MMIM Modèles mathématiques en informatique musicale

Marc Chemillier

Master Atiam (Ircam), 2009-2010

Notions théoriques sur les langages formels (Partie II)

- Automate avec ε-transition et clôture des langages réguliers
- Reconnaissabilité dans un monoïde quelconque
- Notion de grammaire formelle

1. Automates AFN avec ε-transitions et clôture des langages réguliers

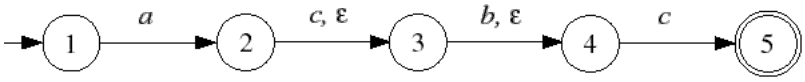
Attention : le fonctionnement normal de l'oracle est celui d'un automate, c'est-à-dire qu'**on n'utilise pas les liens suffixiels** (ils ne servent qu'à la construction comme les échafaudages d'une façade qui sont retirés après travaux). En revanche, dans les applications musicales avec OMax, il est utile de considérer les liens suffixiels comme des ε-transitions de l'automate (c'est-à-dire des transitions étiquetées par le mot vide), donc de les utiliser en génération.

1.1 Définition

Définition. *On généralise encore la notion d'automate en introduisant des transitions avec le mot vide* ε *(automate asynchrone). La fonction de transition est alors définie de* $Q \times (\Sigma \cup \{\varepsilon\})$ *dans* $\mathscr{P}Q$*, ensemble des parties de Q. Ces transitions s'effectuent sans lecture de lettre.*

Comme pour les AFN, un mot *u* est accepté par un AFN avec ε-transitions s'il existe un chemin d'étiquette *u*, partant de l'un des états initiaux, et arrivant à l'un des états finals.

Exemple :



Langage reconnu : $L = \{acbc, acc, abc, ac\}$

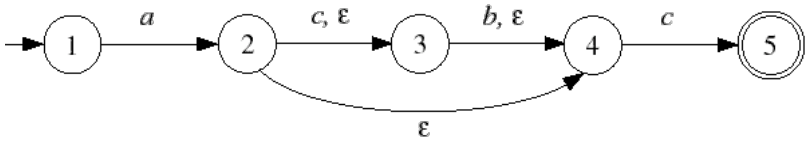
1.2 Transformation en AFN sans ε-transition

Proposition. *Tout langage reconnu par un AFN avec ε–transitions peut être reconnu par un AFN (sans ε–transitions) ayant le même nombre d'états.*

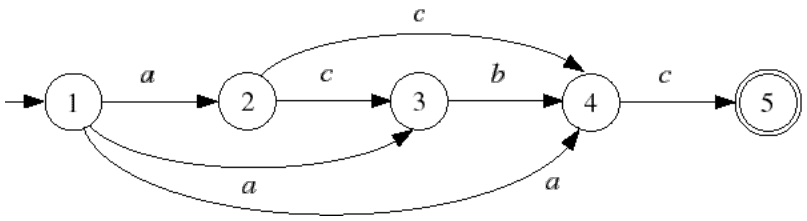
La construction de l'AFN équivalent à un automate avec ε–transitions consiste à prolonger δ par une technique de **fermeture transitive** sur les ε–transitions, puis en ajoutant de nouvelles transitions chaque fois qu'une transition étiquetée par une lettre peut être suivie d'une ε–transition.

Exemple :

• Fermeture transitive des ε-transitions



• Ajout de nouvelles transitions et suppression des ε-transitions



L'AFN obtenu reconnaît bien le même langage : $L = \{acbc, acc, abc, ac\}$

Remarque : Si on utilise les liens suffixiels comme des ε–transitions dans l'oracle, que peut-on dire du langage reconnu ?

-> C'est le monoïde libre Σ^* tout entier, c'est-à-dire que tous les mots sont reconnus !

Proposition. *Le langage reconnu par l'oracle des suffixes complété par des ε–transitions correspondant aux liens suffixiels est le monoïde libre tout entier* Σ^*.

Dem. Toute lettre apparaissant dans le mot est le début d'un suffixe. Comme l'oracle reconnaît tous les suffixes, on peut lire cette lettre à partir de l'état initial vers un état *p*.

D'autre part, tout état (excepté 0) a un lien suffixiel vers un état plus petit. Donc, on peut revenir à 0 à partir de chaque état *p* en suivant les liens suffixiels. Ainsi, on peut boucler sur l'état initial en lisant chaque lettre.

Attention : Ça ne veut pas dire que tous les mots de Σ^* ont la même probabilité de sortir lorsqu'on parcourt l'oracle aléatoirement. C'est **l'aspect probabiliste** qui justifie l'utilisation musicale de l'oracle.

1.3 Définition des langages réguliers et propriétés de clôture

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

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

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

Construction d'un AFN avec ε–transitions pour l'étoile L^* de L (cours Berstel 2004, p. 27) :

on ajoute un état supplémentaire, et des ε–transitions

- de cet état vers lui-même,
- de cet état vers tous les états initiaux de l'AFN pour L,
- de tous les états finals de l'AFN vers l'état ajouté.

Propriété. *Si L est un langage régulier, alors l'étoile L* de L est aussi régulier.*

Par des constructions analogues, on obtient des automates pour les opérations suivantes sur les langages réguliers :

- union (on fait l'union des automates),
- intersection (on utilise le produit cartésien des états),
- concaténation.

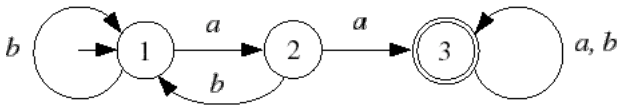
Rappelons aussi qu'on a vu la construction d'un AFD pour le complémentaire d'un langage régulier.

Théorème. *L'ensemble des langages réguliers de* Σ^* *est fermé par les opérations suivantes :*

- *opérations ensemblistes (union, intersection, complémentaire),*
- *produit (concaténation), étoile.*

2. Reconnaissabilité dans un monoïde quelconque

2.1 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.

Pour les mots de longueur 2 :

• $b^2 = b$.

• il y a un élément *absorbant* vérifiant $ox = xo = o$ pour tout x. Lequel ?

-> $o = a^2$ est absorbant, car il envoie tous les états sur 3.

Notons qu'il ne peut exister deux éléments absorbants distincts, car $o_1 = o_1 o_2 = o_2$.

Pour les mots de longueur 3, on en déduit des égalités lorsqu'ils ont un carré a^2 ou b^2 :

$a^3 = a^2 b = ba^2 = a^2$,

$b^3 = b$,

$ab^2 = ab$,

$b^2 a = ba$.

Restent deux mots sans carré, qui vérifient les égalités vues plus haut : $aba = a$, $bab = b$.

Finalement, les huit mots de longueur 3 ne donnent pas de relation nouvelle sur les états.

Les mots sur *a, b* définissent donc cinq relations distinctes sur les états de l'automate. L'ensemble des relations sur les états {1, 2, 3} forme un monoïde fini. Les mots sur *a, b* engendrent un sous-monoïde qui contient six éléments :

• les cinq précédents,

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

Le sous-monoïde est donc égal à $\{Id, a, b, a^2, ab, ba\}$ et voici sa table :

	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 exactement ceux qui définissent une relation sur les états égale à a^2.

Soit l'application φ qui a un mot associe la relation correspondante sur les états $\{1, 2, 3\}$. C'est un morphisme du monoïde libre Σ^* (infini) dans le monoïde (fini) des relations entre états. Un mot w est reconnu par l'automate si et seulement si $\varphi(w) = a^2$.

Donc le langage reconnu par l'automate est exactement l'ensemble $\varphi^{-1}(a^2)$. 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 peut retrouver simplement les propriétés de clôture par les opérations ensemblistes vues plus haut avec les AFN grâce à un lemme de la théorie des ensembles :

Lemme. *Si φ est une application de A dans B, et X et Y des parties de B, on a :*

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

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

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

Exemple : Soient les ensembles $A = \{1, 2, 3, 4, 5\}$, $B = \{6, 7, 8, 9\}$, la fonction φ définie par : $\varphi(1) = \varphi(2) = 6$, $\varphi(4) = 8$, $\varphi(5) = 9$, et soient $X = \{6, 7\}$, $Y = \{7, 8\}$.

On a :

• $\varphi^{-1}(X \cup Y) = \varphi^{-1}(\{6, 7, 8\}) = \{1, 2, 4\} = \{1, 2\} \cup \{4\} = \varphi^{-1}(X) \cup \varphi^{-1}(Y)$,

• $\varphi^{-1}(X \cap Y) = \varphi^{-1}(\{7\}) = \varnothing = \{1, 2\} \cap \{4\} = \varphi^{-1}(X) \cap \varphi^{-1}(Y)$,

• $\varphi^{-1}(X \setminus Y) = \varphi^{-1}(\{6\}) = \{1, 2\} = \{1, 2\} \setminus \{4\} = \varphi^{-1}(X) \setminus \varphi^{-1}(Y)$.

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

• $\varphi(\{1\} \cap \{2\}) = \varphi(\varnothing) = \varnothing$, mais $\varphi(\{1\}) = \varphi(\{2\}) = \{6\}$ donc $\varphi(\{1\}) \cap \varphi(\{2\}) = \{6\}$.

On en déduit :

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

Dem. Pour le complément, soit $X \in \mathrm{Rec}(M)$. On a un morphisme $\varphi : M \to N$ (que l'on peut toujours supposer surjectif) et Z inclus dans N tel que $X = \varphi^{-1}(Z)$. D'où :

• $M \setminus X = \varphi^{-1}(N) \setminus \varphi^{-1}(Z) = \varphi^{-1}(N \setminus Z)$.

Pour l'union et l'intersection, soient X_1 et $X_2 \in \mathrm{Rec}(M)$. On a deux morphismes

$\varphi_1 : M \to N_1$ et Z_1 inclus dans N_1 avec $X_1 = \varphi_1^{-1}(Z_1)$,

$\varphi_2 : M \to N_2$ et Z_2 inclus dans N_2 avec $X_2 = \varphi_2^{-1}(Z_2)$.

Pour appliquer la formule ensembliste ci-dessus, il faut que X_1 et X_2 soient images réciproques par un morphisme unique. L'astuce est d'utiliser le monoïde produit $N_1 \times N_2$ avec le morphisme $\varphi(u) = (\varphi_1(u), \varphi_2(u))$. Cela donne :

$X_1 = \varphi_1^{-1}(Z_1) = \varphi^{-1}(Z_1 \times N_2)$,

$X_2 = \varphi_2^{-1}(Z_2) = \varphi^{-1}(N_1 \times Z_2)$.

Donc :

• $X_1 \cup X_2 = \varphi_1^{-1}(Z_1) \cup \varphi_2^{-1}(Z_2) = \varphi^{-1}(Z_1 \times N_2) \cup \varphi^{-1}(N_1 \times Z_2) = \varphi^{-1}((Z_1 \times N_2) \cup (N_1 \times Z_2))$,

• $X_1 \cap X_2 = \varphi_1^{-1}(Z_1) \cap \varphi_2^{-1}(Z_2) = \varphi^{-1}(Z_1 \times N_2) \cap \varphi^{-1}(N_1 \times Z_2) = \varphi^{-1}((Z_1 \times N_2) \cap (N_1 \times Z_2)) = \varphi^{-1}(Z_1 \times Z_2)$,

Remarque : Dans un monoïde quelconque M,

• on n'a pas la fermeture de $\mathrm{Rec}(M)$ par **produit et étoile**,

• $\mathrm{Rec}(M)$ ne contient pas nécessairement les **parties finies**.

Par exemple, dans le monoïde libre Σ^*, un singleton $\{x\}$ est reconnaissable (cf. l'automate appelé « écorché de x »).

Alors que dans un groupe infini M, les singletons $\{x\}$ ne sont pas reconnaissables.

En effet, la partie Z telle que $\{x\} = \varphi^{-1}(Z)$ est réduite à l'élément $z = \varphi(x)$.

Si N est fini, on a $\mathrm{card}(M) > \mathrm{card}(N)$, donc φ n'est pas injectif. Dans un groupe, cela implique qu'il existe $y \in \ker(\varphi)$ différent de l'élément neutre.

D'où $\varphi(xy) = \varphi(x)\varphi(y) = \varphi(x) = z$, donc $\varphi^{-1}(z)$ contient $\{x, xy\} \neq \{x\}$.

Le fait qu'on perde les propriétés de clôture par produit et étoile conduit à introduire une autre famille notée Rat(*M*) du monoïde quelconque *M*, qui vérifie précisément la clôture par produit et étoile.

2.2 Parties rationnelles et théorème de Kleene

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.*

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

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

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).*

3. Notion de grammaire formelle

Définition. *Une grammaire est définie par la donnée (T, N, P, S)*

- *de deux ensembles disjoints de symboles, les terminaux T que l'on note par des minuscules et les non terminaux N notés par des majuscules,* $\Sigma = N \cup T$*,*
- *d'un ensemble fini P de productions (ou règles de réécriture) de la forme* $u \rightarrow v$ *où u et v sont des séquences de symboles de N ou T,*
- *d'un symbole particulier S non terminal appelé axiome.*

Si $u \rightarrow v$ est une production de la grammaire, la séquence *xuy* peut se dériver en *xvy*. Cela signifie que le fragment *u* peut être remplacé par *v* à l'intérieur des séquences dans lesquelles il apparaît.

Le langage engendré par la grammaire est l'ensemble de toutes les séquences de symboles ***terminaux*** que l'on peut obtenir par dérivation à partir de l'axiome, en utilisant les productions. Les non terminaux apparaissent donc comme des variables intermédiaires qu'il faut éliminer pour obtenir les mots engendrés par la grammaire.

La hiérarchie de Chomsky distingue quatre classes de grammaires formelles, selon la forme de leurs productions :

• Type 0 : aucune restriction.

• Type 1 (contextuelles ou context-sensitive en anglais) : productions ayant au plus **un non terminal** à gauche, c'est-à-dire de la forme $uAv \rightarrow uwv$ où

- A est un non terminal,
- u, v des séquences de terminaux, et
- w une séquence quelconque avec au moins un symbole.

• Type 2 (algébriques ou context-free en anglais) : productions ayant un symbole unique à gauche (non terminal), c'est-à-dire de la forme $A \rightarrow w$ où

- A est un non terminal, et
- w une séquence ***quelconque*** (terminaux ou non terminaux) avec au moins un symbole.

• Type 3 (régulières ou linéaires) : productions ayant un symbole unique à gauche (non terminal) et au plus un non terminal à droite situé à la fin, c'est-à-dire de la forme $A \rightarrow wB$ ou $A \rightarrow w$, où

- A et B sont des non terminaux, et
- w une séquence de ***terminaux exclusivement*** avec au moins un symbole

(donc la partie droite ne peut contenir au plus qu'un seul non terminal).

Les types sont emboîtés les uns dans les autres, chacun étant un cas particulier du type précédent.

Les grammaires de type 1 sont dites « contextuelles », car les séquences u et v jouent le rôle de contextes dans lesquels on peut remplacer le symbole non terminal A par la séquence w. Notons que la condition sur le nombre de symboles de w interdit toute production dans laquelle le membre de droite serait plus court que le membre de gauche.

Les grammaires algébriques (type 2) sont caractérisées par le fait que le membre gauche des productions est réduit à un seul symbole (non terminal). Les grammaires régulières (type 3) sont des cas particuliers de grammaires de type 2, avec une restriction imposée au membre de droite w, qui ne peut contenir qu'un seul symbole non terminal B situé à la fin.

Théorème. *Les langages réguliers (reconnus par AFD ou AFN) sont exactement les langages engendrés par des grammaires de type 3 (régulières).*

Remarque : Un même langage peut être décrit par **plusieurs grammaires de types différents**. Par définition, il existe une grammaire correspondant à son type (0, 1, 2 ou 3), mais il peut en exister une autre d'un type plus général (numéro inférieur dans la numérotation). Il arrive qu'elle soit plus commode pour caractériser le langage.

Exemple : Soit la grammaire

$S \rightarrow AB$,

$A \rightarrow ab$,

$B \rightarrow abb$.

Quel est le langage engendré ?

-> $L = \{ababb\}$ réduit à une séquence unique.

Quel est le type de la grammaire ?

-> type 2 algébrique, sa première production contenant deux non terminaux en partie droite.

Pourtant, L est un langage fini, donc de type régulier.

Grammaire régulière engendrant L ?

$S \rightarrow abB$,

$B \rightarrow abb$.

Les langages de type algébrique sont caractérisés par des phénomènes de récursion infinie :

$S \rightarrow aSb$,

$S \rightarrow ab$,

qui engendre le langage $L' = \{ab, aabb, aaabbb, \text{etc.}\}$.

Les langages de type régulier ne comportent pas ce phénomène. Donc L' ne peut pas être engendré par une grammaire régulière.

Références

• références générales

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

Gross M., Lentin A., *Notions sur les grammaires formelles*, Paris, Gauthier-Villars, 1967.

• grammaire de substitution harmonique en jazz

Steedman, Mark, A Generative Grammar for Jazz Chord Sequences, *Music Perception*, vol. 2 (1) (1984) 52-77.

Steedman, Mark, The Blues and the Abstract Truth: Music and Mental Models, A. Garnham, J. Oakhill, (eds.), *Mental Models In Cognitive Science*, Mahwah, NJ: Erlbaum 1996, p. 305-318 (PostScript).

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

Chemillier M., Toward a formal study of jazz chord sequences generated by Steedman's grammar, *Soft Computing*, special issue on Formal Systems and Music, G. Assayag, V. Cafagna, M. Chemillier (eds.) **8** (9) (2004) 617-622.