MMIM Modèles mathématiques en informatique musicale

Marc Chemillier

Master Atiam (Ircam), 2008-2009

Notions théoriques sur les langages formels (Partie II)

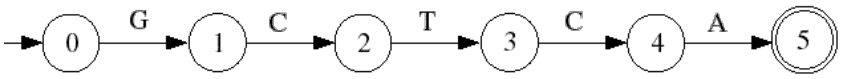
- Oracle des suffixes
- Notion de grammaire formelle
- Reconnaissabilité dans un monoïde quelconque

1. 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 déterminisation), 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 = 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.

Remarque : Cette fonction de lien suffixiel est inspirée d'algorithmes classiques de « pattern matching » qui utilisent une fonction appelée « fonction de saut » (algorithme de Morris et Pratt).

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 $p+1 = \delta(p, a)$.

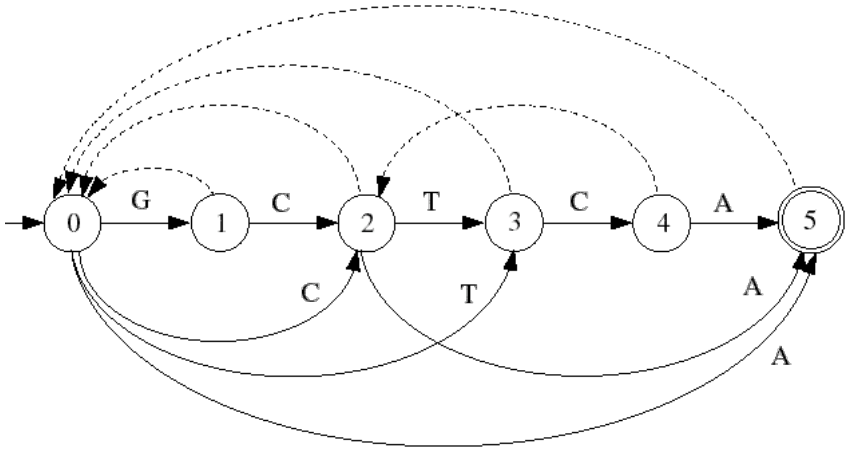
Pour rajouter les transitions, on suit les liens suffixiels déjà existants $f(p)$, $f(f(p))$, etc.

• 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

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 que les liens suffixiels sont des ε-transitions de l'automate, donc de les utiliser en génération.

2. 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*** de terminaux ou non terminaux,

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

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).*

Si l'on désigne par $\mathscr{L}_0$, $\mathscr{L}_1$, $\mathscr{L}_2$, $\mathscr{L}_3$ les **ensembles de langages** engendrés respectivement par les grammaires de types 0, 1, 2, 3, on a la suite d'inclusions suivante :

$$\mathscr{L}_0 \supset \mathscr{L}_1 \supset \mathscr{L}_2 \supset \mathscr{L}_3.$$

Ainsi, tout langage régulier (appartenant à $\mathscr{L}_3$) est aussi un langage algébrique (appartenant à $\mathscr{L}_2$), car les grammaires régulières sont des cas particuliers de grammaires algébriques. Lorsqu'on parle du « type » d'un langage, on désigne **le plus petit ensemble** qui contient ce langage, dans la suite d'inclusions ci-dessus. Ce type correspond à la grammaire la plus simple permettant de décrire ce langage.

Les ensembles étant emboîtés les uns dans les autres, un même langage peut être décrit par **plusieurs grammaires de types différents**, qui ne correspondent pas toutes au type du langage lui-même.

Exemple : Les ensembles finis de séquences sont des langages de type régulier, comme le langage $\{ababb\}$ réduit à une séquence unique. Pourtant, ce langage peut être engendré par la grammaire $S \rightarrow AB$, $A \rightarrow ab$, $B \rightarrow abb$ qui est algébrique, sa première production contenant deux non terminaux en partie droite. Mais la grammaire régulière $S \rightarrow abB$, $B \rightarrow abb$ engendre le même langage, qui peut donc être engendré par deux grammaires de types différents.

Les langages de type algébrique sont caractérisés par des phénomènes de récursion infinie, comme le langage $\{ab, aabb, aaabbb,$ etc.$\}$ engendré par la grammaire $S \rightarrow aSb$, $S \rightarrow ab$. Les langages de type régulier ne comportent pas ce phénomène, et c'est la raison pour laquelle la grammaire algébrique précédente peut être remplacée par une grammaire régulière.

Lemme d'itération (langages algébriques). *Si L est un langage algébrique, il existe un entier n tel que pour tout mot w de L de longueur* $|w| \geq n$*, il existe une factorisation telle que :*

$w = xuyvz$ *avec* $|uv| > 0$ *et* $|uyv| \leq n$ *et* $\forall k,\ xu^k yv^k z$ *est dans L.*

Exemple : Langage des carrés

Soit le langage $D = \{uu,\ u \in \Sigma^*\}$ avec $\Sigma = \{a, b\}$.

Le langage des carrés D n'est pas algébrique.

En effet, soit le mot $w = a^n b^n a^n b^n$ qui est un carré.

D'après la factorisation $w = xuyvz$ du lemme d'itération, on a $|uyv| \leq n$ et $xuuyvvz$ est un carré.

• Il n'est pas possible que uyv soit inclus dans une puissance d'une des lettres a^n ou b^n, car en doublant u et v, ce ne serait plus un carré.

• Pour la même raison, il n'est pas possible que u et v soient inclus respectivement dans deux puissances de lettre différentes et consécutives.

• Enfin, il n'est pas possible que l'un soit à cheval sur deux puissances différentes, car comme $|uyv| \leq n$, l'autre est alors inclus dans une puissance, donc le redoublement crée une dissymétrie et ce n'est plus un carré.

Attention : le langage des palindromes (pairs) est algébrique.

Si u est un mot, on note $u\sim$ le *miroir* de u, c'est-à-dire le mot lu à l'envers (*cba* miroir de *abc*).

Soit le langage $P = \{uu\sim,\ u \in \Sigma^*\}$ avec $\Sigma = \{a, b\}$.

Le langage P est algébrique. Il est engendré par la grammaire :

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow \varepsilon$

Que dire du langage des palindromes impairs ?

$S \rightarrow a$

$S \rightarrow b$

Application musicale : Langage des *bols* des joueurs de *tabla* d'Inde du Nord

• Shakti, La danse du bonheur, *A Handful Of Beauty* (Columbia, 1977).

Le terme « *bol* » désigne les onomatopées utilisées par les percussionnistes du Nord de l'Inde, *dha, dhin, ghe, te, ke*, etc. pour identifier les diverses frappes des *tabla*.

L'ethnomusicologue Jim Kippen a étudié les règles implicites d'agencement de ces *bols* utilisées par les musiciens. Avec l'informaticien Bernard Bel, ils ont réalisé le logiciel Bol Processor permettant de saisir au clavier des séquences de *bols* à la vitesse où elles sont interprétées (jusqu'à environ six *bol*s par seconde) et de vérifier si la séquence est correcte ou non au moyen d'une grammaire formelle.

Voici une séquence de *bols* constituée d'un motif de 4 temps et de ses 3 variations :

dhatidhage	*nadhatrkt*	*dhatidhage*	*dheenagena*
dhatrktdha	*tidhagena*	*dhatidhage*	*teenakena*
tatitake	*natatrkt*	*tatitake*	*teenakena*
dhatrktdha	*tidhagena*	*dhatidhage*	*dheenagena*

Règles implicites de variation s'appliquant dans cette séquence ?

- répétition,
- permutation,
- substitution de *bols*
- dépendance entre parties de la séquence

- ligne 2 : permutation de la ligne 1 (début), substitution partielle *ge/ke, dhee/tee* (fin),

- ligne 3 : dépendance avec la ligne 1, substitution *dha/ta, ge/ke, dhee/tee*,
- ligne 4 : dépendance avec le début de la ligne 2 (même permutation)

(=*dhatidhage* *nadhatrkt* *dhatidhage* *dheenagena*)
(=*dhatrktdha* *tidhagena*) (=*dhatidhage* *teenakena*)
*(:*tatitake* *natatrkt* *tatitake* *teenakena*)
(:*dhatrktdha* *tidhagena*) (=*dhatidhage* *dheenagena*)

• on note les non terminaux avec un signe (=) , et on utilise le signe (:) pour marquer une dépendance, c'est-à-dire que dans une règle de la forme $A \rightarrow (=B)\ (:B)$, les deux dérivations ultérieures de *B* doivent être des copies identiques,

• *V* indique une permutation de *bols*, et * indique un substitution complète.

$S \rightarrow (=A_{16})\ (=V_8)\ (=A'_8)\ *(:A_{16})\ (:V_8)\ (=A_8)$
$A_{16} \rightarrow$ *dhatidhage* *nadhatrkt* *dhatidhage* *dheenagena*
$A_8 \rightarrow$ *dhatidhage* *dheenagena*
$A'_8 \rightarrow$ *dhatidhage* *teenakena*

Remarque : La règle $A \rightarrow (=B)\ (:B)$ est algébrique (symbole unique non terminal à gauche), mais le mécanisme d'application de cette règle n'est pas celui donné dans la définition générale (car il introduit des dépendances). Cela modifie **la nature du langage engendré**. Problème : on ne peut plus baser son étude sur celle des règles uniquement.

Exemple : Supposons que *B* donne par dérivation tous les mots de Σ^* sur $\Sigma = \{a, b\}$.

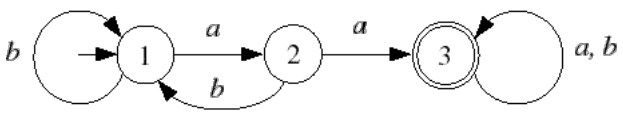
Une grammaire régulière peut réaliser cette dérivation, et en ajoutant la règle ci-dessus, on obtient une grammaire algébrique.

Mais le langage engendré est le langage des carrés $D = \{uu,\ u \in \Sigma^*\}$.

Or on a vu que *D* n'est pas algébrique.

3. Reconnaissabilité dans un monoïde quelconque

3.1 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 : la lettre a met en relation q_1 avec 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, c'est-à-dire que a^2 met en relation q_1 avec q_2 ssi $q_2 \in \delta(q_1, aa)$.

Exemple : On complète la table de transition de l'automate ci-dessus en indiquant les états obtenus pour les mots bb et aba. En comparant les relations obtenues, on en déduit des égalités vérifiées par ces relations.

	1	2	3	
a	2	3	3	
b	1	1	3	
b^2	1	1	3	$= b$
aba	2	3	3	$= a$

L'ensemble des relations sur les états {1, 2, 3} forme un monoïde fini. Si l'on complète la table de transition ci-dessus avec tous les mots de longueur 2, 3, 4 etc., on constate que cela ne donne que cinq relations distinctes. Les relations associées aux mots sur a, b forment donc un sous-monoïde, qui contient six éléments : les cinq précédents et la relation identité Id (chaque état 1, 2, 3 est envoyé sur lui-même) qui est l'élément neutre. Le sous-monoïde est donc $\{Id, a, b, a^2, ab, ba\}$ et 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

Les mots reconnus par l'automate sont exactement ceux qui se réduisent à a^2 dans le monoïde des relations.

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

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

On utilise le monoïde produit $N_1 \times N_2$ et on définit un morphisme $\varphi(u) = (\varphi_1(u), \varphi_2(u))$.

On a :

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

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

Donc :

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

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

De même, pour $X \in \mathrm{Rec}(M)$, 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 :

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

Remarque : Dans un monoïde quelconque M, on n'a pas la fermeture de $\mathrm{Rec}(M)$ par produit et étoile.

3.2 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).*

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.

• oracle des facteurs (simulation stylistique)

Allauzen, Cyril & Maxime Crochemore, Mathieu Raffinot, Factor oracle : A new structure for pattern matching, *SOFSEM '99*: *Proceedings of the 26th Conference on Current Trends in Theory and Practice of Informatics*, Lecture Notes in Computer Science, Springer-Verlag, 1999, p. 291-306 (présentation).

Mancheron Alban , Moan Christophe, Combinatorial Characterization of the Language Recognized by Factor and Suffix Oracles, *International Journal of Foundations of Computer Science*, 16 (6) (2005) 1179-1191.

Assayag, Gérard, Shlomo Dubnov, Olivier Delerue, Guessing the Composer's Mind: Applying Universal Prediction to Musical Style, *Proceedings of the ICMC (Int. Computer Music Conf.)*, 1999, p. 496-499 (pdf).

Assayag, Gérard, Shlomo Dubnov, Using factor oracles for machine improvisation, *Soft Computing*, special issue on Formal Systems and Music, G. Assayag, V. Cafagna, M. Chemillier (eds.), 8 (9) (2004) 604–610 (pdf).

• 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.

• grammaire des *bols* en musique indienne

Kippen Jim, Bel Bernard, Modelling music with grammars: formal language representation in the Bol Processor, Alan Marsden & Anthony Pople (eds.), *Computer Representations and Models in Music*, London, Academic Press, 1992, p. 207-238 (pdf).