

MMIM Méthodes mathématiques en informatique musicale

Marc Chemillier

Atiam (Ircam), 13 février 2007

Chap. 4 : Improvisation et ordinateur (simulation stylistique)

- Notion d'automate fini
 - o Automate fini déterministe (AFD)
 - o Automate fini non-déterministe (AFN)
- Chaînes de Markov
- Simulation stylistique et oracle des suffixes

1. Vocabulaire

1.1 Définitions générales

Σ^* représente l'ensemble des mots sur l'alphabet Σ .

Le *mot vide* noté ϵ est le seul mot de longueur nulle.

Un *langage* est un ensemble de mots, c'est-à-dire un sous-ensemble de Σ^* .

Un mot $u \in \Sigma^*$ est *facteur* du mot $w \in \Sigma^*$ s'il existe $v, v' \in \Sigma^*$ tels que $w = vuv'$. Si $v = \epsilon$, on dit que u est *préfixe*. Si $v' = \epsilon$, on dit que u est *suffixe*.

Exemple : *abb* est facteur de *babba*, et *ba* est à la fois préfixe et suffixe, mais *aa* n'est pas facteur.

1.2 Opérations sur les langages

- opérations classiques sur les ensembles :

union \cup , intersection \cap , différence \setminus , complémentaire,

- opérations héritées de la concaténation :

La concaténation de deux langages est définie par :

$$L_1 L_2 = \{uv, u \in L_1 \text{ et } v \in L_2\}.$$

Exemple : $L_1 = \{a, ab\}$, $L_2 = \{c, bc\}$, $L_1 L_2 = \{ac, abc, abbc\}$.

La *puissance* d'un langage L est définie inductivement :

$$L^0 = \{\epsilon\}, L^{n+1} = L^n L.$$

L'*étoile* d'un langage L , notée L^* , est :

$$L^* = \{\epsilon\} \cup L \cup L^2 \dots \cup L^n \cup \dots$$

Ce langage contient un nombre infini de mots, qui sont les répétitions indéfinies de mots de L .

Exemple : $L = \{ab, b\}$, L^* est l'ensemble de tous les mots tels que aa n'est pas facteur, et a n'est pas suffixe.

On note également L^+ le langage :

$$L^+ = L \cup L^2 \dots \cup L^n \cup \dots$$

2. Définition des automates finis déterministes (AFD)

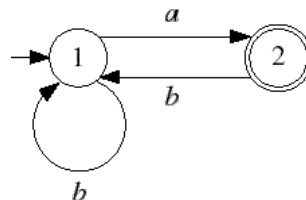
2.1 Définition générale

Définition. Un automate fini déterministe AFD sur un alphabet Σ est la donnée d'un n -uplet (Q, δ, i, F) où :

- Q est un ensemble fini d'états,
- δ est une fonction de transition de $Q \times \Sigma$ dans Q ,
- i est un état particulier de Q dit initial,
- F est une partie de Q d'états dits finals.

L'automate est dit complet lorsque la fonction δ est partout définie sur $Q \times \Sigma$.

Exemple :



$$Q = \{1, 2\},$$

$i = 1$, état noté avec une petite flèche entrante,

$F = \{2\}$, état noté avec deux cercles.

$$\delta : Q \times \Sigma \rightarrow Q$$

$$(1, a) \rightarrow 2$$

$$(1, b) \rightarrow 1$$

$$(2, b) \rightarrow 1$$

Cet automate n'est pas complet, car $\delta(2, a)$ n'est pas défini.

Pour décrire un automate, il est commode d'utiliser une table de transitions :

	1	2
1		
2		

a	2	
b	1	1

1.2 Prolongement de la fonction de transition

Le calcul de l'automate consiste à suivre des flèches, en partant d'un état initial et en s'arrêtant dans un état final. Le mot correspondant à ce calcul est la suite des étiquettes des flèches.

On prolonge δ par induction, en une fonction sur les mots de $Q \times \Sigma^* \rightarrow Q$:

(attention : la fonction δ est définie au départ sur $Q \times \Sigma$ et pas sur $Q \times \Sigma^*$):

$$\delta(q, \epsilon) = q,$$

$$\delta(q, wa) = \delta(\delta(q, w), a)$$

pour tous $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$.

$\delta(q, w)$ = état atteint après lecture du mot w depuis un état q .

Propriété d'associativité. Pour tous mots $u, v \in \Sigma^*$, on a : $\delta(q, uv) = \delta(\delta(q, u), v)$.

Exemple : dans l'automate ci-dessus

a correspond au calcul $1 \rightarrow 2$,

aba correspond au calcul $1 \rightarrow 2 \rightarrow 1 \rightarrow 2$.

donc on écrira :

$$\delta(1, aba) = 2.$$

À partir de l'état 2, on ne peut lire un mot commençant par a , mais on peut lire $babb$ par exemple : $\delta(2, babb) = 1$.

La propriété d'associativité donne :

$$\delta(1, abababb) = \delta(\delta(1, aba), babb) = \delta(2, babb) = 1.$$

Le mot aba correspond à un calcul terminal car :

$$\delta(1, aba) = 2 \in F \text{ terminal.}$$

En revanche, $abababb$ ne correspond pas à un calcul terminal de l'automate :

$$\delta(1, abababb) = 1 \text{ non terminal.}$$

1.3 Langage reconnu par un AFD

Définition. Le langage reconnu (ou accepté) par un automate AFD est l'ensemble des mots qui correspondent à un calcul de l'automate partant d'un état initial et s'arrêtant dans un état final.

Exemple : le langage reconnu par l'automate ci-dessus est noté

$$a(bb^*a)^*$$

Les automates finis sont les modèles de machine les plus simples : ils n'ont aucun support de mémoire externe (comme la pile d'un automate à pile).

Leur mémoire est donc finie (espace constant), et correspond à leur nombre d'états. Par exemple, dans l'automate ci-dessus, l'état 1 permet de se souvenir qu'il faut lire un a pour sortir.

Exemples :

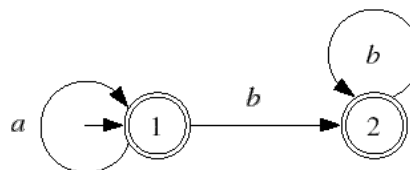
- distributeur de café : les pièces introduites sont les symboles de l'alphabet, l'état terminal est atteint quand le montant est supérieur au montant demandé,
- mécanisme contrôlant le code d'accès d'une porte : les chiffres tapés sont les symboles, l'état terminal est celui qui déclenche l'ouverture,
- rubiks cube : l'alphabet est l'ensemble des rotations des 6 faces.

3. Clôture par complément des langages reconnus par AFD

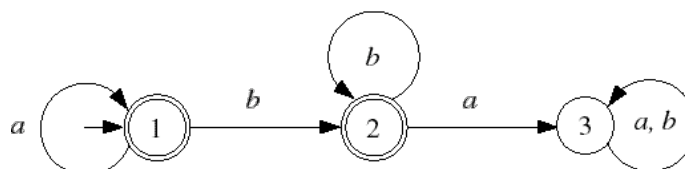
Propriété. Si L est un langage reconnu par AFD, alors son complémentaire $\Sigma^* \setminus L$ l'est aussi.

Exemple : $L = \{a^i b^j, i, j \in \mathbb{N}\}$,

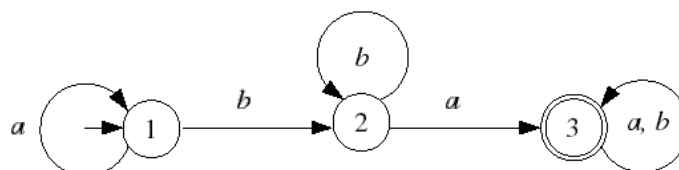
$$\Sigma^* \setminus L = \{w \in \Sigma^*, ba \text{ est facteur de } w\}$$



On complète l'automate :



Puis on intervertit les états finals :



Construction :

Si $A = (Q, \delta, i, F)$ est un AFD complet qui reconnaît L , alors

$A = (Q, \delta, i, Q \setminus F)$ reconnaît $\Sigma^* \setminus L$.

4. Définition des automates finis non-déterministes (AFN)

4.1 Définition générale

Définition. Un *automate fini non-déterministe* AFN sur un alphabet Σ est la donnée d'un n -uplet (Q, δ, I, F) où :

- Q est un ensemble fini d'états,
- δ est une fonction de transition de $Q \times \Sigma$ dans $\mathcal{P}(Q)$, ensemble des parties de Q ,
- I est une partie de Q d'états dits initiaux,
- F est une partie de Q d'états dits finals.

1.2 Fonctionnement

Un automate fini non-déterministe est un automate tel que dans un état donné, il peut y avoir plusieurs transitions avec la même lettre.

Au temps initial, la machine est dans un état initial $i \in I$.

Si à l'instant t la machine est dans l'état q et lit a , alors à l'instant $t + 1$

- si $\delta(q, a) = \emptyset$, la machine se bloque,
- si $\delta(q, a) \neq \emptyset$, la machine se met dans l'un des états $\in \delta(q, a)$ (et lit le symbole suivant).

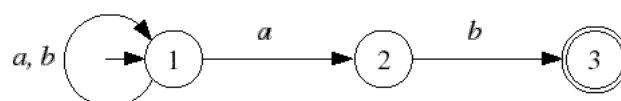
On voit que le fonctionnement n'est pas complètement « déterminé », car on ne sait pas à l'avance quel état la machine doit choisir dans $\delta(q, a)$, d'où le terme non-déterministe.

On pourrait donc penser que les AFN n'ont aucun intérêt dans la pratique.

En fait, il n'en est rien car on peut montrer que tout AFN peut être remplacé par un AFD équivalent (voir algorithme de déterminisation).

Donc on se sert des AFN, car ils peuvent être très utiles pour exprimer certains problèmes, et on les remplace ensuite par des AFD.

Exemple : ensemble des mots qui se terminent par ab



$Q = \{1, 2, 3\}$,

$I = \{1\}$,

$$F = \{3\}.$$

δ définie par la table de transition :

	1	2	3
a	$\{1,2\}$	\emptyset	\emptyset
b	1	3	\emptyset

1.3 Langage reconnu par un AFN

Un mot u est accepté par un AFN s'il existe un chemin d'étiquette u , partant de l'un des états initiaux, et arrivant à l'un des états finals.

Définition. Le langage reconnu (ou accepté) par un automate AFN est l'ensemble des mots acceptés par l'AFN, c'est-à-dire qui correspondent à un calcul de l'automate partant d'un état initial et s'arrêtant dans un état final.

5. Détermination d'un AFN

5.1 Algorithme de détermination d'un AFN

Un AFD est un cas particulier d'AFN, avec $\text{Card}(\delta(q, a)) \leq 1$ pour tous $q \in Q$, $a \in \Sigma$.

Donc tout langage reconnu par un AFD est reconnu par un AFN.

Plus surprenant, on a la réciproque.

Théorème (Rabin-Scott). *Tout langage reconnu par un AFN peut être reconnu par un AFD.*

Le principe de la construction est de prendre comme états de l'AFD les ensembles d'états de l'AFN. L'unique état initial de l'AFD est l'ensemble I des états initiaux de l'AFN.

Construction pour la détermination d'un AFN :

Si $A = (Q, \delta, I, F)$ est un AFN qui reconnaît L , alors

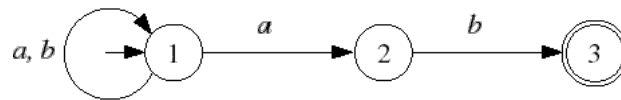
$A_{\text{det}} = (\mathcal{P}(Q), \delta_{\text{det}}, I, F_{\text{det}})$ est un AFD qui reconnaît le même langage L , avec :

- $F_{\text{det}} = \{X \in \mathcal{P}(Q), X \cap F \neq \emptyset\},$
- $\delta_{\text{det}}(X, a) = \bigcup_{q \in X} \delta(q, a).$

Pratiquement, on ne construit que les états accessibles à partir de I , de proche en proche :

on part de l'état initial I , puis on calcule toutes les transitions qui partent de I , puis on recommence avec les nouveaux états obtenus, etc.

Exemple :



état initial : $I = \{1\}$

transition par a : $\{1, 2\}$

transition par b : $\{1\}$

état $\{1, 2\}$

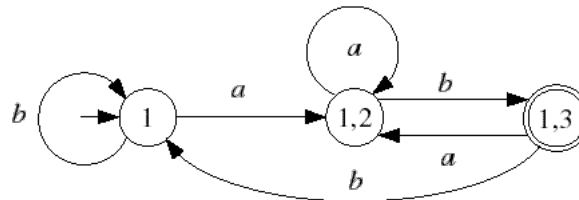
transition par a : $\{1, 2\}$

transition par b : $\{1, 3\}$

état $\{1, 3\}$

transition par a : $\{1, 2\}$

transition par b : $\{1\}$



1.2 Équivalence entre AFD et AFN

Conclusion. *Les langages reconnus par les AFN sont exactement les langages reconnus par les AFD.*

Avec des AFN, il est facile de montrer de nombreuses propriétés de clôture de langages reconnus par AFN (notamment union et concaténation). Mais comme tout langage reconnu par un AFN l'est par un AFD et réciproquement, ces propriétés de clôture sont vraies également pour les langages reconnus par AFD.

Théorème. *Si L, L' sont des langages reconnus par AFD, alors il en est de même :*

- *du complémentaire $\Sigma^* \setminus L$,*
- *de l'intersection $L \cap L'$,*
- *de l'union $L \cup L'$,*
- *de la concaténation LL' ,*
- *de l'étoile L^**

6. AFN avec probabilités de transitions : chaînes de Markov

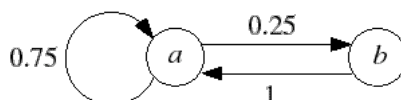
Une chaîne de Markov est un AFN dont les transitions sont munies de probabilités.

Exemple : Construction des probabilités de transitions dans un mot.

Soit le mot $w = aaabaa$

On compte le nombre de fois où chaque lettre est suivie d'une même lettre :

	a	b
a	3	1
b	1	0



Dans cet exemple, le contexte est de longueur 1, mais on peut étudier des contextes plus long, par exemple de longueur 2 :

- combien de fois aa est suivi de a ? de b ?
- combien de fois ab est suivi de a ? de b ?
- etc

? (creer-table "Bonjour, ceci est le cours sur les automates, c'est-à-dire une structure de calcul très simple, mais qui a beaucoup de propriétés intéressantes")

```

((#\é 3 (#\r 1) (#\s 1) (#\t 1)) (#\q 1 (#\u 1)) (#\p 4 (#\r
2) (#\Space 1) (#\l 1)) (#\è 1 (#\s 1)) (#\d 3 (#\e 2) (#\i
1)) (#\à 1 (#\ - 1)) (#\ - 2 (#\d 1) (#\à 1)) (#\' 1 (#\e 1))
(#\m 3 (#\p 1) (#\a 2)) (#\a 7 (#\n 1) (#\Space 1) (#\i 1)
(#\l 1) (#\t 1) (#\u 2)) (#\l 5 (#\Space 1) (#\c 1) (#\e 3))
(#\t 10 (#\é 2) (#\u 1) (#\r 2) (#\ - 1) (#\e 2) (#\o 1)
(#\Space 1)) (#\s 13 (#\a 1) (#\s 1) (#\i 1) (#\ , 1) (#\u 1)
(#\Space 5) (#\t 3)) (#\i 7 (#\n 1) (#\é 1) (#\s 1) (#\m 1)
(#\r 1) (#\Space 2)) (#\e 15 (#\a 1) (#\ , 1) (#\Space 6) (#\s
6) (#\c 1)) (#\c 8 (#\u 1) (#\a 1) (#\t 1) (#\' 1) (#\o 2)
(#\i 1) (#\e 1)) (#\Space 21 (#\i 1) (#\p 1) (#\b 1) (#\q 1)
(#\m 1) (#\t 1) (#\d 2) (#\u 1) (#\a 2) (#\s 3) (#\l 2) (#\e
1) (#\c 4)) (#\ , 3 (#\Space 3)) (#\r 10 (#\i 1) (#\o 1) (#\è
1) (#\u 1) (#\e 3) (#\Space 1) (#\s 1) (#\ , 1)) (#\u 11 (#\p
1) (#\i 1) (#\l 1) (#\c 2) (#\n 1) (#\t 1) (#\r 4)) (#\j 1
(#\o 1)) (#\n 4 (#\t 2) (#\e 1) (#\j 1)) (#\o 6 (#\p 1) (#\m
1) (#\u 3) (#\n 1)) (#\B 2 (#\e 1) (#\o 1))
  
```

? (markov 5000)

"'es s complcai ma truromai cturésteste des quoronjop caless s
lc'e dint s primp e, les cop comaure qurouciés cop satés
uctér, e qunjounjomantérèsauières quront e ciés cière s s
près mp esss auci curoulesainjompres cureaintères, c'e
dintériétététuc'ecuimanjompr a couprèsalcturesi di ecoi près
le aiésuc'estér térintrèsa b"

? (markov 5000)

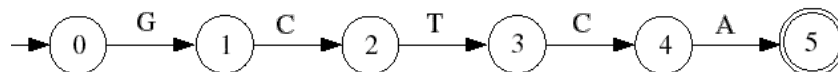
" i pre al ce c'esulestérsururs ines aur sa t-à-à-dess cout-à-de les e, diromales p cintes, cis le ul destêtre s trestérature b"

7. Oracle des suffixes

L'*oracle des suffixes* est un automate qui reconnaît tous les suffixes d'un mot x , mais avec quelques mots en plus. On peut donner directement l'AFD correspondant (sans détermination), par une construction très simple à mettre en œuvre.

L'*écorché* d'un mot x de longueur n est l'AFD obtenu avec $n + 1$ états, et n flèches correspondant aux lettres successives du mot. Dans cet automate, on voit clairement que les états ont un rôle de « mémoire » : chaque état mémorise la position dans le mot x .

Exemple : écorché du motif $x = \text{GCTCA}$



Petite remarque : pourquoi les lettres G, C, T, A ?

Le génome est fait d'ADN. Les gènes contenus dans le génome sont codés sous forme chimique le long des molécules d'ADN. Celles-ci sont constituées par l'enchaînement de "maillons" élémentaires nommés nucléotides. Les nucléotides ont une partie variable - une base, du point de vue chimique - qui peut exister sous 4 formes différentes ; ces formes sont symbolisées par les lettres A, T, G et C. Les instructions sont donc écrites dans un alphabet chimique à 4 lettres seulement.

L'une des motivations qui a conduit à définir l'oracle des suffixes est l'étude des séquences d'ADN. Il est en général utilisé de façon négative, quand on veut vérifier qu'aucun suffixe d'un motif n'apparaît dans une séquence.

Construction de l'oracle des suffixes : on part de l'écorché de x , et on rajoute des transitions à l'aide d'une fonction dite de « lien suffixiel » entre états. On construit simultanément la fonction f et les transitions de l'automate.

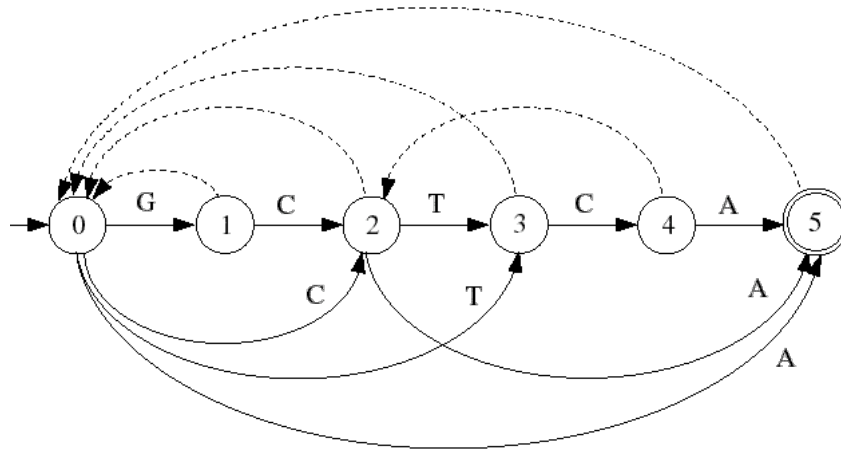
Supposons l'oracle construit jusqu'à l'état p , avec les liens suffixiels de tous les états jusqu'à p compris. La lettre suivante a de x donne un nouvel état $p+1 = \delta(p, a)$.

Pour rajouter les transitions, on suit les liens suffixiels déjà existants $f(p), f(f(p)), \text{etc.}$

- si $\delta(f(p), a)$ non défini, on ajoute une transition $\delta(f(p), a) = p+1$, et on continue à suivre les liens,
- si $\delta(f(p), a)$ est défini pour $p \neq 0$, on stoppe (pas de nouvelle transition) et on crée le lien suffixiel de $p+1$ en posant $f(p+1) = \delta(f(p), a)$,

- si $p = 0$, on stoppe (pas de nouvelle transition) et on crée le lien suffixiel de $p+1$ en posant $f(p+1) = 0$.

Pour le motif GCTCA, l'oracle des suffixes donne l'AFD suivant (les liens suffixiels sont indiqués par des flèches en pointillé au-dessus) :



On peut vérifier que les suffixes de GCTCA sont reconnus :

GCTCA

CTCA

TCA

CA

A

Existe-t-il d'autres mots reconnus par l'oracle qui ne sont pas suffixes de GCTCA ?

Oui, il y en a un :

GCA