Théorie des Langages

Cours 2 : Induction, automates finis

Lionel Rieg

Grenoble INP - Ensimag, 1re année

Rappels sur les langages

Définitions vues au cours précédent :

- Alphabet/vocabulaire V,   lettre/symbole $x \in V$
- Mot w,   mot vide ε,
- taille $|w|$,   occurrences $|w|_x$
- Concaténation $w_1.w_2$, $L_1.L_2$
- Itération w^n, L^n, $L^0 = \{\varepsilon\}$, L^*
- Langage $L \subseteq V^*$
- Expression régulière $\emptyset,\ \epsilon,\ x,\ +,\ .,\ ^*$
- Définition de langages par point fixe

Comment définir un langage / un ensemble ?

- Par extension
  - On énumère les éléments de l'ensemble.
  - $\{a, b\}^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$
  - $P = \{0, 2, 4, 6, 8, 10, \ldots\}$
- Par compréhension
  - On décrit les caractéristiques des éléments de l'ensemble.
  - $P = \{n \in \mathbb{N} \mid \exists k \in \mathbb{N}, n = 2k\}$
- Par induction structurelle
  - On explique comment construire les éléments de l'ensemble.
  - Fréquemment utilisé en informatique
  - P est le **plus petit** ensemble (pour l'inclusion) tel que :
    - $0 \in P$, et
    - si $n \in P$ alors $n + 2 \in P$.

Définition par induction structurelle

Principe général : on définit un ensemble en spécifiant :

- Des cas de base : quels sont les éléments les « plus simples » de l'ensemble ?
- Des règles de construction : comment peut-on, en partant d'éléments de l'ensemble, en construire de nouveaux ?

Exemples

Définitions inductives de V^* et $L \stackrel{\text{def}}{=} \{a^n b^n \mid n \geqslant 1\}$:

V^* : **Base :** $\varepsilon \in V^*$
Induction : pour tout x de V, si $w \in V^*$ alors $xw \in V^*$

L : **Base :** $ab \in L$
Induction : si $w \in L$ alors $awb \in L$

Définition générale

Définition (Ensemble inductif)

Soit U un ensemble ; définir un ensemble $E \subseteq U$ par induction structurelle consiste à donner :

- un ensemble non vide d'atomes $B = \{b_1, \ldots, b_n\} \subseteq U$
- un ensemble $K = \{\kappa_1, \ldots, \kappa_m\}$ de constructeurs inductifs, où $\kappa_i : U^{a_i} \to U$ et $a_i > 0$ pour tout i (a_i : arité de κ_i)

E est alors le plus petit ensemble tel que :

- $B \subseteq E$
- $\forall i \in [1, m]$, si $(e_1, \ldots, e_{a_i}) \in E^{a_i}$, alors $\kappa_i(e_1, \ldots, e_{a_i}) \in E$

Lien avec le thèorème du point fixe :

$$E = \mu(f) \quad \text{avec} \quad f \ : \ X \mapsto B \cup \bigcup_{\kappa_i \in K} \kappa_i(X, \ldots, X)$$

$f^0(\emptyset) = \emptyset \qquad f^1(\emptyset) = B \qquad f^2(\emptyset) = \ldots$

Exemples

Exemples

V^*, listes, arbres, formules logiques, ...

Posons $V = \{a, b\}$, et soit L_0 le langage défini par induction structurelle de la façon suivante :

- Base : $\varepsilon \in L_0 \qquad (b_1 = \varepsilon)$
- Induction (constructeurs) : si $u, v \in L_0$ alors
  - $a\,u\,b\,v \in L_0 \qquad (\kappa_1(u, v) = a\,u\,b\,v)$
  - $b\,u\,a\,v \in L_0 \qquad (\kappa_2(u, v) = b\,u\,a\,v)$

Quelques mots dans L_0 : $\quad \varepsilon,\ ab,\ ba,\ aabb,\ abab,\ bbaaaabb$

Exercice : construire les trois derniers mots

- $aabb$: 1 seule façon de construire : $a\,(a\,\varepsilon\,b\,\varepsilon)\,b\,\varepsilon \;=\; \kappa_1(\kappa_1(b_1, b_1), b_1)$
- $abab$: 2 façons : $a\,(b\,\varepsilon\,a\,\varepsilon)\,b\,\varepsilon$ et $a\,\varepsilon\,b\,(a\,\varepsilon\,b\,\varepsilon)$
- $bbaaaabb$: 1 seule façon : $b\,(b\,\varepsilon\,a\,\varepsilon)\,a\,(a\,(a\,\varepsilon\,b\,\varepsilon)\,b\,\varepsilon)$

Énumération d'un ensemble inductif

Théorème (corollaire du point fixe de Kleene)

Soit E un ensemble défini par induction sur l'ensemble d'atomes B et l'ensemble de constructeurs K.
Alors $E = \bigcup_{n \geqslant 0} E_n$, où la suite (E_n) est définie par :

$$\begin{aligned} E_0 &\stackrel{def}{=} B \\ E_{n+1} &\stackrel{def}{=} E_n \cup \{\kappa_i(e_1, \ldots, e_{a_i}) \mid \kappa_i \in K, e_1, \ldots, e_{a_i} \in E_n\} \end{aligned}$$

algorithme $E =$
- $n \leftarrow 0$, $E_0 \leftarrow B$
- **répéter**
  - $E_{n+1} \leftarrow E_n \cup \{\kappa_i(e_1, \ldots, e_{a_i}) \mid \kappa_i \in K, e_1, \ldots, e_{a_i} \in E_n\}$
  - $n \leftarrow n + 1$
- **jusqu'à** $E_{n+1} = E_n$
- **renvoyer** E_n

Question : Est-ce que ça termine toujours ?

Quand B et K sont finis, l'algorithme termine ssi E est fini.

Fonction définie inductivement

Définition

Soit E un ensemble défini inductivement par l'ensemble d'atomes B et l'ensemble de constructeurs K, et soit U' un ensemble quelconque.
Pour définir une fonction $f : E \to U'$, il suffit d'expliciter :

- les images des atomes $f(b_1), \dots, f(b_n)$;
- la façon dont la fonction « interagit » avec les constructeurs : comment exprimer $f(\kappa_i(e_1, \dots, e_{a_i}))$ en fonction de $f(e_1), \dots, f(e_{a_i})$.

RMQ : $f(E)$ est un ensemble inductif !
Correspond au lemme de commutation de point fixe

Exemple (Longueur d'un mot)

$|\_| : V^* \to \mathbb{N}$

- Cas de base : $|\varepsilon| = 0$
- Constructeurs inductifs : $|xw| = 1 + |w|$

Lien avec la programmation récursive

L'induction structurelle formalise la programmation récursive !

Exemple (Somme des éléments d'une liste)

- En OCaml :

```
let rec somme_list l =
  match l with
    | []       -> 0
    | x :: xs -> x + somme_list xs
```

- En Haskell :

```
sommeList :: [Int] -> Int
sommeList [] = 0
sommeList (x:xs) = x + sommeList xs
```

Exemples

Soient les fonctions dl et $\mathrm{pa} : L_0 \to \mathbb{N}$ telles que $\forall w \in L_0$,

$$\begin{array}{rcl} \mathrm{dl}(w) & = & |w|_a \\ \mathrm{pa}(w) & = & \max \{|x| \mid x \text{ préfixe de } w \text{ et } x \in \{a\}^*\} \end{array}$$

Exercice : définir les fonctions dl et pa par induction structurelle

Rappel de L_0 :
- Base : $\varepsilon \in L_0$
- Induction : si $u, v \in L_0$ alors $a\,u\,b\,v \in L_0$
  si $u, v \in L_0$ alors $b\,u\,a\,v \in L_0$

L_0 est défini par 1 cas de base et 2 constructeurs donc 3 cas à considérer.

dl
- $\mathrm{dl}(\varepsilon) = 0$
- $\forall u, v \in L_0,\ \mathrm{dl}(a\,u\,b\,v) = 1 + \mathrm{dl}(u) + \mathrm{dl}(v)$
- $\forall u, v \in L_0,\ \mathrm{dl}(b\,u\,a\,v) = 1 + \mathrm{dl}(u) + \mathrm{dl}(v)$

pa
- $\mathrm{pa}(\varepsilon) = 0$
- $\forall u, v \in L_0,\ \mathrm{pa}(a\,u\,b\,v) = 1 + \mathrm{pa}(u)$
- $\forall u, v \in L_0,\ \mathrm{pa}(b\,u\,a\,v) = 0$

Preuve par induction structurelle

Définition

Soit E un ensemble défini inductivement par l'ensemble d'atomes B et l'ensemble de constructeurs K, et soit P une propriété sur E.
Pour montrer que $P(e)$ est vraie pour tout $e \in E$, on peut :

- montrer que $P(b_1), \ldots, P(b_n)$ sont vrais ;
- pour tout $\kappa_i \in K$, montrer que si $P(e_1), \ldots, P(e_{a_i})$ sont tous vrais, alors $P(\kappa_i(e_1, \ldots, e_{a_i}))$ l'est également.

Remarque

La preuve par induction structurelle est une généralisation de la preuve par récurrence :

- *Base :* 0
- *Constructeur : la fonction successeur* $s : n \mapsto n + 1$

Démo Coq/Rocq

Application

Soit $M_0 = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$. Montrer que $L_0 \subseteq M_0$.

Propriété $P(w)$ sur L_0 : $w \in M_0$.
Par induction structurelle :

- Base : $|\varepsilon|_a = |\varepsilon|_b$. Ok
- Induction : 2 constructeurs κ_1 et κ_2
  - $\kappa_1(\_,\_)$:
    Soient u et v deux mots de L_0. Supposons $P(u)$ et $P(v)$, c.-à-d. $|u|_a = |u|_b$ et $|v|_a = |v|_b$.
    Alors : $|a\,u\,b\,v|_a \;=\; 1 + |u|_a + |v|_a \;=\; 1 + |u|_b + |v|_b \;=\; |a\,u\,b\,v|_b$
  - $\kappa_2(\_,\_)$:
    De la même façon :
    $|b\,u\,a\,v|_a \;=\; 1 + |u|_a + |v|_a \;=\; 1 + |u|_b + |v|_b \;=\; |b\,u\,a\,v|_b$

Exercice $(\star)$: Montrer que $M_0 \subseteq L_0$.

Définition des expressions régulières

Exercice : Définir par induction structurelle l'ensemble des expressions régulières sur un vocabulaire V quelconque.

- **Base :**
  - $\emptyset$ est une expression régulière
  - ϵ est une expression régulière
  - Si $x \in V$, alors x est une expression régulière
- **Induction :** Si E_1 et E_2 sont des expressions régulières, alors
  - $E_1 + E_2$ est une expression régulière
  - $E_1.E_2$ est une expression régulière
  - ${E_1}^*$ est une expression régulière

. et * sont ici des constructeurs, pas des opérations sur les langages !

Langage représenté par une ER

Une expression régulière sur V est un mot sur $V \cup \{\varnothing, \epsilon,), (, +, ^*, .\}$ qui représente un langage sur V.

Exercice : Définir $\mathcal{L}(E)$ le langage représenté par une ER E par induction structurelle.

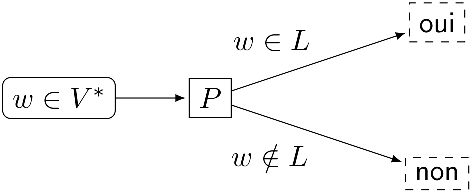
- **Base :**
  - Si $E = \varnothing$ alors $\mathcal{L}(E) = \emptyset$
  - Si $E = \epsilon$ alors $\mathcal{L}(E) = \{\varepsilon\}$
  - Si $E = x \quad (x \in V)$ alors $\mathcal{L}(E) = \{x\}$
- **Induction :**
  - Si $E = E_1 + E_2$ alors $\mathcal{L}(E) = \mathcal{L}(E_1) \cup \mathcal{L}(E_2)$
  - Si $E = E_1.E_2$ alors $\mathcal{L}(E) = \mathcal{L}(E_1).\mathcal{L}(E_2)$
  - Si $E = {E_1}^*$ alors $\mathcal{L}(E) = \mathcal{L}(E_1)^*$

à gauche : constructeurs des ER
à droite : opérations sur les langages

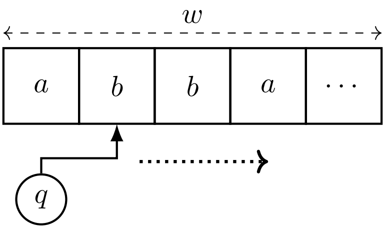
Automates finis

Les automates finis

On s'intéresse à définir des « programmes » qui reconnaissent des langages.



Les programmes les plus « simples » sont les automates finis.

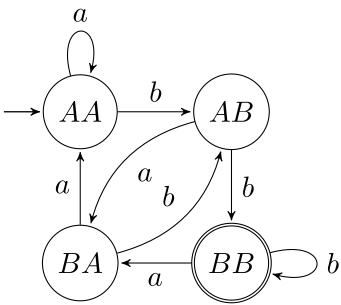


À chaque pas d'exécution, l'automate peut changer d'état et/ou lire un symbole et se positionner sur le symbole suivant.

Exemple d'automate fini

Voici un automate fini.
Il contient :

- des états
- le vocabulaire
- des transitions
- des états initiaux
- des états acceptants



$$\mathcal{L}(\mathcal{A}) = V^*.\{bb\}$$

Utilisation d'un automate :

1. commencer dans un état initial
2. lire les lettres d'un mot en suivant les transitions
3. à la fin du mot, regarder si l'état est acceptant

Définition formelle

Définition

Un automate fini (AF) est un quintuplet $\langle Q, V, \delta, I, F \rangle$, où :

- Q est un ensemble fini d'états
- V est le vocabulaire d'entrée
- $\delta \subseteq Q \times (V \cup \{\varepsilon\}) \times Q$ est la relation de transition
- $I \subseteq Q$ est l'ensemble des états initiaux
- $F \subseteq Q$ est l'ensemble des états acceptants (ou finaux ou finals)

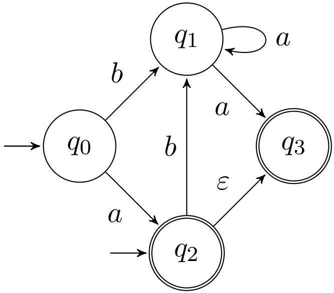
Relation de transition

- Pour $a \in V$, si $(p, a, q) \in \delta$, alors étant dans l'état p et lisant un a, l'automate peut passer dans l'état q et avancer dans le mot.
- Si $(p, \varepsilon, q) \in \delta$, alors étant dans l'état p, l'automate peut passer à l'état q sans avancer dans le mot.

Exemple et représentation graphique

Soit A l'automate défini par :

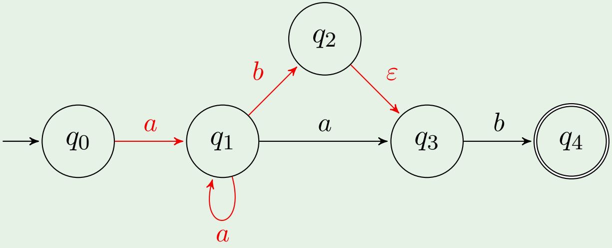
$$
\begin{array}{rcl}
Q & = & \{q_0, q_1, q_2, q_3\}\,,\ V \ = \ \{a, b\}\,, \\
\delta & = & \{(q_0, b, q_1), (q_0, a, q_2), (q_1, a, q_1), (q_1, a, q_3), (q_2, b, q_1), (q_2, \varepsilon, q_3)\} \\
I & = & \{q_0, q_2\}\,,\ F \ = \ \{q_2, q_3\}
\end{array}
$$



Chemins et langage reconnu par un automate

$\mathcal{L}(A)$
= « Ens. des mots permettant de passer d'un état initial à un état final »
= « Les mots qui étiquettent un chemin d'un état initial à un état final »

Exemple (Chemins pour aab)



$$\begin{aligned}
\chi_1 &= (q_0, a, q_1)(q_1, a, q_3)(q_3, b, q_4) \\
\chi_2 &= (q_0, a, q_1)(q_1, a, q_1)(q_1, b, q_2)(q_2, \varepsilon, q_3)
\end{aligned}$$

Chemins et langage reconnu par un automate

$\mathcal{L}(A)$
$=$ « Ens. des mots permettant de passer d'un état initial à un état final »
$=$ « Les mots qui étiquettent un chemin d'un état initial à un état final »

Définition (Chemin)

Soit $A = \langle Q, V, \delta, I, F\rangle$ un automate. L'ensemble des chemins dans A est défini inductivement de la façon suivante :

Base Pour tout $p \in Q$, $()$ est un chemin (vide) dans A de p à p ;

Induction Pout tous $p, q, q' \in Q$ et $a \in V \cup \{\varepsilon\}$,
si $(p, a, q) \in \delta$ et χ est un chemin dans A de q à q',
alors $(p, a, q).\chi$ est un chemin dans A de p à q'.

Convention

Dans un chemin non vide, on ne note en général pas le « () » final.

Trace, longueur d'un chemin

Définition

Soit $(q_0, a_1, q_1)(q_1, a_2, q_2) \cdots (q_{n-1}, a_n, q_n)()$ un chemin dans A.
Ce chemin est de longueur n et de trace $a_1 a_2 \cdots a_n$.

$$\mathsf{lgr} : \begin{cases} () & \mapsto \quad 0 \\ (p, a, q)\chi & \mapsto \quad 1 + \mathsf{lgr}(\chi) \end{cases} \qquad \mathsf{tr} : \begin{cases} () & \mapsto \quad \varepsilon \\ (p, a, q)\chi & \mapsto \quad a \cdot \mathsf{tr}(\chi) \end{cases}$$

Exemples

$\chi_1 = (q_0, a, q_1)(q_1, a, q_3)(q_3, b, q_4)()$ longueur : 3 trace : aab

$\chi_2 = (q_0, a, q_1)(q_1, a, q_1)(q_1, b, q_2)(q_2, \varepsilon, q_3)()$ longueur : 4 trace : aab

Définitions (Mot reconnu par un automate, Langage régulier)

Un mot w est reconnu par A si et seulement si
il existe un chemin dans A d'un état initial à un état final, de trace w.
On appelle langage régulier tout langage reconnu par un automate fini.

Comment construire un automate ?

1. Identifier quelles informations sont nécessaires pour reconnaître les mots du langage.
2. Définir Q comme les valeurs possibles de ces informations.
3. Construire les transitions de δ comme la mise à jour des informations.
4. Déterminer les états acceptants F.

Exemple (Nombres multiples de 3 en base 10)

1. Critère de divisibilité : la somme itérée des chiffres est multiple de 3
2. $Q = \{0, \ldots, 9\}$
3. $\delta = \{(q, x, q') \mid q + x = q' \vee q + x = 9 + q'\}$
4. $F = \{0, 3, 6, 9\}$

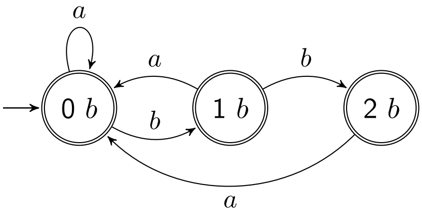
Ce n'est pas forcément la solution optimale mais elle fonctionne toujours.

Exercices

Exercice 1

Construire un automate fini qui reconnaît le langage

$$L = \{w \in \{a,b\}^* \mid w \text{ ne contient } \textbf{pas plus de} \text{ deux } b \text{ consécutifs}\}$$

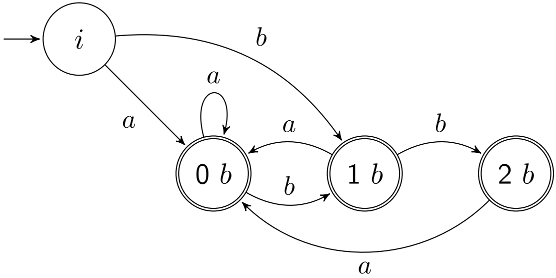


Exercices

Exercice 2

Construire un automate fini qui reconnaît le langage

$$L = \left\{ w \in \{a,b\}^+ \;\middle|\; w \text{ ne contient } \textbf{pas plus de} \text{ deux } b \text{ consécutifs} \right\}$$



Automate non-déterministe

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

1. Card$(I) > 1$ (plus d'un état intial), et/ou
2. $\exists\,(q, a, p)$ et $(q, a, r) \in \delta$ avec $p \neq r$, et/ou
3. $\exists\,(q, \varepsilon, p) \in \delta$

Dans les trois cas, « on ne sait pas quoi faire » :

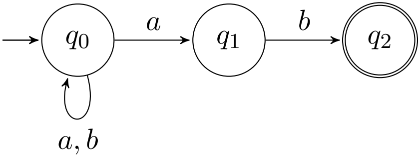
1. « où dois-je commencer ? »
2. « je suis en q, je vois le symbole a, où vais-je ? »
3. « je suis en q, $\forall$ symbole je peux choisir de passer en p ou non »

Non-déterminisme :

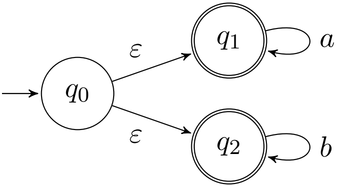
- ne donne pas immédiatement un « programme » reconnaisseur
- mais facilite la définition des automates !

Exemples

1. Non-déterministe, sans ε-transition



2. Non-déterministe, avec ε-transition



Automate déterministe

À l'inverse, un AF $\langle Q, V, \delta, I, F \rangle$ est dit déterministe si

1. Card$(I) = 1$ (exactement un état intial), et
2. Si (q, a, p) et $(q, a, r) \in \delta$, alors $p = r$, et
3. $\nexists\, (q, \varepsilon, p) \in \delta$

Ainsi, « on sait toujours quoi faire » : les transitions possibles sont uniques.

Conséquences de la définition

- L'automate a un seul état initial
- δ est une fonction partielle : $Q \times V \rightharpoonup Q$:
  Si $(p, a, q) \in \delta$, on pourra noter $\delta(p, a) = q$
- Donne directement un « programme » reconnaisseur
- Mais certaines transitions peuvent manquer !

Automate complet

Définition

Un automate est complet si de chaque état et chaque symbole, une transition est toujours possible : $\forall (q, a) \in Q \times V, \exists p \in Q, (q, a, p) \in \delta$.

Pour un AF déterministe complet, δ est une fonction totale : $Q \times V \to Q$.
Un automate peut être non-déterministe mais complet !

Comment compléter un automate (sans changer son langage) ?

1. Ajouter un état puits : un état non acceptant qui boucle sur lui-même pour tous les symboles.
2. Ajouter toutes les transitions manquantes vers cet état.

Questions : Veut-on toujours un automate déterministe complet ?
Est-ce toujours mieux ?

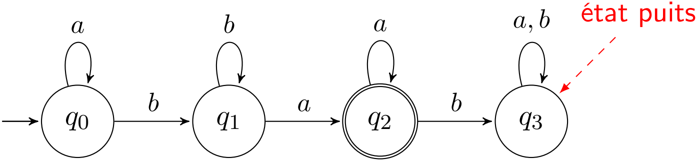
Extension de la fonction de transition aux mots

Définition

Soit $A = \langle Q, V, \delta, \{i\}, F\rangle$ un AFD complet. On définit la fonction $\delta^* : Q \times V^* \to Q$ par induction de la façon suivante : pour tout $p \in Q$,

- $\delta^*(p, \varepsilon) = p$
- $\delta^*(p, aw) = \delta^*(\delta(p, a), w)$

Exemple :



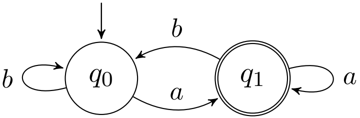
$\delta^*(q_0, aab) = q_1$ et $\delta^*(q_0, abba) = q_2$

L'extension de la fonction de transition est parfois notée δ au lieu de δ^*.

Comme les chemins sont uniques : $\mathcal{L}(A) = \{w \in V^* \mid \delta^*(i, w) \in F\}$

Test d'appartenance pour les AFD complets

$L = \{a,b\}^* \{a\}^+$



$$\begin{aligned} bbaba \in L \iff \delta^*(q_0, bbaba) \in F &\iff \delta^*(q_0, baba) \in F \\ \iff \delta^*(q_0, aba) \in F &\iff \delta^*(q_1, ba) \in F \\ \iff \delta^*(q_0, a) \in F &\iff \delta^*(q_1, \varepsilon) \in F \iff q_1 \in F \end{aligned}$$

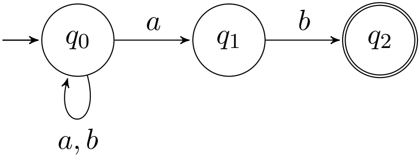
fonction reconnaître(q : état, w : mot) **renvoie Booléen** =
  tant que $w \neq \varepsilon$ **faire**
    $s \leftarrow$ premier_symbole(w)
    $w \leftarrow$ reste_mot(w)
    $q \leftarrow \delta(q, s)$
  fin tant que
  renvoyer $(q \in F)$

$\forall w \in V^*$, $w \in \mathcal{L}(A)$ si et seulement si reconnaître(q_0, w) = **vrai**

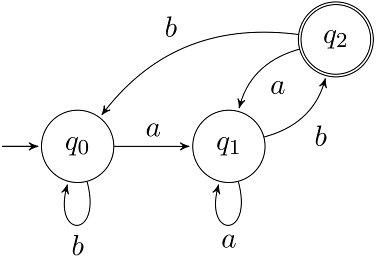
Automates équivalents

Deux automates A et A' sont équivalents ssi $\mathcal{L}(A) = \mathcal{L}(A')$.

- Automate A :



- Automate A' :

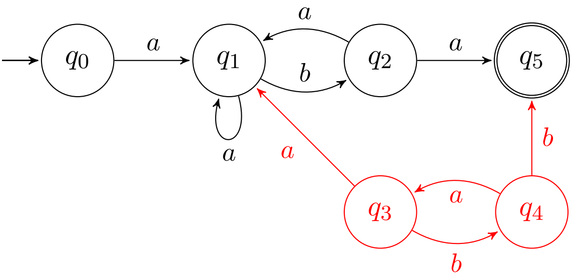


États accessibles

Définition

Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate.
Un état $p \in Q$ est accessible si on peut passer d'un état $q_0 \in I$ à p en se servant des transitions de δ.
Un automate est initialement connecté si tous ses états sont accessibles.

Exemple :

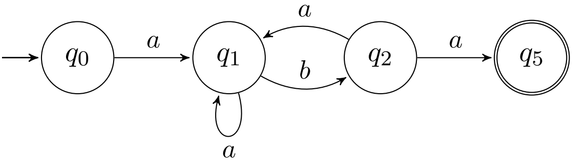


États accessibles

Définition

Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate.
Un état $p \in Q$ est accessible si on peut passer d'un état $q_0 \in I$ à p en se servant des transitions de δ.
Un automate est initialement connecté si tous ses états sont accessibles.

Exemple :



Un AFD complet $A = \langle Q, V, \delta, \{i\}, F \rangle$ est initialement connecté ssi

$$\forall p \in Q,\ \exists w \in V^*,\ \delta^*(i, w) = p$$