MMIM Modèles mathématiques

en informatique musicale

Marc Chemillier

Master M2 Atiam (Ircam), 2011-2012

Reconnaissabilité et rationalité dans un monoïde quelconque

- Automate d'un langage et équivalences associées
  - Modèle du jeu de dés
  - Equivalences par les bonnes finales et les bons contextes
- Transductions d'un langage
  - Reconnaissabilité dans un monoïde quelconque
  - Rationalité et théorème de Kleene

1. Automate d'un langage et équivalences associées

1.1 Modèle du jeu de dés

Langage $L = \{abc, bbb, abb, bbc\}$

Ce langage a-t-il une structure de « jeu de dés » avec des <u>éléments interchangeables indépendamment de ce qui suit et ce qui précède</u> ?

Le langage L est <u>fini</u>, donc reconnaissable par automate. On obtient facilement un AFN en prenant les écorchés de tous les mots de L.

Problème : <u>**l'union des écorchés ne dit rien sur l'existence d'éléments interchangeables au sein des mots de *L*.**</u>

<u>Automate **canonique** associé au langage</u> :

$a^{-1}L = \{bc, bb\}$

$b^{-1}L = \{bb, bc\} = a^{-1}L$

$c^{-1}L = \varnothing$

$b^{-1}\{bb, bc\} = \{b, c\}$

$b^{-1}\{b, c\} = \{\varepsilon\}$

$c^{-1}\{b, c\} = \{\varepsilon\}$

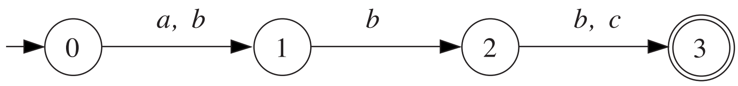
états de l'automate :

$0 = L$

$1 = \{bc, bb\}$

$2 = \{b, c\}$

$3 = \{\varepsilon\}$



-> on s'aperçoit que c'est un « jeu de dés de Mozart », c'est-à-dire que ce langage se **<u>factorise</u>** :

$L = \{a, b\}\{b\}\{b, c\}$

1.2 Equivalences par les bonnes finales et les bons contextes

L un langage de Σ^*, u un mot quelconque de Σ^*

$u^{-1}L = \{y \in \Sigma^*, uy \in L\}$

-> terminaisons possibles de u pour obtenir un mot de L (« bonnes finales »)

Equivalence <u>par les bonnes finales</u> :

$u \approx v$ si et seulement s'ils ont les mêmes bonne finales : $u^{-1}L = v^{-1}L$

Autre définition qui revient au même :

$u \approx v$ si et seulement si pour tout mot y de Σ^*, $uy \in L \Leftrightarrow vy \in L$

On voit que $u \in L$ si et seulement si le mot vide ε appartient à $u^{-1}L$. Donc si $u \in L$, tous les mots de la classe de u pour l'équivalence par les bonnes finales sont aussi dans L.

Proposition ([Gross & Lentin 1967], p. 144). *Le langage L est une union de classes pour $\approx$.*

Théorème ([Gross & Lentin 1967], p. 145). *La relation d'équivalence <u>par les bonnes finales</u> relativement à un langage L est la <u>moins fine</u> des relations d'équivalence compatibles à droite avec la concaténation et telles que L soit une union de classes.*

$\mathscr{R}$ relation d'équivalence, que veut dire « $\mathscr{R}$ est <u>plus fine</u> que $\approx$ » ?

-> en tant que partie de $\Sigma^* \times \Sigma^*$, $\mathscr{R}$ est incluse dans $\approx$ (rend équivalents moins d'éléments)

-> le nombre de classes de $\mathscr{R}$ est <u>supérieur</u> au nombre de classes de $\approx$

Si la relation d'équivalence par les bonnes finales $\approx$ n'a qu'un <u>nombre fini de classes</u>, alors il n'existe qu'un <u>**nombre fini d'ensembles $u^{-1}L$**</u> quand u décrit Σ^*. Dans ce cas, on peut construire un <u>automate fini reconnaissant L</u> comme on a vu pour le jeu de dés. L'état initial est l'ensemble L lui-même, et les états terminaux sont les ensembles contenant le mot vide ε.

Inversement, si L est reconnu par un automate fini, les états définissent les ensembles $u^{-1}L$.

Equivalence <u>par les bons contextes</u> :

$u \equiv v$ si et seulement si pour tous mots x, y de Σ^*, $xuy \in L \Leftrightarrow xvy \in L$

On a évidemment : $u \equiv v$ (bons contextes) implique $u \approx v$ (bonnes finales)

-> en tant que partie de $\Sigma^* \times \Sigma^*$, $\equiv$ est incluse dans $\approx$ (rend équivalents moins d'éléments)

-> le nombre de classes de $\equiv$ est <u>supérieur</u> au nombre de classes de $\approx$

Si l'équivalence par les bons contextes $\equiv$ n'a qu'un <u>nombre fini de classes</u>, il est en de même de l'équivalence par les bonnes finales $\approx$, donc on peut construire un automate reconnaissant L.

Inversement, si L est reconnu par un automate fini, les couples d'états définissent un nombre fini de bons contextes possibles.

<u>compatibilité avec la concaténation de l'équivalence par les bons contextes</u> :

$u \equiv u'$ et $v \equiv v'$ impliquent $uv \equiv u'v'$

En effet :

$x(uv)y \in L \Leftrightarrow x(u)vy \in L \Leftrightarrow x(u')vy \in L \Leftrightarrow xu'(v)y \in L \Leftrightarrow xu'(v')y \in L \Leftrightarrow x(u'v')y \in L$

-> on peut définir une opération sur l'ensemble des classes :

$[u][v] = [uv]$

L'ensemble des classes est un <u>monoïde quotient</u>.

Exemple : $L = \{abc, bbb, abb, bbc\} = \{a, b\}\{b\}\{b, c\}$

Seuls les mots facteurs d'un mot de L ont un ensemble de « bons contextes » non vide. Le mot vide ε est toujours facteur de n'importe quel mot de L :

ε -> bons contextes = $\{(\varepsilon, abc), (\varepsilon, bbb), (\varepsilon, abb), (\varepsilon, bbc), (a, bc), (b, bb), (a, bb), (b, bc), (ab, c), (bb, b), (ab, b), (bb, c), (abc, \varepsilon), (bbb, \varepsilon), (abb, \varepsilon), (bbc, \varepsilon)\}$

a -> bons contextes = $\{(\varepsilon, bc), (\varepsilon, bb)\}$

b -> bons contextes = $\{(\varepsilon, bc), (\varepsilon, bb), (b, b), (b, c), (a, c), (a, b)\}$

c -> bons contextes = $\{(ab, \varepsilon), (bb, \varepsilon)\}$

ab -> bons contextes = $\{(\varepsilon, c), (\varepsilon, b)\}$

bb -> bons contextes = $\{(a, \varepsilon), (\varepsilon, c), (\varepsilon, b)\}$

bc -> bons contextes = $\{(a, \varepsilon), (b, \varepsilon)\}$

mots de L : abc, bbb, abb, bbc -> bons contextes = $\{(\varepsilon, \varepsilon)\}$

tous les autres : G = ensemble des mots **<u>non facteurs</u>** de mots de L -> bons contextes = $\varnothing$

-> d'où neuf classes distinctes : $[\varepsilon]$, $[a]$, $[b]$, $[c]$, $[ab]$, $[bb]$, $[bc]$, $[abc] = L$, G

	$[\varepsilon]$	$[a]$	$[b]$	$[c]$	$[ab]$	$[bb]$	$[bc]$	L	G
$[\varepsilon]$	$[\varepsilon]$	$[a]$	$[b]$	$[c]$	$[ab]$	$[bb]$	$[bc]$	L	G
$[a]$	$[a]$	G	$[ab]$	G	G	L	L	G	G
$[b]$	$[b]$	G	$[bb]$	$[bc]$	G	L	L	G	G
$[c]$	$[c]$	G	G	G	G	G	G	G	G
$[ab]$	$[ab]$	G	L	L	G	G	G	G	G
$[bb]$	$[bb]$	G	L	L	G	G	G	G	G
$[bc]$	$[bc]$	G	G	G	G	G	G	G	G
L	L	G	G	G	G	G	G	G	G
G	G	G	G	G	G	G	G	G	G

φ <u>surjection canonique</u> de Σ^* dans le monoïde quotient : $\varphi(u) = [u]$

L s'identifie à la classe $[abc]$, c'est-à-dire $L = \varphi^{-1}([abc])$

Dans le cas général, L est une <u>union de classes</u>, c'est-à-dire qu'il existe une partie Z de l'ensemble des classes (monoïde quotient) telle que :

$L = \varphi^{-1}(Z)$.

2. Transduction d'un langage

2.1 Reconnaissabilité dans un monoïde quelconque

Un transducteur = **automate étiqueté par des couples** n'est pas tout à fait la même chose qu'un automate étiqueté par des lettres :

-> sur des couples, pas d'unicité de la décomposition : $(aaa, bc) = (a, b)(aa, c) = (aa, b)(a, c)$.

-> Σ^* ensemble des séquences de lettres : monoïde libre sur un alphabet fini Σ

$\neq$ $\Sigma_1^* \times \Sigma_2^*$ ensemble de couples de séquences …**monoïde quelconque** !

On peut étendre à un monoïde quelconque M la notion de « reconnaissabilité » en utilisant l'image réciproque d'un monoïde fini (dans le cas de Σ^* : monoïde quotient pour les « bons contextes »), mais cela pose quelques problèmes :

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.*

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

Propriétés de clôture par les opérations ensemblistes :

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

On retrouve les propriétés de clôtures par complément et union déjà vues directement sur les automates AFD et AFN. Sur les automates, on peut voir également :

• reconnaissabilité des parties finies (= automate obtenu avec les « écorchés » des mots)

• clôture par produit (connexion de deux automates) et étoile (boucle sur un automate).

Problème : Dans un monoïde quelconque M,

• Rec(M) ne contient pas nécessairement les **parties finies** :

Exemple : dans un groupe infini M, les singletons $\{x\}$ ne peuvent pas être reconnus par image inverse de morphisme dans un monoïde fini N. En effet, soit $z = \varphi(x)$. Si N est fini, on a

$\text{card}(M) > \text{card}(N)$, donc φ n'est pas injectif. Dans un groupe, cela implique que $\ker(\varphi)$ n'est pas réduit à l'élément neutre. Soit $y \in \ker(\varphi)$ différent de l'élément neutre. On a $\varphi(xy) = \varphi(x)\varphi(y) = \varphi(x) = z$, donc $\varphi^{-1}(z)$ contient $\{x, xy\} \neq \{x\}$.

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

Le fait qu'on perde les propriétés de clôture par produit et étoile conduit à introduire une autre famille notée $\text{Rat}(M)$ 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.*

Attention : Dans le **produit de monoïdes libres** $\Sigma_1^* \times \Sigma_2^*$, on a

$\text{Rec}(\Sigma_1^* \times \Sigma_2^*) \neq \text{Rat}(\Sigma_1^* \times \Sigma_2^*)$.

Une transduction rationnelle est un élément de $\text{Rat}(\Sigma_1^* \times \Sigma_2^*)$.

Un transducteur est un automate avec un seul état initial, dont les flèches sont étiquetées par des **couples de mots** de $\Sigma_1^* \times \Sigma_2^*$.

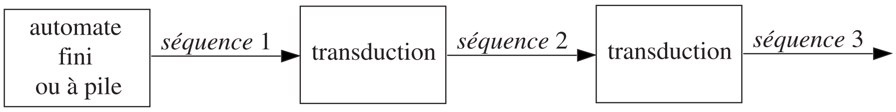
Théorème [Berstel, Théorème 6.1, p. 78]. *Une partie de $\Sigma_1^* \times \Sigma_2^*$ est une transduction rationnelle de $Rat(\Sigma_1^* \times \Sigma_2^*)$ si et seulement si elle est réalisée par un transducteur.*

Dem. C.N. Utilise le théorème de Nivat (voir le livre de Berstel).

C.S. Soit un transducteur d'état initial i et d'états finals F, avec un nombre fini de flèches. On considère l'ensemble E des flèches comme un alphabet et on forme le monoïde libre E^*. On remplace les étiquettes du transducteur par les flèches elles-mêmes. Le transducteur devient un automate sur l'alphabet E reconnaissant un langage régulier K de E^*. On considère le morphisme f de E^* dans $\Sigma_1^* \times \Sigma_2^*$ qui a une flèche associe son étiquette. La transduction est l'image directe $f(K)$ d'un langage régulier par un morphisme, donc c'est une partie rationnelle de $\Sigma_1^* \times \Sigma_2^*$.

Monoïde libre Σ^* (alphabet fini)	**Monoïde quelconque M**
langages reconnaissables $\mathrm{Rec}(\Sigma^*)$ = langages rationnels $\mathrm{Rat}(\Sigma^*)$ (Théorème de Kleene)	langages **reconnaissables** $\mathrm{Rec}(M)$ ≠ langages **rationnels** $\mathrm{Rat}(M)$
langages reconnaissables/rationnels $\mathrm{Rat}(\Sigma^*)$ = fermé par les opérations ensemblistes (union, complément, intersection) + produit, étoile	langages **reconnaissables** $\mathrm{Rec}(M)$ = fermé par les opérations ensemblistes (union, complément, intersection), langages **rationnels** $\mathrm{Rat}(M)$ = fermé par union, produit, étoile
langages reconnaissables/rationnels : image **directe et réciproque** par morphisme = reconnaissable/rationnel	langages **reconnaissables** $\mathrm{Rec}(M)$: image **réciproque** par morphisme = reconnaissable langages **rationnels** $\mathrm{Rat}(M)$: image **directe** par morphisme = rationnel
langages reconnaissables/rationnels : = reconnus par **automate fini**	transductions **rationnelles** $\mathrm{Rat}(\Sigma_1^* \times \Sigma_2^*)$ = réalisées par transducteur
langages reconnaissables/rationnels : image directe/réciproque par transduction rationnelle de $\mathrm{Rat}(\Sigma_1^* \times \Sigma_2^*)$ = reconnaissable/rationnel	langages **reconnaissables** image directe/réciproque par transduction rationnelle de $\mathrm{Rat}(M_1 \times M_2)$ = **rationnel**
morphisme = transduction rationnelle	**Contre-exemple** : alphabet infini
composition des transductions rationnelles de $\mathrm{Rat}(\Sigma_1^* \times \Sigma_2^*)$ et $\mathrm{Rat}(\Sigma_2^* \times \Sigma_3^*)$ = transduction rationnelle (Elgot & Mezei) [Berstel, Théorème 4.4, p. 68]	composition des transductions rationnelles de $\mathrm{Rat}(M_1 \times M_2)$ et $\mathrm{Rat}(M_2 \times M_3)$ = transduction rationnelle si le monoïde intermédiaire est **libre** $M_2 = \Sigma^*$

Les transductions rationnelles interviennent dans beaucoup de générateurs musicaux (Barbaud, Cope, ImproteK) où elles s'utilisent en cascade à partir d'une séquence de départ :



Dans le cas d'ImproteK :

• improvisation : génération d'une improvisation en suivant une grille donnée

• harmonisation : génération d'un chiffrage harmonique à partir d'une mélodie donnée

• arrangement : génération de voicings à partir d'un chiffrage harmonique

Sujets de recherche actuels :

• quand on ré-harmonise une improvisation, comparer le chiffrage harmonique obtenu avec le chiffrage de la grille d'origine

• exprimer les différences entre les deux chiffrages sous forme de « substitutions », c'est-à-dire de **règles de réécriture** (grammaire de Steedman)

Références

• équivalences liées à un langage

Gross, Maurice & André Lentin, *Notions sur les grammaires formelles*, Gauthier-Villars, 1967. Bibliothèques de la ville de Paris : Réserve centrale 160 GRO

PARIS6-BUPMC-Math Info Recherche : 4.83 GRO

• transductions, reconnaissabilité dans un monoïde quelconque

Berstel, Jean, *Transductions and context-free languages*, Teubner, 1979 (biblio).

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

http://ehess.modelisationsavoirs.fr/marc/publi/grammaires/grammaires.html

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