MMIM Méthodes mathématiques en informatique musicale

Marc Chemillier

Atiam (Ircam), 14 février 2006

Chap. 4 : Improvisation et ordinateur (simulation stylistique)

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

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 $v = \varepsilon$, on dit que u est *préfixe*. Si $v' = \varepsilon$, 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 \, complémentaire,

• opérations héritées de la concaténation :

La concaténation de deux langages est définie par :

$L_1L_2 = \{uv, u \in L_1 \text{ et } v \in L_2\}$.

Exemple : $L_1 = \{a, ab\}$, $L_1 = \{c, bc\}$, $L_1L_2 = \{ac, abc, abbc\}$.

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

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

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

$L^* = \{\varepsilon\} \cup L \cup L^2 \ldots \cup L^n \cup \ldots$

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 :

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

2. Définition des automates finis déterministes (AFD)

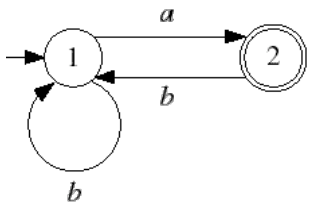
2.1 Définition générale

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

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 $Q \times \Sigma^* \to Q$:

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

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

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.

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 $1 \to 2$,

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

En revanche, *abab* ne correspond pas à un calcul terminal de l'automate : $\delta(1, abab) = 1$ 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 :

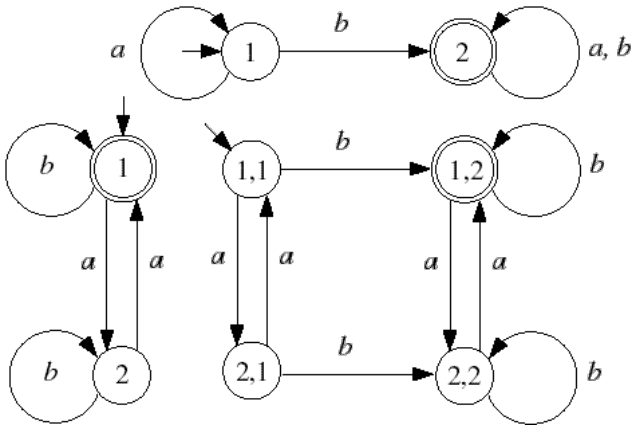
- distributeur de café : les pièces introduites sont les symboles de l'alphabet, l'état terminal est atteint quand le montant est supérieur au montant demandé,
- mécanisme contrôlant le code d'accès d'une porte : les chiffres tapés sont les symboles, l'état terminal est celui qui déclenche l'ouverture,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

3. Propriétés de clôture des langages reconnus par AFD

3.1 Clôture par intersection

Propriété. *Si L_1, L_2 sont des langages reconnus par AFD, alors $L_1 \cap L_2$ l'est aussi.*

Exemple : $L_1 = \{w \in \Sigma^*, w$ a au moins un $b\}$,
$L_2 = \{w \in \Sigma^*, w$ a un nombre pair de $a\}$

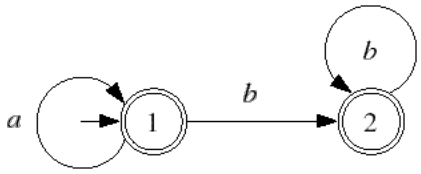


Construction par produit d'automates :
Si $A_i = (Q_i, \delta_i, i_i, F_i)$ reconnaît L_i pour $i = 1, 2$, alors
$A = (Q_1 \times Q_2, \delta, (i_1, i_2), F_1 \times F_2)$ reconnaît $L_1 \cap L_2$.

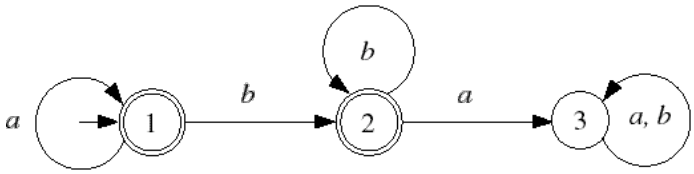
1.2 Clôture par complémentation

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

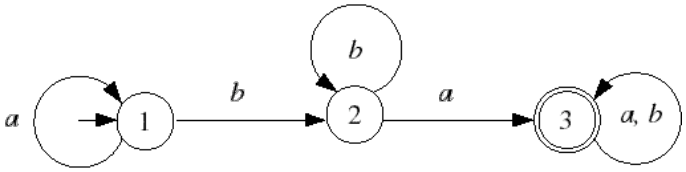
Exemple : $L = \{a^i b^j, i, j \in \mathbf{N}\}$,
$\Sigma^* \backslash L = \{w \in \Sigma^*, ba$ est facteur de $w\}$



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

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

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* $\mathscr{P}(Q)$, *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 $t + 1$

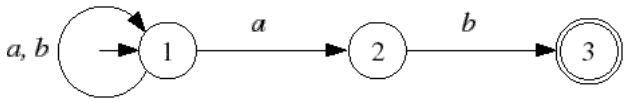
- si $\delta(q, a) = \varnothing$, la machine se bloque,

si $\delta(q, a) \neq \varnothing$, 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 va choisir dans $\delta(q, a)$, d'où le terme non-déterministe.

Malgré cela, les AFN peuvent être très utiles pour exprimer certains problèmes.

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}	$\varnothing$	$\varnothing$
b	1	3	$\varnothing$

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. Déterminisation d'un AFN

5.1 Algorithme de déterminisation 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 déterminisation d'un AFN :

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

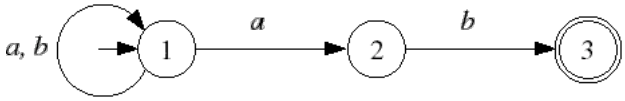
$A_{det} = (\mathscr{P}(Q), \delta_{det}, I, F_{det})$ est un AFD qui reconnaît le même langage L, avec :

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

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 $\{1, 2\}$

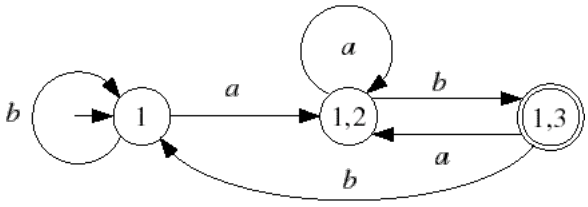
transition par a : $\{1, 2\}$

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

état $\{1, 3\}$

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

transition par b : $\{1\}$



5.2 Équivalence entre AFD et AFN

Conclusion : les langages reconnus par les AFN sont exactement les langages reconnus par les AFD.

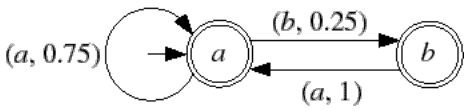
6. Chaînes de Markov

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

Exemple : 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 cturestéste des quronjop caless s
lc'e dint s primp e, les cop comaure qurouciés cop satés
uctér, e qunjounjomantérèsauiétres quront e ciés ciére s s
près mp esss auci curoulesainjompres cureaintéres, c'e
dintériétététuc'ecuimanjompr a couprèsalcturesi di ecoui près
le aiésuc'estr 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 destétre s trestéroure b"