Graphes et Langages

Chapitre 6
Automates à états finis

Leo Donati Noëlle Stolfi

Université de Nice Sophia Antipolis
IUT Nice Côte d'Azur
DUT Informatique

2015-2016

Chapitre 6 : Automates à états finis

Automates et mathématiques

Principe

Les automates sont des constructions mathématiques qui représentent des "machines à raisonner".
Conçues en grande partie lorsque les ordinateurs n'existaient pas.
Permettent de prouver théoriquement ce qu'un ordinateur réel peut ou ne peut pas faire.

- Automates à états finis
- Machines séquentielles
- Automates à pile
- Random Access Machine (RAM)
- Machines de Turing

Automates à états finis

Définition

Un automate à états finis M est défini par un quintuplet $M = (\Sigma, E, I, F, \delta)$ où

- Σ est un alphabet,
- E est un ensemble non vide qui représente les états de l'automate,
- $I \in E$ est un état particulier appelé état initial,
- $F \subset E$ est le sous-ensemble des états finaux
- $\delta : E \times \Sigma \to E$ est la fonction de transition qui explique le fonctionnement de l'automate.

Fonction de transition

Fonctionnement

Si q_1 est un état et a est une lettre de Σ, alors

$$\delta(q_1, a) = q_2$$

indique que si l'automate est dans l'état q_1 alors la lecture de la lettre a provoque une transition vers q_2.

Matrice de transition

On peut décrire la fonction δ par une matrice de transition :

- une ligne pour chaque état de E
- une colonne pour chaque lettre de l'alphabet Σ
- dans la case (q, a) on, place $\delta(q, a)$

Exemple

Exemple

$M = (\Sigma, E, I, F, \delta)$ avec

- $\Sigma = \{a, b, c\}$
- $E = \{1, q_1, q_2, q_3, q_4\}$
- $F = \{q_3\}$
- δ est donné par la matrice de transition

δ	a	b	c
I	q_1	I	I
q_1	q_2	q_1	q_1
q_2	q_3	q_2	q_2
q_3	q_4	q_3	q_3
q_4	q_4	q_4	q_4

Graphe étiqueté d'une automate

Représentation de l'automate

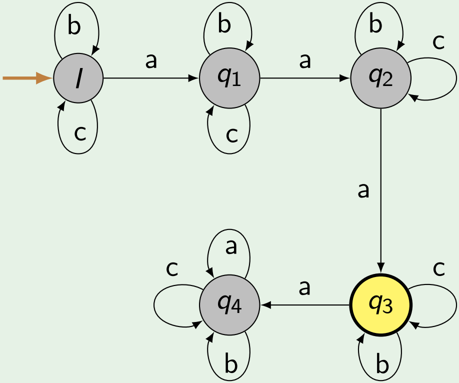
On construit un graphe orienté étiqueté :

- les sommets sont les états de E,
  - l'état initial I est signalé par une flèche
  - les états finaux F sont doublement entourés
- on place un arc entre le sommet q_1 et le sommet q_2 avec une étiquette a si et seulement si $\delta(q_1, a) = q_2$

Exemple - suite

Exemple

	a	b	c
I	q_1	I	I
q_1	q_2	q_1	q_1
q_2	q_3	q_2	q_2
q_3	q_4	q_3	q_3
q_4	q_4	q_4	q_4



Transitions associées à des mots

Généralisation

Si on a une succession de lettres $a_0, a_1, a_2, \ldots a_k$ et d'états $q_0, q_1, q_2, \ldots q_{k+1}$ tels que

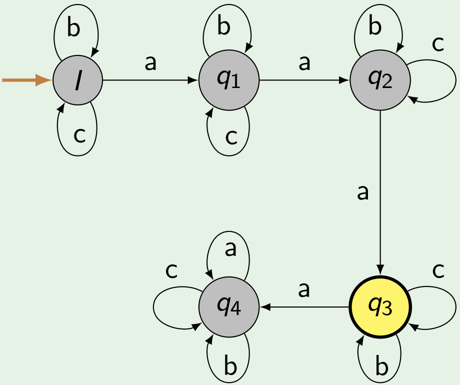
- $\delta(q_0, a_0) = q_1$ (a_0 provoque une transition de q_0 à q_1
- $\delta(q_1, a_1) = q_2$
- $\delta(q_2, a_2) = q_3$
- $\ldots$
- $\delta(q_k, a_k) = q_{k+1}$

alors on dit que le mot $M = a_0 a_1 a_2 \cdots a_k$ provoque une transition de l'état q_0 vers l'état q_{k+1}, et on note (par abus de notation)

$$\delta(q_0, M) = q_{k+1}$$

Exemple - suite

Exemple



- $\delta(I, abbac) = q_2$
- $\delta(q_1, abbac) = q_3$
- $\delta(q_2, abbac) = q_4$
- $\delta(q_3, abbac) = q_4$
- $\delta(q_4, abbac) = q_4$

Mots acceptés

Définition

On dit que le mot $W \in \Sigma^*$ est accepté par l'automate à états finis $M = (\Sigma, E, I, F, \delta)$ si et seulement si la lecture du mot W provoque une transition de l'état initial I vers un état final (ou acceptant). C'est à dire que

$$\delta(I, W) \in F$$

Langage accepté

Définition

- Le langage accepté (ou décidé) par un automate à états finis M est l'ensemble L de tous les mots sur l'alphabet Σ acceptés par l'automate.
- On dit aussi que l'automate M décide le langage L.

Exemple

Dans l'exemple précédent le langage décidé par l'automate est l'ensemble des mots sur $\{a, b, c\}$ qui ont exactement trois a.
Avec une expression régulière :

$$(b+c)^* \cdot a \cdot (b+c)^* \cdot a \cdot (b+c)^* \cdot a \cdot (b+c)^*$$

Langages rationnels

Définition

On dit qu'un langage L est rationnel (on parle aussi de langage automatique) s'il existe un automate à états finis M qui décide L.

Questions

- Quels sont les langages rationnels ?
- Quel est le lien entre les langages rationnels et les langages réguliers ?

Chapitre 6 : Automates à états finis

Singleton

Théorème

Tout langage composé d'un unique mot est un langage rationnel.

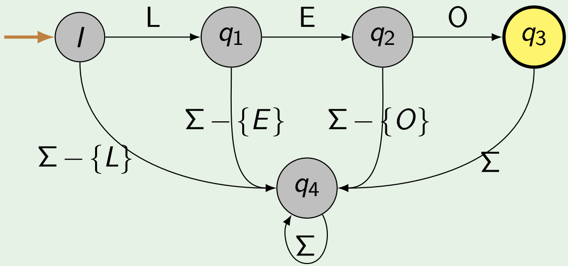
Exemple : $L = \{LEO\}$

Singleton

Théorème

Tout langage composé d'un unique mot est un langage rationnel.

Exemple : $L = \{LEO\}$



Pairs-Impairs

Exemple

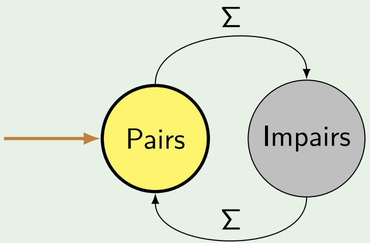
Quel que soit l'alphabet Σ, je considère le langage L formé par les mots de longueur paire.
Est-ce que L est rationnel ?

Pairs-Impairs

Exemple

Quel que soit l'alphabet Σ, je considère le langage L formé par les mots de longueur paire.
Est-ce que L est rationnel ?

Réponse



Complémentaire d'un langage

Théorème

Le complémentaire d'un langage rationnel est encore un langage rationnel.

Preuve

Si $M = (\Sigma, E, I, F, \delta)$ est un automate a états finis qui accepte le langage L alors

$$M' = (\Sigma, E, I, E \setminus F, \delta)$$

dont les états finaux sont les états non finaux de M et inversement, est un automate à états finis qui accepte $\overline{L}$.

Produit cartésien d'automates

Principe

Si $M = (\Sigma, E, I, F, \delta)$ et $M' = (\Sigma, E', I'; F', \delta')$ sont deux automates sur le même alphabet, alors le produit cartésien d'automates $M \times M'$ est un automate qui simule le fonctionnement simultané de M et de M' :

- son alphabet est Σ
- ses états sont $E \times E'$
- son état initial est le couple (I, I')
- et sa fonction de transition

$$\begin{array}{rll} \delta \times \delta' : & (E \times E') \times \Sigma & \rightarrow (E \times E') \\ & ((q, q'), a) & \mapsto (\delta(q, a), \delta(q', a)) \end{array}$$

Utilisation

Grâce au produit cartésien d'automates, en choisissant opportunément l'ensemble des états finaux, on va pouvoir construire des automates

- pour l'intersection de langages rationnels
- pour la somme de langages rationnels
- pour la différence de langages rationnels

Intersection

Théorème

L'intersection de deux langages rationnels est aussi un langage rationnel.

Preuve

Si $M = (\Sigma, E, I, F, \delta)$ accepte L et $M' = (\Sigma, E', I'; F', \delta')$ accepte L' alors on construit le produit cartésien $M \times M'$ avec comme états finaux le produit cartésien $F \times F'$.

Union

Théorème

L'union de deux langages rationnels est aussi un langage rationnel.

Preuve

Si $M = (\Sigma, E, I, F, \delta)$ accepte L et $M' = (\Sigma, E', I'; F', \delta')$ accepte L' alors on construit le produit cartésien $M \times M'$ avec comme états finaux les couples de

$$(F \times E') \cup (E \times F')$$

Différence

Théorème

La différence de deux langages rationnels est aussi un langage rationnel.

Preuve

Si $M = (\Sigma, E, I, F, \delta)$ accepte L et $M' = (\Sigma, E', I'; F', \delta')$ accepte L' alors on construit le produit cartésien $M \times M'$ avec comme états finaux les couples de $F \times (E' \setminus F')$.

Chapitre 6 : Automates à états finis

Déterminisme/Non déterminisme

Déterminisme

C'est quand l'évolution d'un système d'un état vers un autre est totalement déterminé par son état actuel et son interaction avec son environnement.

Non déterminisme

C'est quand on ne peut pas prévoir avec certitude l'état futur d'un système car pour un état donné il peut évoluer de plusieurs façons différentes avec le même input.

Automates non déterministes

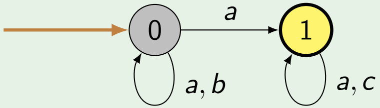
Définition

Un automate à états finis non déterministes (AFN) ressemble à un état fini déterministe (AFD) sauf pour sa fonction de transition δ : pour un couple donné $(q, a) \in E \times \Sigma$, $\delta(q, a)$ sera maintenant un ensemble (éventuellement vide) d'états $\{q_1, q_2, \ldots\}$.
Ce qui signifiera que la lecture de la lettre a peut provoquer une transition vers l'un de ces différents états, de façon non prévisible.
On a donc pour un AFN :

$$\delta : E \times \Sigma \to \mathcal{P}(E)$$

Exemple d'AFN

Exemple sur $\Sigma = \{a, b, c\}$



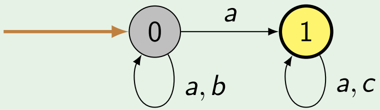
C'est un automate non déterministe car :

- $\delta(0, a) = \{0, 1\}$, deux transitions possibles,
- $\delta(0, c) = \emptyset$, aucune transition,
- $\delta(1, b) = \emptyset$.

Pourtant on voit bien quel est le langage accepté :

Exemple d'AFN

Exemple sur $\Sigma = \{a, b, c\}$



C'est un automate non déterministe car :

- $\delta(0, a) = \{0, 1\}$, deux transitions possibles,
- $\delta(0, c) = \emptyset$, aucune transition,
- $\delta(1, b) = \emptyset$.

Pourtant on voit bien quel est le langage accepté :

$$(a+b)^* a(a+c)^*$$

Langage associé à un AFN

Lecture d'un mot

Une lecture d'un mot $W = a_1 a_2 \ldots a_k \in \Sigma^*$ par un automate à états finis non déterministe est une suite d'états $q_0, q_1, q_2, \cdots q_k$ tels que :

- $q_0 = I$,
- $\delta(q_i, a_i)$ contient l'état q_{i+1}

Mots acceptés par un AFN

On dit qu'un mot $W \in \Sigma^*$ est accepté par un AFN, s'il existe une lecture de W qui se termine par un état acceptant.

Lien entre AFD et AFN

Théorème

Si L est un langage accepté par un automate fini non déterministe, alors on pour construire un automate fini déterministe qui accepte le même langage.

Conséquences

Ajouter le non déterminisme n'augmente pas la puissance des automates mais simplifie leur conception.

Transitions ε

Transitions instantanées

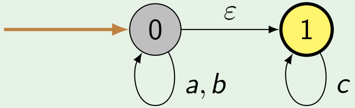
- Dans un automate non déterministe on autorise une même lettre à provoquer plusieurs transitions différentes.
- Une transition instantanée ou transition-ε, est un changement d'état de l'automate qui n'est provoqué par aucune lecture de lettre.

Définition

Un automate fini non déterministe à transitions ε (AFN-ε) ressemble à un AFN sauf pour sa fonction de transition δ qui est cette fois définie sur $E \times (\Sigma \cup \{\varepsilon\})$

Exemple de AFN-ε

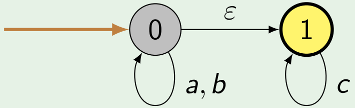
Exemple sur $\Sigma = \{a, b, c\}$



L'automate peut passer, tout seul de l'état 0 à l'état 1.
Le langage défini par cet automate est :

Exemple de AFN-ε

Exemple sur $\Sigma = \{a, b, c\}$



L'automate peut passer, tout seul de l'état 0 à l'état 1.
Le langage défini par cet automate est :

$$(a+b)^*c^*$$

Mot accepté par un AFN-ε

Définition

Un mot W est accepté par un AFN-ε M, s'il existe une lecture de W par l'automate, en tenant compte aussi d'éventuelles transitions instantanées, qui porte l'automate depuis l'état initial vers un état final.

Théorème

Si L est un langage accepté par un AFN-ε, alors on peut construire un AFN qui accepte le même langage et donc aussi un automate déterministe qui accepte L.

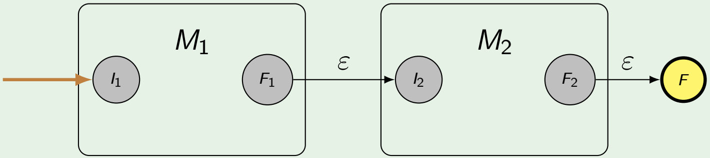
Produit de langages rationnels

Théorème

Le produit de langages rationnels est encore un langage rationnel.

Preuve

On construit un AFN-ε à partir des automates M_1 et M_2 qui acceptent L_1 et L_2 de la façon suivante :



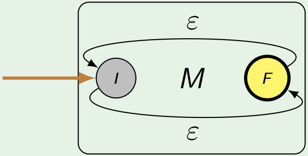
Opération étoile

Théorème

Si L est un langage rationnel, alors L^* est aussi un langage rationnel.

Preuve

A partir de l'automate M qui accepte L on peut construire un AFN-ε qui accepte L^* de la façon suivante :



Chapitre 6 : Automates à états finis

Théorème de Kleene

Théorème

Les langages rationnels sont les langages réguliers

Idée de preuve

- Les langages réguliers sont la plus petite classe de langage qui contient les langages finis et est fermée pour l'union, le produit et l'étoile
- Les langages rationnels sont aussi fermés pour ces 3 opérations donc contiennent forcément les langages rationnels.
- D'autre part à partir de chaque automate à état fini, on peut construire l'expression régulière décrivant le langage accepté. Donc les langages rationnels sont réguliers.

Conséquence

Comme conséquence du théorème de Kleene on a :

- l'intersection de langages réguliers est un langage régulier
- le complémentaire d'un langage régulier est un langage régulier
- la différence de langages réguliers est un langage régulier.

Mais aussi

- l'automate à état fini associé à une expression régulière représente un "programme" capable de dire (en un temps fini) si un mot donné appartient ou non au langage décrit par l'expression régulière.
- On a résolu le problème de décision.

Limite des langages rationnels

Problème

Il reste toutefois des questions toujours sans réponse :

- Existe-t-il des langages non rationnels ?
- Si oui, comment les reconnait-on ?

La réponse à ces deux questions est donnée par le lemme de l'étoile :

- propriété que doit posséder un langage rationnel
- les langages ne possédant pas cette propriété ne sont pas rationnels !

Lemme de l'étoile

Pumping lemma

Soit L un langage rationnel ; il existe un entier N tel que tout mot $W \in L$ de longueur supérieure à N peut se décomposer en 3 morceaux : $W = x \cdot y \cdot z$ tels que :

- $|x \cdot y| < N$
- $|y| > 0$
- $\forall k \in \mathbb{N}$,

$$x \cdot y^k \cdot z \in L$$

De plus, cet entier N est inférieur ou égal au nombre d'état d'un AFD acceptant L.

Preuve

Preuve du lemme de l'étoile

Soit M un AFD à N états qui accepte le langage L, et soit W un mot de plus de N lettres accepté par cet automate. On écrit

$$W = a_1 a_2 \dots a_k, \quad \text{avec } k > N$$

et soit q_i l'état dans lequel se trouve l'automate après avoir lu les i premières lettres du mot W :

$$q_i = \delta(I, a_1 a_2 a_3 \dots a_i), \quad \text{pour } i = 1 \dots k$$

Comme $k > N$ tous les états q_i ne peuvent pas être différents, il y en a au moins un qui apparaît deux fois.

Suite de la preuve

Soit donc q_j le premier état de M qui se répète, c'est à dire que l'on a un circuit qui commence et finit en q_j

$$q_j = q_{j'}, \quad \text{avec } j' > j$$

alors on pose

$$\begin{aligned} x &= a_1 a_2 \ldots a_j \\ y &= a_{j+1} a_{j+2} \ldots a_{j'} \\ z &= a_{j'+1} a_{j'+2} \ldots a_k \end{aligned}$$

On a bien que

$$M = x \cdot y \cdot z$$

et y correspond aux lettres qui forment la boucle sur q_j.

Fin de la preuve

On peut parcourir cette boucle autant de fois que l'on veut ce qui fait que $\forall m \geq 1$

$$\delta(I, x \cdot y^m \cdot z) \in F \implies x \cdot y^m \cdot z \in L$$

Application

Corollaire

Le langage $L = \{a^k \cdot b^k; \quad k > 0\}$ sur l'alphabet $\Sigma = \{a, b\}$ n'est pas un langage rationnel.

Preuve par l'absurde

Supposons par l'absurde que L soit rationnel. Alors il existerait un AFD qui accepte L.
Soit N le nombre d'états de cet automate et considérons le mot :

$$W = a^N \cdot b^N$$

Alors $W \in L$ et $|W| = 2N > N$. Donc en appliquant le lemme de l'étoile au mot W on aurait que $W = x \cdot y \cdot z$ avec $|x \cdot y| < N$.

Suite de la démonstration

Donc les chaînes de caractères x et y ne peuvent pas contenir de lettre b mais seulement des a.
Ce qui fait qu'en prenant le mot

$$W' = x \cdot y^2 \cdot z$$

on a que

- d'une part $W \in L$ par le lemme de l'étoile, et
- d'autre part $W' \notin L$ car il a plus de a que de b.

On est donc arrivé à une contradiction.

Chapitre 6 : Automates à états finis

Machine séquentielle

Problématique

L'inconvénient d'un automate à états finis, c'est qu'il est juste capable d'accepter ou pas des mots. Il répond donc simplement par oui ou par non.
Une machine séquentielle est un automate à état fini déterministe doté d'une capacité d'affichage (ou d'impression) :

- à chaque changement d'état provoqué par une lettre, la machine séquentielle a la possibilité de produire un output.
- En conséquent, à chaque mot en entrée, va correspondre un mot en sortie.
- Il n'y a donc plus de notion d'état final mais de fonction calculée par la machine séquentielle.

Définition

Machine séquentielle

Une machine séquentielle est un AFD avec deux éléments additionnels :

- un alphabet Γ utilisé pour les messages de sortie
- une fonction d'affichage $\sigma : E \times \Sigma \rightarrow \Gamma^*$ couplée à la fonction de transition δ.

Donc si

$$\begin{aligned} \delta(q_1, a) &= q_2 \\ \sigma(q_1, a) &= W \end{aligned}$$

on dira que la lacture de la lettre a fait passer la machine séquentielle de l'état q_1 à l'état q_2 en produisant le mot $W \in \Gamma^*$.

Fonction calculée par une machine séquentielle

Définition

La fonction calculée par une machine séquentielle est la fonction

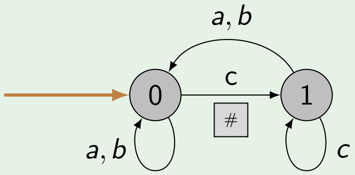
$$f : \Sigma^* \to \Gamma^*$$

qui à tout mot W construit sur l'alphabet Σ en entrée, associe le mot $f(W)$ sur l'alphabet Γ obtenu en concaténant tous les messages produits par σ lors de la lecture des lettres de W à partir de l'état initial.

Exemple

Machine séquentielle

$\Sigma = \{a, b, c\}$ et $\Gamma = \{\#\}$



- $f(aaaabbbccbc) = \#\#$
- $f(acbcacc) = \#\#\#$

Cette machine séquentielle "compte" en unaire, le nombre de groupes de c consécutifs dans le mot en entrée.