MMIM Modèles mathématiques en informatique musicale

Marc Chemillier

Master M2 Atiam (Ircam), 2010-2011

Notions théoriques sur les langages formels

- Définitions générales
  - Mots, langages
  - Monoïdes
- Notion d'automate fini
  - Automate fini déterministe (AFD)
  - Automate fini non déterministe (AFN)
- Reconnaissabilité dans un monoïde quelconque

1. Définitions générales

1.1 Mots

Un *mot* fini u est une suite de symboles. La longueur de u notée $|u|$ est le nombre de symboles du mot. Le *mot vide* noté ε est le seul mot de longueur nulle. L'ensemble des symboles noté Σ est appelé *alphabet*, et l'ensemble des mots sur l'alphabet Σ est noté Σ^*.

Pour deux mots $u = u_1 \ldots u_n$ et $v = v_1 \ldots v_m$, on définit la *concaténation* uv comme le mot obtenu en mettant les lettres de v à la suite de celles de u :

$uv = u_1 \ldots u_n v_1 \ldots v_m$.

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.

Un mot fini u est *périodique* si $u = x^n$ pour $n \geq 2$. Tout mot non périodique est dit *primitif*.

L'idée fondamentale de ce cours est que la notion de mot permet de représenter le principe de succession d'événements dans une séquence musicale, d'où son intérêt pour la modélisation en informatique musicale.

1.2 Langages

Les sous-ensembles de Σ^* sont appelés des *langages* (c'est-à-dire des ensembles de mots, ou de séquences musicales dans le contexte de l'informatique musicale).

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 : $L_1 = \{a, ab\}$, $L_2 = \{c, bc\}$, $L_1L_2 = \{ac, abc, abbc\}$.

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 :

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

1.3 Monoïdes

Un *monoïde* est un ensemble muni d'une opération

• associative,

• possédant un élément neutre 1.

Un *sous-monoïde* est un sous-ensemble fermé pour l'opération et contenant l'élément neutre. L'ensemble Σ^* des mots sur l'alphabet Σ est un monoïde pour la concaténation, dont l'élément neutre est le mot vide ε. Ce monoïde est engendré par l'ensemble de ses lettres, c'est-à-dire l'alphabet Σ.

Soient deux mots $u = u_1 \ldots u_n$ et $v = v_1 \ldots v_m$. L'égalité $u = v$ équivaut à l'égalité de toutes les lettres de u et v une à une, c'est-à-dire $u_i = v_i$ pour tout $i \leq n = m$. Pour cette raison, on dit que Σ^* est le *monoïde libre* sur Σ.

Remarque : Cette notion de « liberté » est similaire à celle des espaces vectoriels lorsqu'on dit qu'une famille de vecteurs est libre si l'égalité de deux combinaisons linéaires de ces vecteurs implique l'égalité de chacun de leurs coefficients.

Un sous-monoïde de Σ^* est-il toujours « libre » par rapport à l'ensemble qui l'engendre ?

-> Non. Par exemple, le sous-monoïde engendré par $C = \{a, ab, c, bc\}$ n'est pas libre :

$(a)(bc) = (ab)(c)$

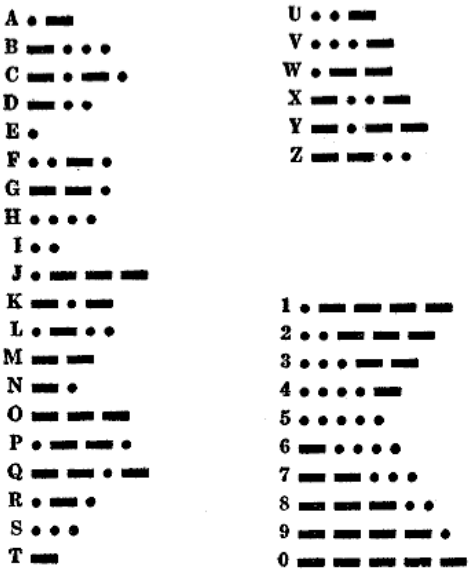
Un *code* est une partie C de Σ^* qui engendre un sous-monoïde libre de Σ^*. Tout élément du sous-monoïde s'écrit d'une manière unique comme suite d'éléments de C. Cela signifie qu'on peut « décoder » les éléments de C^*.

• L'ensemble Σ^n des mots de longueur n est un code, en particulier l'alphabet Σ est un code.

• Attention : Le code morse n'est pas un code dans le sens ci-dessus.

INTERNATIONAL MORSE CODE

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to five dots.



On a E = « • », T = « — », et N = « —• », donc on a N = TE. Le décodage en morse se fait grâce à la présence de séparateurs entre les lettres.

Un *morphisme* de monoïde est une application φ telle que $\varphi(xy) = \varphi(x)\varphi(y)$ et $\varphi(1) = 1$.

Toute application d'un alphabet Σ dans un monoïde quelconque se prolonge dans Σ^* en un unique morphisme de monoïdes. Il en résulte que pour définir un morphisme sur Σ^*, il suffit de définir l'image de toutes ses lettres (de la même manière dans un espace vectoriel, on définit une application linéaire par l'image de tous les vecteurs de la base).

2. Notion d'automate fini

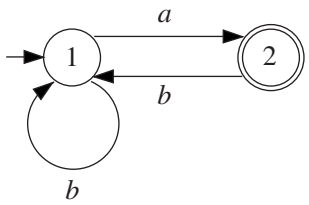
2.1 Définition des automates finis déterministes (AFD)

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.

$\delta : Q \times \Sigma \to Q$

$(1, a) \to 2$

$(1, b) \to 1$

$(2, b) \to 1$

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

2.2 Langage reconnu par un AFD

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

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.*

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,

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).

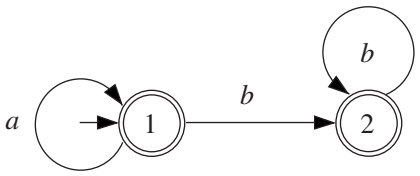
Leur mémoire est donc finie (espace constant), et correspond à leur nombre d'états. Par exemple, dans l'automate ci-dessus, l'état 1 permet de se souvenir qu'il faut lire un *a* pour sortir.

2.3 Clôture par complément des langages reconnus par AFD

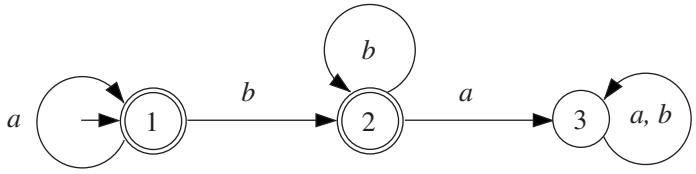
Propriété. *Si L est un langage reconnu par AFD, alors son complémentaire* $\Sigma^* \setminus L$ *l'est aussi.*

Exemple : L est l'ensemble des mots comportant une suite de a suivie éventuellement d'une suite de b, soit $L = \{a^i b^j, i, j \in \mathbf{N}\}$,

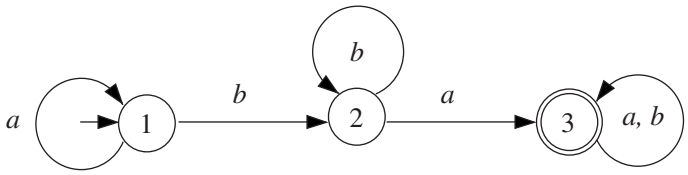
$\Sigma^*\backslash L = \{w \in \Sigma^*, ba \text{ est facteur de } w\}$



On <u>complète</u> l'automate :



Puis on intervertit les états finals :



Construction :

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

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

3. Généralisation aux automates finis non déterministes (AFN)

3.1 Définition des AFN et langages réguliers

Définition. *Un <u>automate fini non déterministe</u> 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*.

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

Un AFN est un automate dans lequel d'un état peuvent partir <u>plusieurs flèches avec la même étiquette</u>.

Ainsi, les AFD apparaissent comme des cas particuliers d'AFN avec pour chaque état une seule flèche pour chaque étiquette : $\mathrm{Card}(\delta(q, a)) \leq 1$ pour tous $q \in Q$, $a \in \Sigma$.

Il en résulte que tout langage reconnu par un AFD est reconnu par un AFN.

Plus surprenant, on a la réciproque, c'est-à-dire que les AFD et les AFN reconnaissent exactement les mêmes langages.

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

La déterminisation d'un AFN est la construction de l'AFD correspondant. Elle se fait en prenant comme états de l'AFD les ensembles d'états de l'AFN, c'est-à-dire $\mathscr{P}(Q)$.

Définition. *On appelle langage régulier tout langage de* Σ^* *reconnu (ou accepté) par un automate fini, qu'il soit AFD ou AFN.*

S'il y a équivalence des langages reconnus par AFD et AFN, quel est l'intérêt des AFN ?

-> Ils peuvent faciliter certaines constructions, par exemple :

- union de deux langages,
- ensemble des mots se terminant par un motif donné.

3.2 Clôture par union des langages reconnus par AFN

Construction d'un AFN pour l'union de deux langages réguliers :

Si $A_1 = (Q_1, \delta_1, i_1, F_1)$ et $A_2 = (Q_2, \delta_2, i_2, F_2)$ sont des AFD disjoints reconnaissant L_1 et L_2, alors $A = (Q_1 \cup Q_2, \delta_1 \cup \delta_2, \{i_1, i_2\}, F_1 \cup F_2)$ est un AFN reconnaissant $L_1 \cup L_2$.

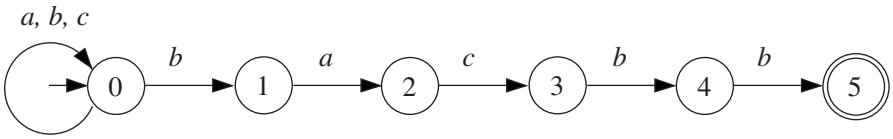
Pourquoi l'automate union A n'est pas un AFD ?

-> Parce qu'il a plusieurs états initiaux $\{i_1, i_2\}$.

Pour les propriétés de clôture de ce type (union, intersection, complément), il existe une approche algébrique plus générale utilisant la notion de reconnaissabilité dans un monoïde quelconque (voir plus loin).

3.3 Construction d'un AFN pour les mots se terminant par *x*

L'écorché d'un mot x de longueur n est l'AFD de $n + 1$ états avec n flèches correspondant aux lettres successives du mot. L'ensemble des mots se terminant par x est $\Sigma^* x$. On l'obtient en ajoutant une boucle sur l'état initial de l'écorché de x avec toutes les lettres de l'alphabet. Exemple : $x = bacbb$



Pourquoi l'automate obtenu n'est pas un AFD ?

-> il a deux flèches partant de l'état initial avec la lettre b.

Application à la recherche d'un motif dans un texte (algorithme de Morris & Pratt) :

• il faut obtenir un AFD reconnaissant $\Sigma^* x$

• on lit le texte t dans cet AFD. Que se passe-t-il si on arrive à l'état terminal ?

-> On a trouvé le motif x dans le texte t.

L'algorithme de Morris & Pratt calcule un AFD pour $\Sigma^* x$ grâce à une fonction de saut définie sur les états p de l'écorché de x. Soit w le mot lu de l'état 0 à l'état p :

$f(p)$ = état d'arrivée du plus long suffixe propre de w qui est aussi préfixe de w (donc de x)

Exemple : Fonction de saut pour l'écorché de $x = bacbb$:

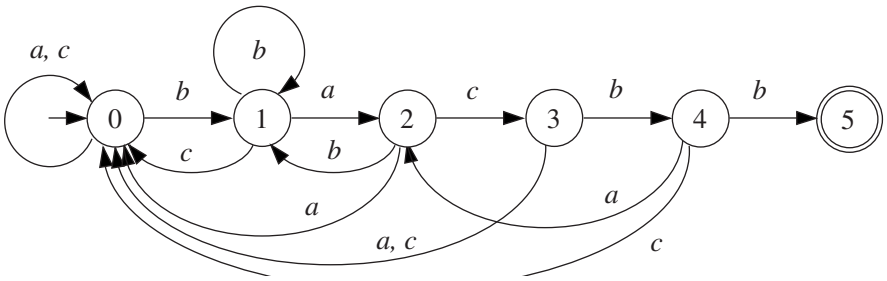
État p	0	1	2	3	4	5
$f(p)$		0	0	0	1	1

On ajoute de nouvelles transitions dans l'écorché de x de la manière suivante :

pour toute lettre a telle que $\delta(p, a)$ non défini, on pose

• $\delta(p, a) = \delta(f(p), a)$ si $p \neq 0$,

• $\delta(0, a) = 0$.



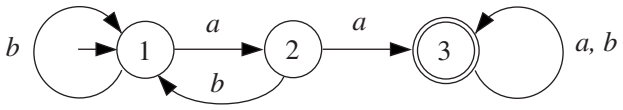
Exemple : Rechercher le motif $x = bacbb$ dans le texte $t = abacbacbba$

a b a c *b a c b b* *a*

0 1 2 3 4 2 3 4 **5** motif trouvé !!!

4. Reconnaissabilité dans un monoïde quelconque et propriétés de clôture

4.1 Automates finis et morphismes de monoïdes



Dans la table de transition d'un automate, on peut considérer que chaque lettre de l'alphabet définit une *relation* entre les états de Q dans Q, soit $a : q_1 \rightarrow q_2$ ssi $q_2 \in \delta(q_1, a)$.
On peut alors composer ces relations (par exemple, le carré a^2 correspond à la relation définie par le mot *aa* dans l'automate) et compléter la table de transition.

	1	2	3	
a	2	3	3	
b	1	1	3	
a^2	3	3	3	
ab	1	3	3	
ba	2	2	3	
b^2	1	1	3	$= b$
aba	2	3	3	$= a$
bab	1	1	3	$= b$

etc.

On voit apparaître certaines égalités parmi les relations entre états, et on peut vérifier que les huit mots de longueur 3 ne donnent pas de relation nouvelle sur les états.

On obtient donc cinq relations distinctes sur les états de l'automate. Les relations associées à a, b engendrent un sous-monoïde du monoïde des relations sur les états {1, 2, 3}, qui est un monoïde fini. Ce sous-monoïde contient six éléments :

• les cinq relations distinctes,

• la relation identité *Id* (chaque état 1, 2, 3 est envoyé sur lui-même) qui est l'élément neutre.

Voici la table du sous-monoïde $\{Id, a, b, a^2, ab, ba\}$:

	Id	a	b	a^2	ab	ba
Id	Id	a	b	a^2	ab	ba
a	a	a^2	ab	a^2	a^2	a
b	b	ba	b	a^2	b	ba
a^2	a^2	a^2	a^2	a^2	a^2	a^2
ab	ab	a	ab	a^2	ab	a
ba	ba	a^2	b	a^2	a^2	ba

Quelle condition doit vérifier la relation associée à un mot pour que celui-ci soit reconnu par l'automate ?

-> Il faut que la relation envoie l'état initial sur l'un des états finals.

Dans l'exemple ci-dessus, il n'y a qu'une relation de ce type, soit a^2 qui envoie 1 sur 3.

Les mots reconnus par l'automate sont ceux dont la relation associée est a^2. On peut écrire cela $\varphi^{-1}(a^2)$ en introduisant le morphisme φ de Σ^* dans le monoïde fini des relations entre états de l'automate, qui à un mot associe la relation correspondante.

Cela conduit à effectuer la généralisation suivante :

Définition. *Une partie X d'un monoïde quelconque M est reconnaissable si et seulement s'il existe un morphisme φ de M dans un monoïde fini N et une partie Z de N tels que $X = \varphi^{-1}(Z)$.*

Proposition. *Par définition, pour des monoïdes quelconques, l'image réciproque par un morphisme d'une partie reconnaissable est reconnaissable.*

Attention : ce n'est pas vrai pour l'**image directe**.

On a déjà vu la clôture des langages réguliers par complément et union.

On obtient directement toutes les propriétés de clôture par opérations ensemblistes des langages reconnus par automate grâce à ces formules de théorie des ensembles :

$\varphi^{-1}(X \cup Y) = \varphi^{-1}(X) \cup \varphi^{-1}(Y)$

$\varphi^{-1}(X \cap Y) = \varphi^{-1}(X) \cap \varphi^{-1}(Y)$

$\varphi^{-1}(X \setminus Y) = \varphi^{-1}(X) \setminus \varphi^{-1}(Y)$

Théorème. *L'ensemble des parties reconnaissables Rec(M) d'un monoïde quelconque M est fermé par les opérations ensemblistes (union, intersection, complément).*

Remarque : Dans un monoïde quelconque M,

• on n'a pas la fermeture de Rec(M) par les opérations sur les langages (**produit, étoile**),

• Rec(M) ne contient pas nécessairement les **parties finies** contrairement à Rec(Σ^*).

Exemple :

- un singleton $\{x\}$ est reconnaissable dans le monoïde libre Σ^* (écorché de x)

- dans un groupe infini M, les singletons $\{x\}$ ne peuvent pas être reconnaissables :

-> card(N) **fini** < card(M), donc φ n'est pas injectif, donc il existe z, z' distincts tels que $\varphi(z) = \varphi(z')$, donc dans le groupe M on a $y = z'z^{-1}$ tel que $\varphi(y) = 1_M$, ce qui donne $\varphi(xy) = \varphi(x)\varphi(y) = \varphi(x) = z$, donc $\varphi^{-1}(z)$ contient $\{x, xy\} \neq \{x\}$ ce qui montre que $\{x\}$ ne peut pas être reconnu.

4.2 Parties rationnelles et théorème de Kleene

La clôture par produit et étoile et l'inclusion des parties finies ne sont pas vérifiées par Rec(M). On introduit une autre famille Rat(M) du monoïde quelconque M qui vérifie ces propriétés :

Définition. *L'ensemble des parties rationnelles Rat(M) d'un monoïde quelconque M est le plus petit ensemble de parties de M*

• *contenant les parties finies,*

• *fermé par union, produit et étoile.*

A priori, on a : Rat(M) ≠ Rec(M).

Proposition. *Pour des monoïdes quelconques, l'image directe par un morphisme d'une partie rationnelle est rationnelle.*

(attention : pour les parties reconnaissables, c'était l'image réciproque).

Dans le cas particulier du monoïde libre Σ*, on a un **résultat remarquable** :

Théorème (Kleene)**.** *Dans le monoïde libre Σ*, l'ensemble des parties reconnaissables est exactement égal à l'ensemble des parties rationnelles Rec(Σ*) = Rat(Σ*), ce sont les parties régulières reconnaissables par automate fini.*

Corollaire. *L'image directe et l'image réciproque par un morphisme de monoïdes libres d'une partie régulière (reconnaissable ou rationnelle) est régulière (reconnaissable ou rationnelle).*

Références

• références générales

Lothaire M., *Combinatorics on Words*, Encyclopedia of Mathematics, Vol. 17, Addison-Wesley, 1983 (réédité Cambridge University Press, 1997).

Berstel Jean, *Automates et grammaires*, Cours de Licence d'informatique, Université de Marne-la-vallée, 2004-05.

Chemillier, Marc, Grammaires, automates et musique, J.-P. Briot, F. Pachet (éds.), *Informatique musicale*, Traité IC2, Hermès, Paris, 2004, chap. 6, p. 195-230.

Chemillier M., Synchronisation of musical words, *Theoretical Computer Science* **310** (2003) 35-60.

Chemillier M., Structure et méthode algébriques en informatique musicale, Thèse Université Paris 7, LITP, 1990.