Théorie des langages et automates

Automates à états finis

Clément Moreau[1,2]

[1]BRED – Banque Populaire — clement.moreau@bred.fr

[2]Université de Tours

Sommaire

Table of Contents

Mots et alphabet

- Un *alphabet*, noté Σ, est un ensemble fini de symboles.
- Un *mots* (i.e. chaîne de caractères ou string) sur Σ est une suite finie de symboles appartement à Σ.

Exemples

Soit $\Sigma = \{0, 1\}$, alors $x = 10001$ est un mot sur Σ de longueur 5.

Comme tout peut-être représenté en binaire au sein d'un ordinateur, la notion d'alphabet permet de réduire et encoder tout problème en une chaîne de caractères / mot.

Mots et alphabet

Quelques notations :

- La taille d'un mot x est notée $|x|$.
- Il existe un unique mot de longueur 0, ou mot vide, sur tout alphabet Σ, noté ε. Ainsi, $|\varepsilon| = 0$.
- On note Σ^n, L'ensemble des mots de longueur n.
- L'ensemble des mots sur Σ est noté Σ^*. Par exemple :
  - $\{a, b\}^* = \{\varepsilon, a, b, ab, ba, aa, bb, aba, ...\}$
  - $\{a\}^* = \{\varepsilon, a, aa, aaa, ...\} = \{a^n | n \geq 0\}$
  - Par convention, $\emptyset^* = \{\varepsilon\}$

Opérations sur les mots

L'opération de *concaténation*, notée $\cdot$, prend deux mots $x, y \in \Sigma^*$ et retourne un nouveau mot $x \cdot y$ qui consiste à coller y en bout de x.

Exemples

Soit $\Sigma = \{a, b\}, x = abab$ et $y = aab$. Alors $x \cdot y = ababaab$.

Opérations sur les mots

L'opération de *concaténation*, notée $\cdot$, prend deux mots $x, y \in \Sigma^*$ et retourne un nouveau mot $x \cdot y$ qui consiste à coller y en bout de x.

Exemples

Soit $\Sigma = \{a, b\}, x = abab$ et $y = aab$. Alors $x \cdot y = ababaab$.

La structure $(\Sigma^*, \cdot)$ forme un monoïde :

- $\cdot$ est associative : $\forall x, y, z \in \Sigma^*, (x \cdot y) \cdot z = x \cdot (y \cdot z)$
- $|x \cdot y| = |x| + |y|$
- ε est l'élément identité de $(\Sigma^*, \cdot)$: $\forall x \in \Sigma^*, \varepsilon \cdot x = x \cdot \varepsilon = x$

Remarque

Dans la suite, on notera plus simplement l'opération de concaténation xy issue des mots x et y.

Opérations sur les langages

Un langage $A \subset \Sigma^*$ est défini comme un ensemble fini de mots.
Soient deux langages A et B, on note :

- $A \cup B = \{x | x \in A \vee x \in B\}$
- $A \cap B = \{x | x \in A \wedge x \in B\}$
- $\neg A = \{x \in \Sigma^* | x \notin A\}$
- $AB = \{xy | x \in A \wedge y \in B\}$
- A^n, pour $n \geq 0$ est défini inductivement :
  - $A^0 = \{\varepsilon\}$
  - $A^{n+1} = AA^n$
- $A^* = \bigcup_{n \geq 0} A^n$
  $= A^0 \cup A^1 \cup A^2 \cup A^3 \cup \ldots$
- $A^+ = AA^* = A^* - \{\varepsilon\}$

Exemples

$A = \{a, ab\}$ et $B = \{b, ba\}$:

- $AB = \{a, ab\}\{b, ba\}$
  $= \{ab, aba, abb, abba\}$
- $A^2 = \{aa, aab, aba, abab\}$

Table of Contents

Automate fini

- Un *Automate à états finis* est une structure abstraite composée d'états et de transitions permettant de représenter l'intégralité d'un système.

Automate fini

- Un *Automate à états finis* est une structure abstraite composée d'états et de transitions permettant de représenter l'intégralité d'un système.
- Communément, les automates servent à contrôler l'acceptabilité d'un langage / mot entré par l'utilisateur ou le système.

Automate fini

- Un *Automate à états finis* est une structure abstraite composée d'états et de transitions permettant de représenter l'intégralité d'un système.
- Communément, les automates servent à contrôler l'acceptabilité d'un langage / mot entré par l'utilisateur ou le système.
- Quelques exemples d'utilisation des automates :
  - Implémentation de circuits électroniques ;
  - Création et contrôle de langage informatique ;
  - Recherche de chaînes de caractères et motifs dans un texte ;
  - Systèmes automatiques : machines à café, ascenseurs, distributeurs automatique de billets.

Automate fini déterministe

Définition

Un automate fini déterministe (AFD) est un 5-uplet $M = (Q, \Sigma, s, F, \delta)$ tel que :

- Q est un ensemble fini d'états.
- Σ un alphabet de symboles.
- $s \in Q$, l'état initial de M.
- $F \subset Q$, l'ensemble des états finaux de M.
- $\delta : Q \times \Sigma \to Q$ est une fonction de transition qui, étant dans un état $q \in Q$ et observant un symbole $a \in \Sigma$ en entrée, indique le futur état de M.

Automate fini déterministe

Exemples

Soit l'automate M décrit de la façon suivante :

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- $s = q_1$
- $F = \{q_4\}$
- δ est définie telle que :

$$\begin{aligned}
&\delta(q_1, a) = q_2\\
&\delta(q_2, a) = q_3\\
&\delta(q_3, a) = \delta(q_4, a) = q_4\\
&\delta(q, b) = q, \quad \forall q \in Q
\end{aligned}$$

Automate fini déterministe

Exemples

Soit l'automate M décrit de la façon suivante :

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- $s = q_1$
- $F = \{q_4\}$
- δ est définie telle que :

$$\begin{aligned} &\delta(q_1, a) = q_2 \\ &\delta(q_2, a) = q_3 \\ &\delta(q_3, a) = \delta(q_4, a) = q_4 \\ &\delta(q, b) = q, \quad \forall q \in Q \end{aligned}$$

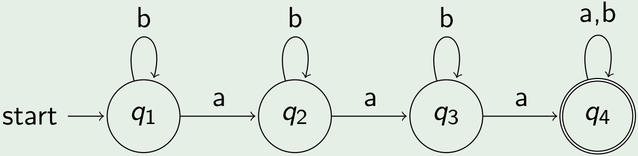
Par une table d'états :

	a	b
$\rightarrow q_1$	q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_3
$*q_4$	q_4	q_4

Automate fini déterministe

Exemples

Par un graphe d'états :



Question : Que fait cet automate ?

- Sur le mot *ababb*.
- Sur le mot *babaaa*.

Automate fini déterministe

Nous étendons la fonction de transition δ de la façon suivante :

$$\tilde{\delta} : Q \times \Sigma^* \to Q$$

La fonction $\tilde{\delta}$ est définie par récursion telle que :

$$\tilde{\delta}(q, \varepsilon) = q$$
$$\tilde{\delta}(q, ax) = \tilde{\delta}(\delta(q, a), x), \quad a \in \Sigma, x \in \Sigma^*$$

Automate fini déterministe

Nous étendons la fonction de transition δ de la façon suivante :

$$\tilde{\delta} : Q \times \Sigma^* \to Q$$

La fonction $\tilde{\delta}$ est définie par récursion telle que :

$$\tilde{\delta}(q, \varepsilon) = q$$
$$\tilde{\delta}(q, ax) = \tilde{\delta}(\delta(q, a), x), \quad a \in \Sigma, x \in \Sigma^*$$

Langage d'un automate

Le *langage* de M est défini comme l'ensemble des mots acceptés par cet automate :

$$L(M) = \{x | x \in \Sigma^*, \tilde{\delta}(s, x) \in F\}$$

Automate fini déterministe

Exemples

On considère le langage L suivant :

$L = \{xaaay \mid x, y \in \{a, b\}^*\}$
$\quad = \{x \in \{a, b\}^* \mid x \text{ contient au moins trois } a \text{ consécutifs}\}$

Déterminer un automate reconnaissant L. Tester que les mots $aabbaaab \in L$ et $ababa \notin L$.

Table of Contents

Non-déterminisme

- Le concept de *Non-déterminisme* est très important en informatique. Ceci réfère à des situations où l'état futur ne dépend plus d'un état unique antérieur mais d'un ensemble d'états.

Non-déterminisme

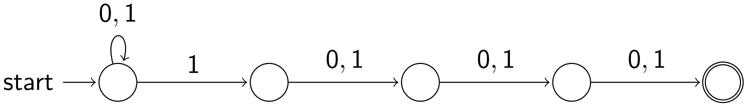
- Le concept de *Non-déterminisme* est très important en informatique. Ceci réfère à des situations où l'état futur ne dépend plus d'un état unique antérieur mais d'un ensemble d'états.
- Le non-déterminisme est courant dans de nombreux systèmes, notamment dans les situations où de nombreuses forces peuvent affecter le cours d'exécution. Par exemple, le comportement d'un processus au sein d'un système distribué.

Non-déterminisme

- Le concept de *Non-déterminisme* est très important en informatique. Ceci réfère à des situations où l'état futur ne dépend plus d'un état unique antérieur mais d'un ensemble d'états.
- Le non-déterminisme est courant dans de nombreux systèmes, notamment dans les situations où de nombreuses forces peuvent affecter le cours d'exécution. Par exemple, le comportement d'un processus au sein d'un système distribué.
- Le non-déterminisme est également un concept primordial dans la conception d'algorithmes efficaces. De nombreux problèmes combinatoires trouvent une solution simples avec des algorithmes non-déterministes et pour lesquels on ne connaît pas d'équivalent déterministe efficace et rapide. C'est le fameux problème $P = NP$.

Automate fini non-déterminisme

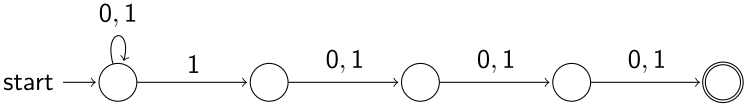
- Un automate fini non-déterministe est un automate où la fonction de transition peut mener à plusieurs états en pour un même symbole d'entrée.



- Intuitivement, l'exécution d'un mot x en entrée ne constitue plus une suite d'états, mais un arbre de calcul où les noeuds appartiennent à Q et les branches à Σ.

Automate fini non-déterminisme

- Un automate fini non-déterministe est un automate où la fonction de transition peut mener à plusieurs états en pour un même symbole d'entrée.



- Intuitivement, l'exécution d'un mot x en entrée ne constitue plus une suite d'états, mais un arbre de calcul où les noeuds appartiennent à Q et les branches à Σ.

Question : Dans le pire des cas, quelle est la taille de l'arbre généré pour un mot en entrée de taille $|x|$ et un automate à $|Q|$ états ?

Automate fini non-déterministe

Définition

Un *automate fini non-déterministe* (AFND) est un 5-uplet $N = (Q, \Sigma, S, F, \Delta)$ où Q, Σ et F ont la même sémantique que pour les AFD et :

- $S \subseteq Q$, l'ensemble des états initiaux.
- $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$, la fonction de transition définie telle que :

$$\Delta(P, a) = \bigcup_{q \in P} \delta(q, a)$$

Automate fini non-déterministe

Nous étendons la fonction de transition Δ de la façon suivante :

$$\tilde{\Delta} : 2^Q \times \Sigma^* \to 2^Q$$

La fonction $\tilde{\Delta}$ est définie par récursion telle que :

$$\tilde{\Delta}(P, \varepsilon) = P$$
$$\tilde{\Delta}(P, ax) = \tilde{\Delta}(\Delta(P, a), x)$$

$$p \overset{x}{\dashrightarrow} q \xrightarrow{a} r$$

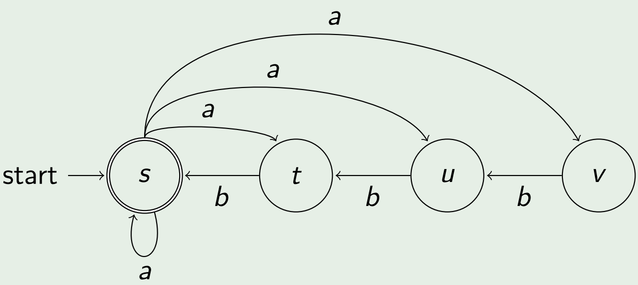
Le *langage* de N est défini tel que :

$$L(N) = \{x | x \in \Sigma^*, \tilde{\Delta}(S, x) \cap F \neq \emptyset\}$$

Automate fini non-déterministe

Exemples

On considère l'automate N suivant :

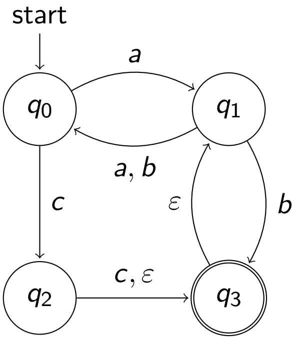


1. Dresser la table de transition Δ de N.
2. Déterminer un mot x commençant par a et tel que $x \notin L(N)$. Donner le calcul de x.

Automate fini non-déterministe avec ε-transition

Une extension des AFND est le concept de ε-transition.

Une ε-transition de l'état p à q, notée $\delta(p, \varepsilon) = q$, permet de passer d'un état p à un état q sans consommer de symbole du mot en entrée.



Soit l'automate M_ε ci-contre, le mot $cb \in L(M_\varepsilon)$.

Automate fini non-déterministe avec ε-transition

Clôture-ε

Soit un ANFD à ε-transition $M = (Q, \Sigma \cup \{\varepsilon\}, \Delta, S, F)$.

La *clôture-ε* d'un ensemble $A \subseteq Q$, notée $C_\varepsilon(A)$, est définie comme l'ensemble des états accessibles depuis tout état $q \in A$ en utilisant uniquement des ε-transitions.

$$C_\varepsilon(A) = \bigcup_{x \in \{\varepsilon\}^*} \tilde{\Delta}(A, x)$$

Exemples

Sur l'automate M_ε précédent, on a $C_\varepsilon(\{q_2\}) = \{q_3, q_2, q_1\}$.

Automate fini non-déterministe avec ε-transition

La fonction de transition Δ des AFND à ε-transitions est modifiée telle que :

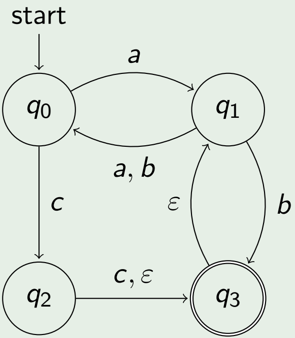
$$\Delta(P, a) = \bigcup_{q \in C_\varepsilon(P)} C_\varepsilon(\delta(q, a))$$

La fonction $\tilde{\Delta}$ est redéfinie telle que :

$$\tilde{\Delta}(P, \varepsilon) = C_\varepsilon(P)$$
$$\tilde{\Delta}(P, ax) = \tilde{\Delta}(\Delta(P, a), x)$$

Exemples

Soit l'automate M_ε défini précédemment :



On donne la table de transition telle que :

	a	b	c	C_ε
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$	$\{q_2\}$	$\{q_0\}$
q_1	$\{q_0\}$	$\{q_0, q_3\}$	$\emptyset$	$\{q_1\}$
q_2	$\emptyset$	$\emptyset$	$\{q_3\}$	$\{q_1, q_2, q_3\}$
$* \ q_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_1, q_3\}$

Automate fini non-déterministe avec ε-transition

Exemples

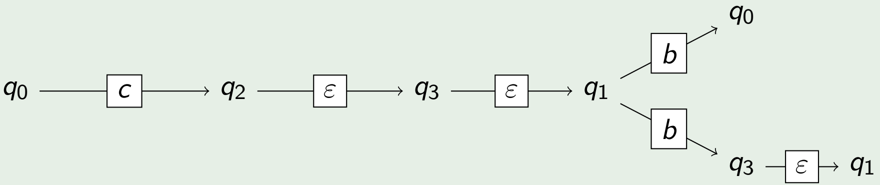
On donne l'exécution du mot cb pour M_ε

$$
\begin{aligned}
\tilde{\Delta}(\{q_0\}, cb) &= \tilde{\Delta}(\Delta(\{q_0\}, c), b) \\
&= \tilde{\Delta}(\delta(q_0, c), b) \\
&= \tilde{\Delta}(\{q_2\}, b) \\
&= \tilde{\Delta}(\Delta(\{q_2\}, b), \varepsilon) \\
&= \tilde{\Delta}(\delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_1, b), \varepsilon) \\
&= \tilde{\Delta}(\{q_0, q_3\}, \varepsilon) \\
&= \{q_0, q_1, q_3\}
\end{aligned}
$$

Automate fini non-déterministe avec ε-transition

Exemples

L'exécution du mot peut également être représentée par un arbre de calcul comme suit :



Automate fini non-déterministe avec ε-transition

Exemples

Déterminer un automate fini qui accepte chacun des langages suivants :

1. L'ensemble des mots sur $\{a, b\}$ contenant au moins trois occurrences du motifs *bbb*. Les chevauchements sont autorisés (ex. *bbbbb*).
2. L'ensemble des mots sur $\{0, 1\}$ dont la représentation en binaire est divisible par 4.
3. L'ensemble des mots sur $\{a\}$ dont la longueur est un multiple de 2 ou de 3.

Table of Contents

Équivalence d'automates

Théorème

Soit N, un automate fini quelconque. Il existe M, un AFD équivalent, soit : $L(N) = L(M)$

Attention

Soit un automate fini à $|Q|$ états. La complexité de l'algorithme de conversion est exponentielle en $O(2^{|Q|})$.

Algorithme de construction par sous-ensembles

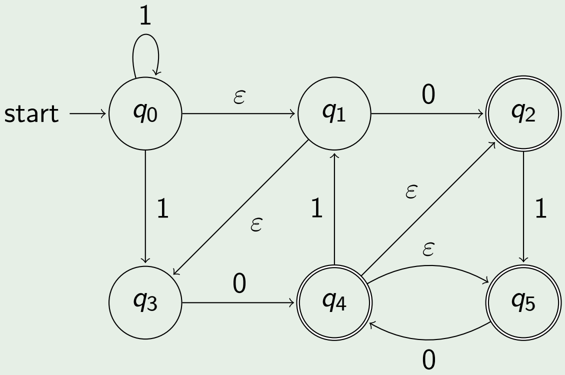
Data: $N = (Q, \Sigma, \Delta, S, F)$, Automate à convertir
Result: $M = (Q_{\text{det}}, \Sigma, \delta_{\text{det}}, s_{\text{det}}, F_{\text{det}})$ Version déterministe de N

$s_{\text{det}} \leftarrow C_\varepsilon(S)$;
$\Pi \leftarrow \emptyset$; ▷ `File représentant l'ensemble des parties`
$\Pi.enQueue(s_{\text{det}})$;
$Q_{\text{det}} \leftarrow s_{\text{det}}$;
$F_{\text{det}} \leftarrow \emptyset$;
do
  $\pi \leftarrow \Pi.deQueue()$; ▷ `On défile π`
  if $\pi \cap F \neq \emptyset$ **then**
    $F_{\text{det}} \leftarrow F_{\text{det}} \cup \pi$;
  end
  for $a \in \Sigma$ **do**
    $q \leftarrow C_\varepsilon(\Delta(\pi, a))$;
    if $q \notin Q_{det}$ **then**
      $\Pi.enQueue(q)$;
      $Q_{\text{det}} \leftarrow Q_{\text{det}} \cup q$;
    end
    $\delta_{\text{det}}(\pi, a) \leftarrow q$;
  end
while $\Pi \neq \emptyset$;

Conversion d'automate

Exemples

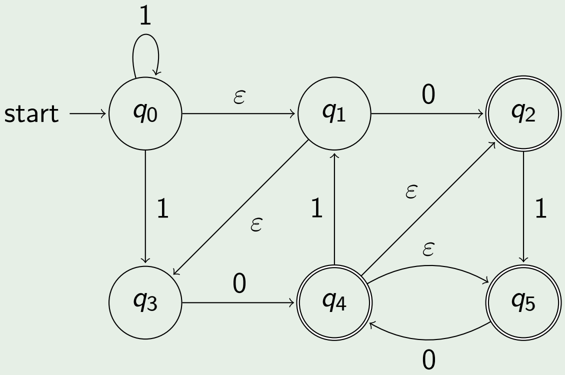
1. Donner la table de transition de l'automate N ci-dessous.



Conversion d'automate

Exemples

1. Donner la table de transition de l'automate N ci-dessous.



2. Convertir N en AFD.

Exemples

	0	1	C_ε
$\rightarrow q_0$	$\emptyset$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
q_1	$\{q_2\}$	$\emptyset$	$\{q_1, q_3\}$
$*q_2$	$\emptyset$	$\{q_5\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$	$\{q_3\}$
$*q_4$	$\emptyset$	$\{q_1\}$	$\{q_2, q_4, q_5\}$
$*q_5$	$\{q_4\}$	$\emptyset$	$\{q_5\}$

On construit l'ensemble des parties accessibles de l'automate.

π	$\rho(\pi)$	0	1
$\rightarrow \{q_0, q_1, q_3\}$	1	2	1

Conversion d'automate

Exemples

	0	1	C_ε
$\rightarrow q_0$	$\emptyset$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
q_1	$\{q_2\}$	$\emptyset$	$\{q_1, q_3\}$
$*q_2$	$\emptyset$	$\{q_5\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$	$\{q_3\}$
$*q_4$	$\emptyset$	$\{q_1\}$	$\{q_2, q_4, q_5\}$
$*q_5$	$\{q_4\}$	$\emptyset$	$\{q_5\}$

On construit l'ensemble des parties accessibles de l'automate.

π	$\rho(\pi)$	0	1
$\rightarrow \{q_0, q_1, q_3\}$	1	2	1
$*\{q_2, q_4, q_5\}$	2	2	3

Conversion d'automate

Exemples

	0	1	C_ε
$\rightarrow q_0$	$\emptyset$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
q_1	$\{q_2\}$	$\emptyset$	$\{q_1, q_3\}$
$*q_2$	$\emptyset$	$\{q_5\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$	$\{q_3\}$
$*q_4$	$\emptyset$	$\{q_1\}$	$\{q_2, q_4, q_5\}$
$*q_5$	$\{q_4\}$	$\emptyset$	$\{q_5\}$

On construit l'ensemble des parties accessibles de l'automate.

π	$\rho(\pi)$	0	1
$\rightarrow \{q_0, q_1, q_3\}$	1	2	1
$*\{q_2, q_4, q_5\}$	2	2	3
$*\{q_5, q_1, q_3\}$	3	2	$\emptyset$

Conversion d'automate

Exemples

	0	1	C_ε
$\rightarrow q_0$	$\emptyset$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
q_1	$\{q_2\}$	$\emptyset$	$\{q_1, q_3\}$
$*q_2$	$\emptyset$	$\{q_5\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$	$\{q_3\}$
$*q_4$	$\emptyset$	$\{q_1\}$	$\{q_2, q_4, q_5\}$
$*q_5$	$\{q_4\}$	$\emptyset$	$\{q_5\}$

On construit l'ensemble des parties accessibles de l'automate.

π	$\rho(\pi)$	0	1
$\rightarrow \{q_0, q_1, q_3\}$	1	2	1
$*\{q_2, q_4, q_5\}$	2	2	3
$*\{q_5, q_1, q_3\}$	3	2	$\emptyset$

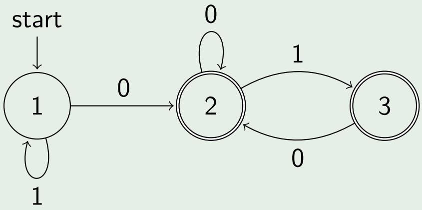


Table of Contents

Langages et automates

- On a découvert jusqu'ici plusieurs formes d'automates et montré qu'elles sont équivalentes.
- Deux questions :
  1. Comment savoir, pour un langage donné, que je peux construire un automate fini pour le représenter ?

Langages et automates

- On a découvert jusqu'ici plusieurs formes d'automates et montré qu'elles sont équivalentes.
- Deux questions :
  1. Comment savoir, pour un langage donné, que je peux construire un automate fini pour le représenter ?
  2. Comment savoir si deux automates sont équivalents ?

Langages et automates

- On a découvert jusqu'ici plusieurs formes d'automates et montré qu'elles sont équivalentes.
- Deux questions :
  1. Comment savoir, pour un langage donné, que je peux construire un automate fini pour le représenter ?
  2. Comment savoir si deux automates sont équivalents ?

Une réponse peut être apportée à l'aide de la notion de *classe d'équivalence*.

Relation d'équivalence

Définition

Soit un ensemble E, une relation d'équivalence $\equiv: E \times E \to \{Vrai, Faux\}$, est une relation obéissant aux axiomes suivants :

1. Reflexivité : $\forall x \in E, x \equiv x$
2. Symétrie : $\forall x, y \in E, x \equiv y \Leftrightarrow y \equiv x$
3. Transitive : $\forall x, y, z \in E, x \equiv y \wedge y \equiv z \Rightarrow x \equiv z$

Classe d'équivalence

La *classe d'équivalence* d'un élément $x \in E$, notée $[x]$ est constituée de tous les éléments qui lui sont équivalents :

$$[x] = \{y | y \equiv x\}$$

Deux propriétés :

- $x \equiv y \Leftrightarrow [x] = [y]$.
- L'ensemble des classes d'équivalence forment une partition de E.

Relation d'équivalence

Exemples

Soit un AFD $M = (Q, \Sigma, s, F, \delta)$ dont tous les états sont accessibles. On définit la relation $\equiv_M$ sur Σ^* telle que :

$$x \equiv_M y \Leftrightarrow \tilde{\delta}(s, x) = \tilde{\delta}(s, y)$$

Relation d'équivalence

Exemples

Soit un AFD $M = (Q, \Sigma, s, F, \delta)$ dont tous les états sont accessibles. On définit la relation $\equiv_M$ sur Σ^* telle que :

$$x \equiv_M y \Leftrightarrow \tilde{\delta}(s, x) = \tilde{\delta}(s, y)$$

1. Qualifier en langage naturel la relation d'équivalence $\equiv_M$.
2. Montrer que $\equiv_M$ est une relation d'équivalence.

Relation de Myhill-Nerode

La relation définie précédemment satisfait d'autres propriétés intéressantes.

Propriétés

1. $\equiv_M$ est une congruente à droite : $\forall x, y \in \Sigma^*, \forall a \in \Sigma, x \equiv y \Rightarrow xa \equiv ya$
2. $\equiv_M$ affine $L(M)$: $\forall x, y \in \Sigma^*, x \equiv_M y \Rightarrow (x \in L(M) \Leftrightarrow y \in L(M))$
3. $\equiv_M$ est d'index fini, c'est-à-dire qu'il existe un nombre fini de classes d'équivalence.

Une relation qui respecte les trois propriétés ci-dessus est appelée *relation de Myhill-Nerode*.

Attention

- Les propriété 1 et 2 découlent de la définition donnée précédemment.
- La propriété 3 découle du fait que les automates ont un nombre *fini* d'états.

Si la propriété 3 n'est pas respectée, alors le langage ne peut être représenté par un automate à états finis.

Langage régulier

Théorème de Myhill-Nerode

Soit un langage $A \subset \Sigma^*$. Les trois propositions suivantes sont équivalentes :

1. A est régulier.
2. Il est existe un automate fini M tel que $A = L(M)$.
3. Il existe une relation de Myhill-Nerode sur A.

On note **Reg**, l'ensemble des langages réguliers.

Exemples

Soit le langage $L = \{0^n 1^n | n \geq 0\}$. A-t-on $L \in \mathbf{Reg}$?

Langage régulier

Théorème de Myhill-Nerode

Soit un langage $A \subset \Sigma^*$. Les trois propositions suivantes sont équivalentes :

1. A est régulier.
2. Il est existe un automate fini M tel que $A = L(M)$.
3. Il existe une relation de Myhill-Nerode sur A.

On note **Reg**, l'ensemble des langages réguliers.

Exemples

Soit le langage $L = \{0^n 1^n | n \geq 0\}$. A-t-on $L \in$ **Reg** ? Non ...

Démonstration :
Soit $k \neq n$. Supposons que $a^k \equiv_L a^n$. Par congruence à droite, on a $a^k b \equiv_L a^n b$. Par récurrence, il vient que $a^k b^n \equiv_L a^n b^n$. C'est impossible car $a^k b^n \notin L$ et $a^n b^n \in L$. L'hypothèse de départ est absurde. On en conclut que $\equiv_L$ n'est pas d'index fini, ce n'est pas une relation de Myhill-Nerode, L n'est pas régulier. □

Automate fini déterministe

Théorème

Reg est clôt pour les opérations $\cap, \neg, \cup, \cdot$ et $*$.

Démonstration.

Soient $L, L' \in$ **Reg**.

1. Preuve pour $L \cdot L'$: Voir TD1
2. Preuve pour L^* : Voir CM2
3. Preuve pour $\neg L$: À vous de jouer...
4. Preuve pour $L \cap L'$: Effectuée en cours.

Automate fini déterministe

Théorème

Reg est clôt pour les opérations $\cap, \neg, \cup, \cdot$ et $*$.

Démonstration.

Soient $L, L' \in$ **Reg**.

1. Preuve pour $L \cdot L'$: Voir TD1
2. Preuve pour L^* : Voir CM2
3. Preuve pour $\neg L$: À vous de jouer...
4. Preuve pour $L \cap L'$: Effectuée en cours.
5. Preuve pour $L \cup L'$: On sait par les lois de de Morgan que $L_1 \cup L_2 = \neg(\neg L_1 \cap \neg L_2)$.
   On sait que $\forall L \in$ **Reg**, alors $\neg L \in$ **Reg**.
   On sait également que $\forall L, L' \in$ **Reg**, alors $L \cap L' \in$ **Reg**.
   Dès lors, $\neg(\neg L_1 \cap \neg L_2) = L_1 \cup L_2 \in$ **Reg**.

□

Table of Contents

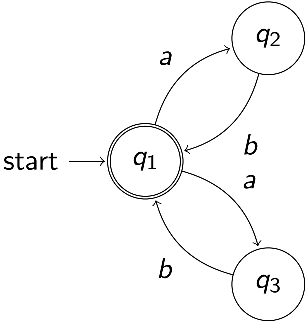
Minimisation d'automate

- Il existe une infinité d'automates reconnaissant le même langage.

Minimisation d'automate

- Il existe une infinité d'automates reconnaissant le même langage.
- Le problème de la minimisation d'un automate consiste, étant donné un AFD M, en la recherche de l'AFD équivalent M_{min}, unique et ayant le nombre minimal d'états.

- Il existe une infinité d'automates reconnaissant le même langage.
- Le problème de la minimisation d'un automate consiste, étant donné un AFD M, en la recherche de l'AFD équivalent M_{min}, unique et ayant le nombre minimal d'états.
- Plusieurs algorithmes et approches existent :
    - Recherche d'états équivalents (Algorithmes de Moore, Algorithme de Hopcroft)
    - Par détermination (Algorithme de Brzozowski)



Algorithme de Moore

Équivalence de Moore d'ordre k

Soient un AFD $M = (Q, \Sigma, s, F, \delta)$ et deux états $p, q \in Q$. Les états p et q sont Moore-équivalents d'ordre k si et seulement si :

$$p \sim_k q \Leftrightarrow L_q^{(k)}(M) = L_p^{(k)}(M)$$

où $L_q^{(k)}(M) = \{x \in \Sigma^* | \delta(q, x) \in F \wedge |x| \leq k\}$
L'ensemble $L_q^{(k)}$ est composé des mots de longueur au plus k qui mènent de l'état q à un état final de M.

Deux états sont indistinguables pour la relation $\sim_k$ si les mêmes mots de longueur au plus k mènent à des états finaux.

Équivalence de Moore d'ordre k

Soient un AFD $M = (Q, \Sigma, s, F, \delta)$ et deux états $p, q \in Q$. Les états p et q sont Moore-équivalents d'ordre k si et seulement si :

$$p \sim_k q \Leftrightarrow L_q^{(k)}(M) = L_p^{(k)}(M)$$

où $L_q^{(k)}(M) = \{x \in \Sigma^* | \delta(q, x) \in F \wedge |x| \leq k\}$
L'ensemble $L_q^{(k)}$ est composé des mots de longueur au plus k qui mènent de l'état q à un état final de M.

Deux états sont indistinguables pour la relation $\sim_k$ si les mêmes mots de longueur au plus k mènent à des états finaux.

Question : Soit un automate $M = (Q, \Sigma, s, F, \delta)$ quelconque. Donner les classes d'équivalence de Q pour $\sim_0$.

Algorithme de Moore

La relation $\sim_k$ peut être calculée par récurrence.

Propriété

Soient deux états $p, q \in Q$, et $k \geq 0$:

$$p \sim_{k+1} q \Leftrightarrow p \sim_k q \wedge \forall a \in \Sigma, \delta(p, a) \sim_k \delta(q, a)$$

Ainsi, pour calculer l'équivalence $\sim_{k+1}$, on détermine les états qui, par un symbole, arrivent dans des classes distinctes de $\sim_k$.

Corollaire

Si $p \sim_k q = p \sim_{k+1} q$, alors $\forall r \geq 0, p \sim_k q = p \sim_{k+r} q$

Algorithme de Moore

Data: $M = (Q, \Sigma, \delta, s, F)$ Automate à convertir,
$\mathcal{P}$ Partition des états

```
P ← (Q − F, F) ; ▷ Équivalence initiale ~₀
Q ← P ; ▷ Ancienne version de P
do
    for a ∈ Σ do
        P_a ← a⁻¹P ; ▷ Action de la lettre a
    end
    P ← P ∩ ⋂_{a∈Σ} P_a ; ▷ Mise à jour de la partition
while P ≠ Q;
```

Remarques

- $a^{-1}\mathcal{P}$ est l'équivalence d'états (i.e. partition) définie telle que :

$$p \equiv q \mod (a^{-1}\mathcal{P}) \Leftrightarrow \delta(p, a) \equiv \delta(q, a) \mod (\mathcal{P})$$

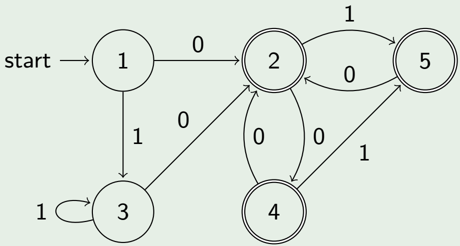
- $\mathcal{P}$ représente la relation $\sim_k$ et $\mathcal{P} \cap \bigcap_{a\in\Sigma} \mathcal{P}_a$ la relation $\sim_{k+1}$.
- La complexité moyenne de l'algorithme de Moore est en $O(n \log(n))$ avec n le nombre d'états de l'automate M.

Algorithme de Moore

Exemples

On reprend l'automate issu de la section précédente.

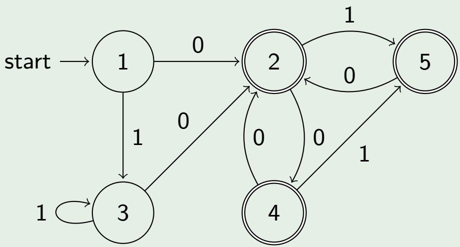
	0	1	$\sim_0$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_1$
$\rightarrow 1$	2	3	$\bullet$			
$*2$	4	5	$\bullet$			
3	2	3	$\bullet$			
$*4$	2	5	$\bullet$			
$*5$	2	$\emptyset$	$\bullet$			
$\emptyset$	$\emptyset$	$\emptyset$	$\bullet$			



Exemples

Soit l'automate ci-dessous.

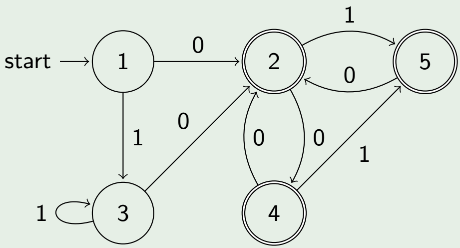
	0	1	$\sim_0$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_1$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_2$
$\rightarrow 1$	2	3	$\bullet$	$\color{red}\bullet$	$\bullet$	$\color{blue}\bullet$			
$*2$	4	5	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$			
3	2	3	$\bullet$	$\color{red}\bullet$	$\bullet$	$\color{blue}\bullet$			
$*4$	2	5	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$			
$*5$	2	$\emptyset$	$\color{red}\bullet$	$\color{red}\bullet$	$\bullet$	$\color{purple}\bullet$			
$\emptyset$	$\emptyset$	$\emptyset$	$\bullet$	$\bullet$	$\bullet$	$\bullet$			



Exemples

On reprend l'automate issu de la section précédente.

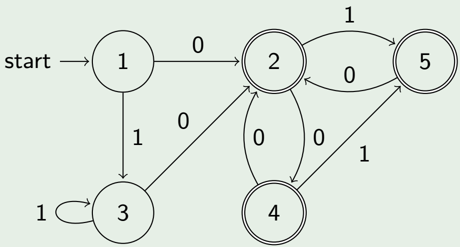
	0	1	$\sim_0$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_1$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_2$
$\rightarrow 1$	2	3	$\textcolor{black}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{blue}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{blue}{\bullet}$	$\textcolor{teal}{\bullet}$
$*2$	4	5	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{purple}{\bullet}$	$\textcolor{magenta}{\bullet}$
3	2	3	$\textcolor{black}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{blue}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{blue}{\bullet}$	$\textcolor{teal}{\bullet}$
$*4$	2	5	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{purple}{\bullet}$	$\textcolor{magenta}{\bullet}$
$*5$	2	$\emptyset$	$\textcolor{red}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{purple}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{purple!50!red}{\bullet}$
$\emptyset$	$\emptyset$	$\emptyset$	$\textcolor{black}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{black}{\bullet}$	$\textcolor{black}{\bullet}$

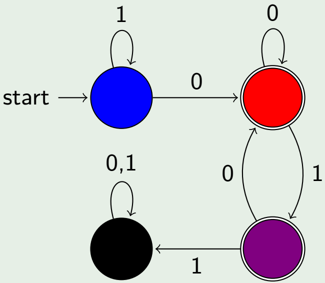


Exemples

On reprend l'automate issu de la section précédente.

	0	1	$\sim_0$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_1$	$0\mathcal{P}$	$1\mathcal{P}$	$\sim_2$
$\rightarrow 1$	2	3	$\color{black}\bullet$	$\color{red}\bullet$	$\color{black}\bullet$	$\color{blue}\bullet$	$\color{red}\bullet$	$\color{blue}\bullet$	$\color{teal}\bullet$
$*2$	4	5	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{purple}\bullet$	$\color{magenta}\bullet$
3	2	3	$\color{black}\bullet$	$\color{red}\bullet$	$\color{black}\bullet$	$\color{blue}\bullet$	$\color{red}\bullet$	$\color{blue}\bullet$	$\color{teal}\bullet$
$*4$	2	5	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{purple}\bullet$	$\color{magenta}\bullet$
$*5$	2	$\emptyset$	$\color{red}\bullet$	$\color{red}\bullet$	$\color{black}\bullet$	$\color{purple}\bullet$	$\color{red}\bullet$	$\color{black}\bullet$	$\color{crimson}\bullet$
$\emptyset$	$\emptyset$	$\emptyset$	$\color{black}\bullet$	$\color{black}\bullet$	$\color{black}\bullet$	$\color{black}\bullet$	$\color{black}\bullet$	$\color{black}\bullet$	$\color{black}\bullet$





Algorithme de Brzozowski

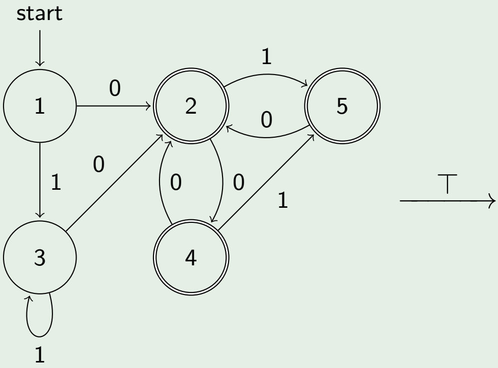
L'algorithme de Brzozowski propose une méthode de minimisation d'un automate basée sur les notions de détermination et transposition.

Automate transposé

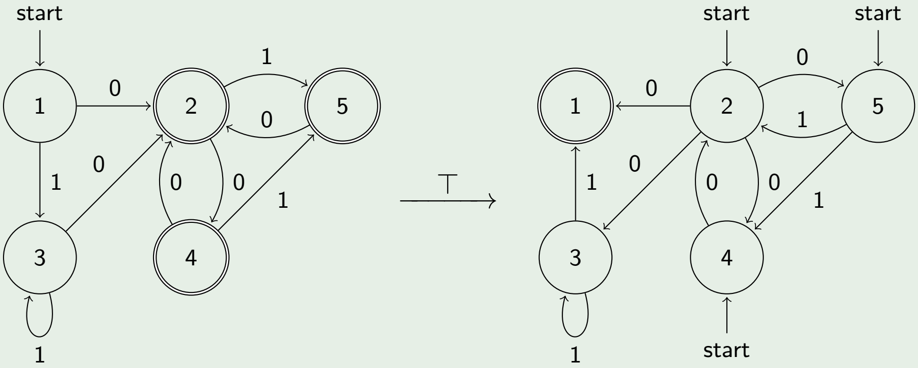
Soit un automate fini $N = (Q, \Sigma, S, F, \Delta)$, l'*automate transposé* (i.e. automate miroir) $N^\top$ est obtenu en échangeant états finaux et initiaux et le sens de toutes les transitions.
Formellement, $N^\top = (Q, \Sigma, F, S, \Delta^\top)$ avec $\Delta^\top(P, a) = \bigcup_{p \in P} \{q | p \in \Delta(q, a)\}$.

Question : Quel est le langage reconnu par $N^\top$ *?*

Exemples



Exemples

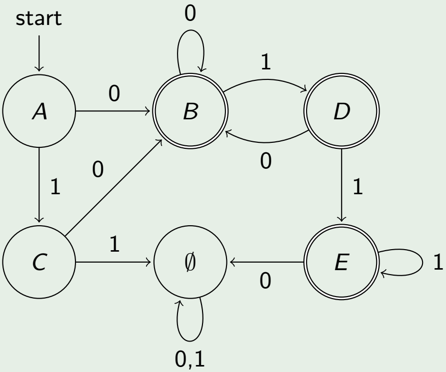


Algorithme de Brzozowski

Exemples

On détermine l'automate transposé obtenu :

π	$\rho(\pi)$	0	1
$\rightarrow \{2,4,5\}$	A	B	C
$*\{1,2,3,4,5\}$	B	B	D
$\{2,4\}$	C	B	$\emptyset$
$*\{1,2,3,4\}$	D	B	E
$*\{1,3\}$	E	$\emptyset$	E



Algorithme de Brzozowski

Théorème

Soit un automate fini N quelconque. La transformation de Brzozowski est définie telle que :

$$\mathcal{B}(N) = \det(\det(N^\top)^\top)$$

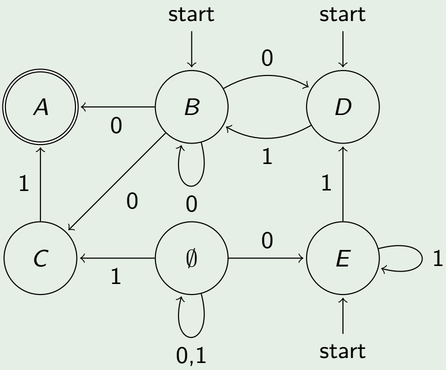
où det correspond à l'opération de transformation d'un automate en AFD équivalent.
Alors, $\mathcal{B}(N)$ est minimal et $L(\mathcal{B}(N)) = L(N)$.

Attention

L'algorithme de transformation possède une complexité exponentielle en $O(2^{|Q|})$.

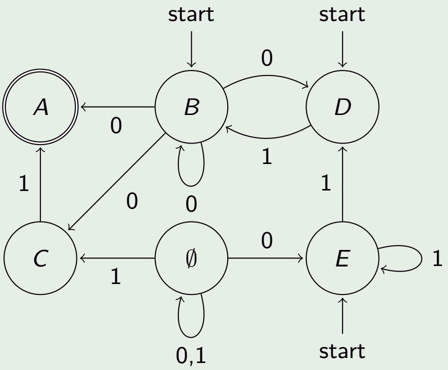
Algorithme de Brzozowski

Exemples



Algorithme de Brzozowski

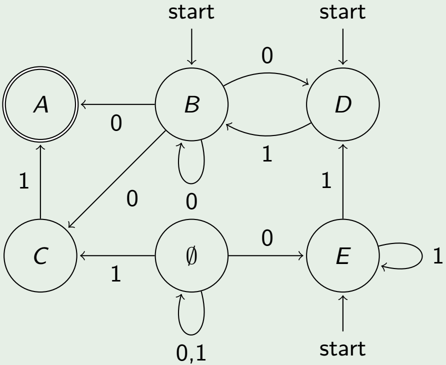
Exemples



π	$\rho(\pi)$	0	1
$\rightarrow \{B, D, E\}$	•	•	•
$*\{A, B, C, D\}$	•	•	•
$*\{A, B\}$	•	•	$\emptyset$

Algorithme de Brzozowski

Exemples



π	$\rho(\pi)$	0	1
$\rightarrow \{B, D, E\}$	$\color{blue}{\bullet}$	$\color{red}{\bullet}$	$\color{blue}{\bullet}$
$*\{A, B, C, D\}$	$\color{red}{\bullet}$	$\color{red}{\bullet}$	$\color{purple}{\bullet}$
$*\{A, B\}$	$\color{purple}{\bullet}$	$\color{red}{\bullet}$	$\emptyset$