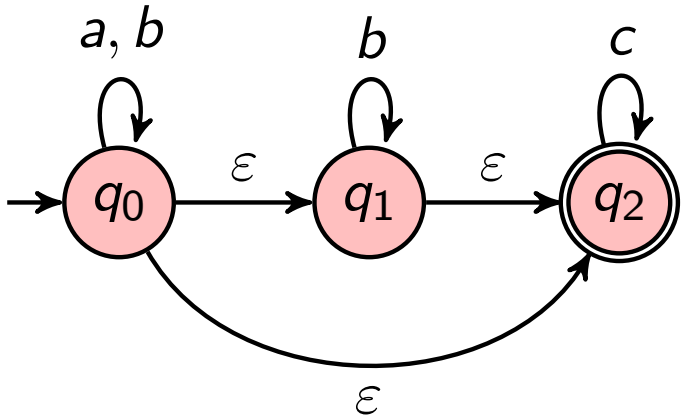
Élimination des ε-transitions par fermeture arrière



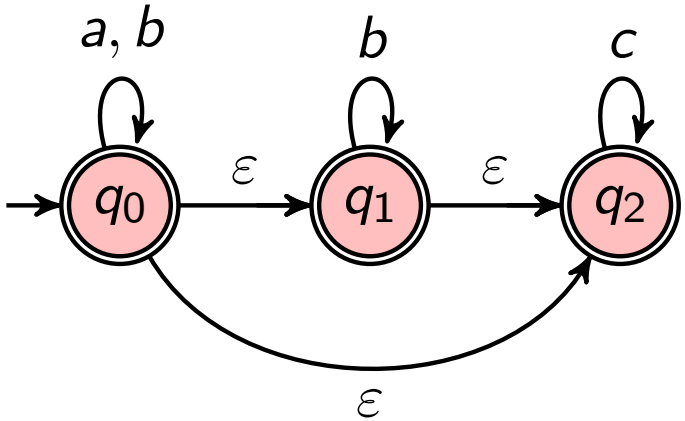
Fermeture transitive des ε-transitions.



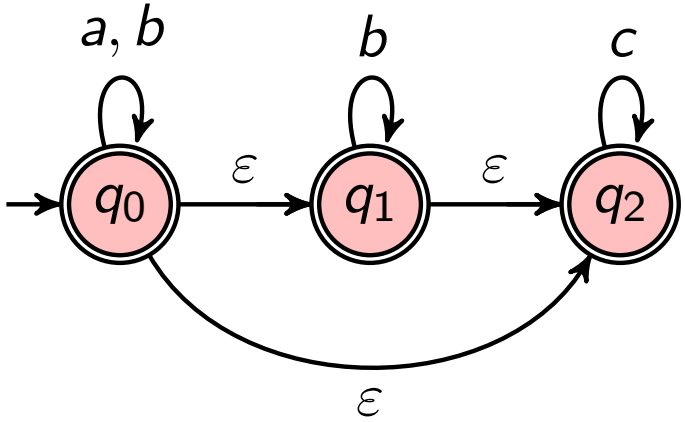
Élimination des ε-transitions par fermeture arrière



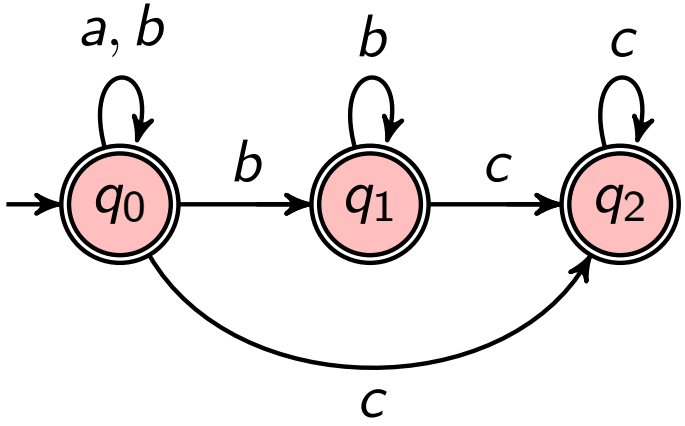
Ajout des états acceptants



Élimination des ε-transitions par fermeture arrière



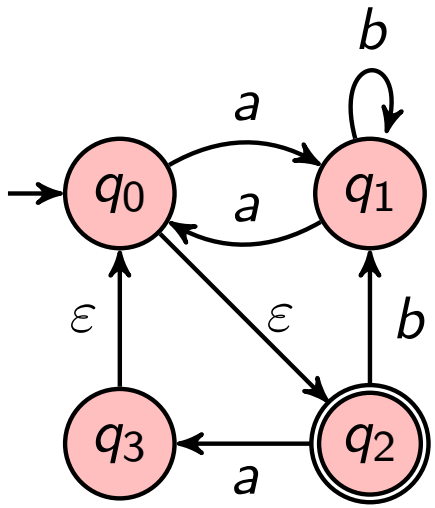
Élimination des ε-transitions et ajout des transitions étiquetées



Ce graphe n'est autre que celui obtenu lors de la construction du NFA à partir du ε-NFA.

Exemple

Éliminer les ε-transitions par fermeture avant et par fermeture arrière pour l'automate suivant.



Langage reconnaissable

Langage reconnaissable

L'équivalence des trois modèles d'automates étant établie, on peut parler de langage **reconnaissable** sans avoir besoin de préciser par quel type d'automate fini. Ce qui permet d'étendre la définition d'un **langage reconnaissable** donnée au début de ce chapitre.

Définition 21 (langage reconnaissable)

Soit Σ un alphabet. Un langage $L \subseteq \Sigma^*$ est dit **reconnaissable** s'il existe un automate $\mathcal{A}$ tel que $L = \mathcal{L}(\mathcal{A})$.