Ambiguïtés Irréductibles dans les Monoïdes de Mots

Claude Del Vigna Vincent Berment

Résumé

Le point de départ de l'étude présentée ici est cette "malice" de certaines langues du Sud-Est Asiatique qui s'écrivent sans que des espaces séparent les mots. Les traitements automatiques de ces langues s'en trouvent compliqués d'autant que, dès le premier niveau, celui des syllabes, le découpage des textes n'est en général pas unique. Autrement dit, rapporté à la combinatoire des mots, le système syllabique de ces langues n'est pas un code. On s'intéresse ici à l'origine des ambiguïtés de découpage, plus précisément au recensement de celles qu'on appelle irréductibles, en ce sens qu'elles sont à l'origine de toutes les autres. On montre que le langage des ambiguïtés irréductibles est rationnel et on présente le moyen d'en calculer une expression régulière en l'étayant de l'expérience de son application à la langue laotienne.

Abstract

The starting point of this study is that the written form of certain South-East Asian languages does not use spaces between words, thus complicating their automatic processing. This is particularly the case on the syllabic level, where generally the text cannot be cut up uniquely. From the formal combinatorics point of view, the syllabic system of these languages is not a code. We will focus on the origin of the splitting ambiguities, more specifically on the inventory of the so-called irreducible ambiguities, in the sense that all others originate from them. We prove that the language of irreducible ambiguities is rational. Then we present a method to compute one of its regular expressions, illustrating the method with the experience of its application to the Lao language.

1991 *Mathematics Subject Classification :* 68Q45, 68R15.

Introduction

ບ້ານຂ້ອຍມີໂຮງຮຽນຫຼັງນຶ່ງ

Dans mon village, il y a une école.

Parmi les langues de l'Asie du Sud-Est, vingt à trente d'entre elles s'écrivent sans que les mots soient séparés par des espaces. C'est le cas du khmer, du thaï, du laotien et du birman pour ne citer qu'elles. Le texte laotien ci-contre en est l'illustration. Le traitement automatique de ces langues se complique de cette caractéristique. Ainsi, dès le niveau syllabique, le découpage d'un texte n'est-il en général pas unique. Formellement, pour une langue donnée, l'ensemble des concaténations ambiguës de syllabes est un idéal[1] du monoïde sur les syllabes de cette langue. L'article établit que cet idéal est engendrable par un sous-langage rationnel, dit des ambiguïtés irréductibles, dont il propose une méthode pour en calculer une expression régulière. Aussi, la finitude de ce sous-langage est-elle une question décidable. En termes plus linguistiques, l'article montre la possibilité de recenser de façon exhaustive les suites ambiguës de syllabes et comment les construire toutes à partir des motifs de base que sont les ambiguïtés irréductibles.

Le travail présenté ici vient à la suite de la réalisation logicielle *LaoWord* (Berment, 1998, 2002), (*LaoWord*, 1998). *LaoWord*, qui s'inscrit dans ce qu'il est convenu d'appeler l'*informatisation des langues minoritaires* [2], est une librairie dynamique qui adapte le traitement de texte Word de Microsoft au laotien. Elle est fondée sur les syllabes en ce sens que lorsqu'on double-clique sur un caractère, la syllabe autour de ce caractère est sélectionnée. Lorsque plusieurs syllabes sont candidates, l'examen de la syllabe à droite et de celle à gauche suffit, en général, pour choisir l'une d'elles. C'est dans le but de cerner formellement cette technique heuristique, qu'on s'intéresse ici au recensement des ambiguïtés propres à un dictionnaire de syllabes.

L'étude est naturellement liée à la théorie des codes (Berstel & Perrin, 1985), (Lothaire, 2002). Mais "en creux", puisqu'aucun des dictionnaires des syllabes des langues concernées n'est un code et qu'on s'intéresse précisément ici aux raisons qui font qu'ils n'en sont pas. Ceci étant, la méthode présentée relève de la famille des algorithmes qui permettent de tester si un dictionnaire est ou n'est pas un code, dont celui de Sardinas et Patterson, 1953, (Berstel & Perrin, 1985), et celui de Spehner, 1975, (Lallement, 1979). On souligne, à propos du dernier, que la notion d'équation irréductible qu'il utilise correspond, au niveau découpage, à celle d'ambiguïté irréductible. On note enfin la parenté entre la notion de recouvrement sur laquelle se fonde la méthode présentée ici et celle de domino développée dans (Weber & Head, 1994) et (Head & Weber, 1995) pour les codes MSD[3].

[1] Un idéal d'un monoïde M est un sous-ensemble I de M tel que $MIM \subseteq I$ (Berstel & Perrin, 1985).

[2] Voir le site `http://isl.ntflex.uni-lj.si/SALTMIL`

[3] Un code MSD (*multiset decipherable*) est un dictionnaire W pour lequel toutes les factorisations d'un quelconque élément de W^+ en éléments de W conduisent au même tableau de fréquences.

1 Ambiguïtés irréductibles

Etant donné un alphabet fini V, on note V^* le monoïde libre sur V, ε la chaîne vide, V^+ l'ensemble $V^* - \{\varepsilon\}$, $|x|$ la longueur d'une chaîne x sur V et, si A et B sont des langages sur V, AB le langage produit (ou concaténation). On appelle *dictionnaire* tout sous-ensemble fini non vide W de V^+. Les éléments de W sont appelés *mots*. Ce sont les correspondants formels des syllabes.

Si $w = (w_1, w_2, ..., w_n)$, $n \geq 1$, est une suite non vide de mots, on note $\gamma(w)$ la concaténation $w_1 w_2 \ ... \ w_n$. De plus, si la suite w est vide, on pose $\gamma(w) = \varepsilon$. On désigne par W^* le sous-monoïde $\{\gamma(w) | w \text{ est une suite sur } W\}$ de V^*, dit *monoïde de mots*, et par W^+ la différence $W^* - \{\varepsilon\}$. Inversement, pour tout p dans W^*, une *W-factorisation* de p est une suite w sur W telle que $p = \gamma(w)$.

Un élément p de W^+ est *ambigu* s'il admet deux W-factorisations différentes. On note $A(W)$, plus simplement A, le sous-ensemble des éléments ambigus de W^+. Un élément p de A est dit *ambigu irréductible* s'il ne peut se factoriser d'aucune des manières $x\,p'\,y$, $x\,p'$ et $p'\,y$ dans lesquelles p' est dans A, x et y dans W^+. On note $AI(W)$, plus simplement AI, l'ensemble des éléments ambigus irréductibles.

LEMME 1.A – Etant donné un dictionnaire W sur un alphabet V, on a :

(1.1a) A est un idéal de W^*, i.e. $W^* A\, W^* \subseteq A$;

(1.1b) $A = W^* AI\, W^*$;

(1.1c) $AI = A - (W^+ A \cup A\, W^+ \cup W^+ A\, W^+)$. ∎

Exemple 1 - Le dictionnaire $W = \{cv, cvc, ccv, ccvc\}$ est celui des syllabes du laotien, rapportés aux deux caractères c (pour consonne) et v (pour voyelle) (Berment, 1997). L'ensemble AI correspondant est fini et égal à $\{cvccv, cvccvc, ccvccv, ccvccvc\}$ et toute concaténation ambiguë d'éléments de W contient nécessairement un de ces motifs.

Exemple 2 - Si $W = \{aa, aabb, bbc, cc, cdd, dd\}$, AI est infini, égal au langage rationnel que décrit l'expression régulière $aabbcc^*dd$.

2 Recouvrements et feuillures

Un *recouvrement* sur un dictionnaire W est un quadruplet (a, b, x, y) dans lequel a et b sont des suites non vides d'éléments de W, x et y des chaînes sur V et tel que

(2.1a) $x\gamma(b) = \gamma(a)y$

(2.1b) $|\gamma(a)| + |\gamma(b)| > |x| + |y|$.

L'inégalité stricte de la condition (2.1b) impose que a et b se chevauchent. Les chaînes x et y sont les *feuillures* du recouvrement, respectivement *gauche* et *droite*. La figure 1 est celle d'un recouvrement.

LEMME 2.A – Si deux recouvrements r et r' sur un dictionnaire W sont tels que la feuillure droite de r est égale à la feuillure gauche de r', i.e. si $r = (a, b, x, z)$ et $r' = (a', b', z, y)$, alors le quadruplet (aa', bb', x, y) est un recouvrement sur W. ∎

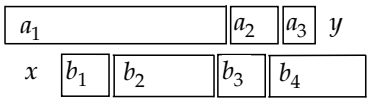

Figure 1 : recouvrement

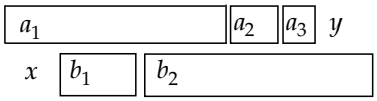

Figure 2 : 1-recouvrement

Eu égard au lemme précédent, deux recouvrements r et r' de la forme $r = (a, b, x, z)$ et $r' = (a', b', z, y)$, sont dits *emboîtables* et le recouvrement (aa', bb', x, y) est appelé leur *emboîtement*, noté $r \circ r'$. L'opération s'étend canoniquement à des ensembles de recouvrements.

On appelle 1-*recouvrement* tout recouvrement (a, b, x, y) dans lequel $a = (a_i\,; 1 \leq i \leq m)$ et $b = (b_j\,; 1 \leq j \leq n)$, et tel que

(2.2a) le quadruplet $((a_1), (b_n), x\,\gamma(b_1, ..., b_{n-1})), \gamma(a_2, ..., a_m)\,y)$ est un recouvrement,

(2.2b) $(n \neq 1) \vee (m \neq 1) \vee (xy \neq \varepsilon)$.

La condition (2.2a) impose que le premier élément de la suite a chevauche le dernier de la suite b, tandis que la disjonction (2.2b) interdit les quadruplets de la forme $((c), (c), \varepsilon, \varepsilon)$, $c \in W$, dans lesquels les feuillures sont vides et les deux suites sont égales et de longueur 1. La figure 2 est celle d'un 1-recouvrement. On note $R^1(W)$, ou R^1, l'ensemble des 1-recouvrements du dictionnaire W et F^1 l'ensemble de leurs feuillures. Lorsque $W = \{aa,\ aabb,\ bbc\,, cc\,, cdd,\ dd\}$, l'ensemble R^1 est celui de la figure 3 et $F^1 = \{\varepsilon,\ a,\ aa,\ aab,\ abb,\ bb,\ bc,\ c,\ cd,\ d,\ dd\}$.

LEMME 2.B – L'ensemble R1 des 1-recouvrements d'un dictionnaire W et celui F^1 de leurs feuillures sont finis. ■

((*aa*), (*aa*), *a*, *a*)	((*aabb*), (*bbc*), *aab*, *bc*)	((*bbc*), (*cdd*), *bb*, *dd*)	((*cc*, *dd*), (*cdd*), *c*, ∈)
((*aa*), (*aabb*), ∈, *bb*)	((*cc*), (*cc*), *c*, *c*)	((*cdd*), (*dd*), *c*, ∈)	((*bbc*, *dd*), (*cdd*), *bb*, ∈)
((*aa*), (*aabb*), *a*, *abb*)	((*cc*), (*cdd*), *c*, *dd*)	((*cdd*), (*dd*), *cd*, *d*)	((*aabb*), (*aa*, *bbc*), ∈, *c*)
((*aabb*), (*bbc*), *aa*, *c*)	((*bbc*), (*cc*), *bb*, *c*)	((*dd*), (*dd*), *d*, *d*)	

Figure 3 : 1-recouvrements de $W = \{aa, aabb, bbc, cc, cdd, dd\}$

3 Ambiguïtés irréductibles et 1-recouvrements

A partir de l'ensemble F^1 des feuillures d'un dictionnaire W, on considère l'ensemble $\widehat{F^1} = (F^1 - \{\varepsilon\}) \cup \{\varepsilon_g, \varepsilon_d\}$ obtenu en remplaçant la feuillure vide ε par deux éléments ε_g et ε_d n'appartenant pas à V^*, de façon à pouvoir distinguer dans la suite les cas où la feuillure vide est à gauche ou bien à droite. A l'ensemble R^1 des 1-recouvrements, on associe alors le plus petit sous-ensemble T du produit $\widehat{F^1} \times R^1 \times \widehat{F^1}$ tel que pour toutes suites non vides a et b de mots, pour tout $x \in F^1$, $x \neq \varepsilon$, et pour tout $y \in F^1$, $y \neq \varepsilon$,

(3.1a) $(a, b, x, y) \in R^1$, alors $(x, (a, b, x, y), y) \in T$;

(3.1b) $(a, b, x, \varepsilon) \in R^1$, alors $(x, (a, b, x, \varepsilon), \varepsilon_d) \in T$;

(3.1c) $(a, b, \varepsilon, y) \in R^1$, alors $(\varepsilon_g, (a, b, \varepsilon, y), y) \in T$ et $(\varepsilon_d, (a, b, \varepsilon, y), y) \in T$;

(3.1d) $(a, b, \varepsilon, \varepsilon) \in R^1$, alors $(\varepsilon_g, (a, b, \varepsilon, \varepsilon), \varepsilon_d) \in T$ et $(\varepsilon_d, (a, b, \varepsilon, \varepsilon), \varepsilon_d) \in T$.

Soit alors le quintuplet $(R^1, \widehat{F^1}, \varepsilon_g, T, \varepsilon_d)$. Puisque R^1 et F^1, donc $\widehat{F^1}$, sont finis (lemme 2.B), ce quintuplet est un automate d'états finis dont R^1 est l'alphabet, $\widehat{F^1}$ l'ensemble des états, ε_g l'état initial, T la relation de transition et ε_d l'état final. Soit L le langage reconnu. Chaque élément l de L est une suite non vide de 1-recouvrements qui, par construction, sont emboîtables dans l'ordre de la suite et dont l'emboîtement "tombe juste" à gauche et à droite, autrement dit, est de la forme $(a, b, \varepsilon, \varepsilon)$. Soit $\widehat{l}$ l'élément $\gamma(a)$, ou $\gamma(b)$, de W^+. On note K le langage $\{\widehat{l} | l \in L\}$ sur V.

On considère, par ailleurs, le quintuplet $(V, \widehat{F^1}, \varepsilon_g, T', \varepsilon_d)$ dans lequel T' est le sous-ensemble $\{(x, \gamma(b), y) | (x, (a, b, x, y), y) \in T\}$ du produit $\widehat{F^1} \times V^* \times \widehat{F^1}$. Ce quintuplet n'est pas *stricto sensu* un automate fini puisque T' n'est pas un sous-ensemble du produit $\widehat{F^1} \times V \times \widehat{F^1}$. Selon la méthode usuelle [4], on lui associe canoniquement un automate fini qu'on désigne par $AUT = (V, Q, \varepsilon_g, T'', \varepsilon_d)$. On fait remarquer que, par construction, l'automate $(R^1, \widehat{F^1}, \varepsilon_g, T, \varepsilon_d)$ est déterministe − il est aussi minimal − alors que l'automate AUT, qui en dérive, n'est, en général, ni l'un ni l'autre.

LEMME 3.A − Pour tout dictionnaire W sur un alphabet V,

(3.2a) $K \subseteq A$;

(3.2b) l'automate AUT reconnaît le langage K ;

(3.2c) K est un langage rationnel. ∎

Compte tenu de la propriété (3.2b), on notera AUT_K l'automate AUT.

LEMME 3.B − Pour tout dictionnaire W sur un alphabet V, $A = W^* K W^*$.

dem. [$\mathbf{A} \subseteq \mathbf{W^* K W^*}$] Si $a = (a_1, a_2, ..., a_m)$, $m \geq 1$, est une suite finie non vide sur W, on désigne par $\phi_a(k)$ l'indice dans $\gamma(a)$ du premier caractère de a_k, $1 \leq k \leq m$, par $\lambda_a(k)$ l'indice de son dernier caractère et par $a[i : j]$, $1 \leq i \leq j \leq m$, la sous-suite a_i, a_{i+1}, ... a_j de a. Enfin, si x est une chaîne sur V, $x[i \to j]$ désigne le facteur de x qui commence à l'indice i et se termine à l'indice j, $1 \leq i \leq j \leq |x|$.

Soit $p \in A$. D'après la propriété (1.1b), p se factorise en $p = x\, p'\, y$, $x \in W^*$, $p' \in AI$ et $y \in W^*$. Il suffit donc de montrer que $p' \in K$ pour établir l'inclusion $A \subseteq W^* K W^*$ visée. Puisque $p' \in AI$, il existe deux suites différentes sur W, $a = (a_1, a_2, ..., a_m)$, $m \geq 1$, et $b = (b_1, b_2, ..., b_n)$, $n \geq 1$, telles que $p' = \gamma(a) = \gamma(b)$ et $\forall i, 1 < i \leq m$, $\forall j, 1 < j \leq n$, $\phi_a(i) \neq \phi_b(j)$. On suppose que $\lambda_a(1) > \lambda_b(1)$ − voir figure 4. A partir des suites a et b, on considère l'ensemble S des couples (i, j), $1 \leq i \leq m$, $1 \leq j \leq n$, tels que $\phi_a(i) \leq \phi_b(j) \leq \lambda_a(i) \leq \lambda_b(j)$. Soit s le cardinal de S. Dans la figure 4,

[4] La méthode consiste à appliquer récursivement la règle : pour chaque triplet $(x, c\, p, y)$ de T', $c \in V$, $p \in V^*$, créer un nouvel état q et les deux triplets (x, c, q) et (q, p, y).

$S = \{(1,2),(3,4),(5,5)\}$ et ses éléments sont les segments obliques en trait double. On a $\forall (i,j) \in S,\ \forall (i',j') \in S,\ (i < i' \Leftrightarrow j < j')$, autrement dit les segments (i,j) ne "se croisent" jamais. On les numérote à partir de 1 en partant de la gauche et on note (i_k, j_k), $1 \leq k \leq s$, le k-ième. On a :

(1) $\forall k,\ 1 \leq k < s,\ \phi_a(i_{k+1}) \leq \lambda_b(j_k)$.

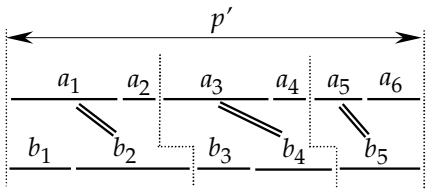

Figure 4 : lemme 3.B

En effet, dans le cas contraire, pour que la suite a "couvre" le caractère d'indice $\lambda_b(j_k)$, il doit exister $i, i_k < i < i_{k+1}$, tel que $\phi_a(i) < \lambda_b(j_k) < \lambda_a(i)$. Dès lors, pour tous j, $j_k < j < j_{k+1}$, soit $\lambda_b(j) < \lambda_a(i)$, soit $\phi_b(j) > \lambda_a(i)$, sinon le couple (i_{k+1}, j_{k+1}) ne serait pas le $(k+1)$-ième élément de S. Il en résulte que le caractère d'indice $\lambda_a(i)$ n'est pas "couvert" par la suite b. Or, par hypothèse, la suite b "couvre" toute la chaîne p'. D'où (1). Par ailleurs,

(2) $i_1 = 1$ et $j_s = n$.

On considère alors, d'une part, les sous-suites α_k, $1 \leq k \leq s$, de la suite a telles que $\alpha_k = a[i_k : i_{k+1} - 1]$ lorsque $k < s$ et $\alpha_s = a[i_s : m]$, d'autre part, les sous-suites β_k, $1 \leq k \leq s$, de la suite b telles que $\beta_1 = b[1 : j_1]$ et $\beta_k = b[j_{k-1} + 1 : j_k]$ lorsque $1 < k \leq s$. Par ailleurs, l'inégalité (1) permet de considérer d'une part, pour tous k, $1 < k \leq s$, les facteurs $x_k = p'[\phi_a(i_k) \rightarrow \lambda_b(j_{k-1})]$ de p', d'autre part, pour tous k, $1 \leq k < s$, les facteurs $y_k = p'[\phi_a(i_{k+1}) \rightarrow \lambda_b(j_k)]$ de p'. De plus, on pose $x_1 = y_s = \varepsilon$. On vérifie alors aisément que, par construction et à partir des propriétés (1) et (2), les quadruplets $(\alpha_k, \beta_k, x_k, y_k)$, $1 \leq k \leq s$, sont des 1-recouvrements − indiqués figure 4 par les lignes verticales en pointillé −, qu'ils sont emboîtables dans l'ordre croissant de leur indice et que leur emboîtement est égal au recouvrement $(a, b, \varepsilon, \varepsilon)$. Donc $p' \in K$.

[$\mathbf{W^* K\, W^* \subseteq A}$] D'après les propriétés (1.1a) et (3.2a).

A partir du lemme 3.B, le théorème suivant fournit une expression explicite du langage AI.

THÉORÈME 3.C - Pour tout dictionnaire W sur un alphabet V,

(3.3a) $AI = K - (W^+K \cup K\,W^+ \cup W^+K\,W^+)$;

(3.3b) AI est un langage rationnel.

dem. **(3.3a)** D'après la propriété (1.1c) et le lemme 3.B, on a : $AI = W^*K\,W^* - (W^+W^*K\,W^* \cup W^*K\,W^*W^+ \cup W\,W^*K\,W^*W^+)$. On établit (3.3a) en remplaçant W^* par $W^+ \cup \{\varepsilon\}$. **(3.3b)** D'après les propriétés de fermeture des langages rationnels pour les opérations ensemblistes.

4 Liens avec l'algorithmique des codes

On présente brièvement dans ce chapitre comment ce qui précède conduit à deux algorithmes pour décider si un dictionnaire est ou n'est pas un code et comment le premier s'apparente à l'algorithme de Sardinas et Patterson (SP) et le second à celui de Spehner. Le chapitre est limité aux dictionnaires, donc aux codes finis.

4.1 *Caractérisation des codes à partir des recouvrements.* Un dictionnaire W est un code si, pour toutes suites a et b non vides sur W, l'égalité $\gamma(a) = \gamma(b)$ implique l'égalité $a = b$, i.e. γ est injective. Soit la suite $(R_n\,; n \geq 0)$ d'ensembles de recouvrements définie par :

(4.1) $R_0 = \{(a, b, \varepsilon, y) \in R^1\}$ et, pour tout $n \geq 0$, $R_{n+1} = R_n \circ R^1$.

On note FD_n, pour tout $n \geq 0$, l'ensemble des feuillures droites des éléments de R_n.

LEMME 4.A - Soit un dictionnaire W. Les propositions ci-dessous sont équivalentes :

(4.2a) W est un code ;

(4.2b) l'ensemble A des éléments ambigus de W^+ est vide ;

(4.2c) le langage L sur l'alphabet des 1-recouvrements de W est vide ;

(4.2d) pour tout $n \geq 0$, $\varepsilon \notin FD_n$;

(4.2e) l'ensemble des états utiles[5] de l'automate $(R^1, \widehat{F^1}, \varepsilon_g, T, \varepsilon_d)$ est vide. ∎

Chacune des propriétés (4.2d) et (4.2e) conduit naturellement à un algorithme de décision pour les codes. Le premier, basé sur la suite des ensembles FD_n, est une visite en largeur des états de l'automate du langage L à partir de l'état ε_g. Il s'apparente à l'algorithme SP. Le second consiste à calculer l'ensemble des états utiles de l'automate du langage L et s'apparente à l'algorithme de Spehner.

4.2 *Eléments de comparaison avec l'algorithme* SP. Si X et Y sont des langages sur V, on note $X^{-1}Y$ le langage $\{z \in V^* | \exists x \in X, \exists y \in Y, y = x\,z\}$. Dès lors, soit la suite de langages définie par :

(4.3) $U_0 = W^{-1}W - \{\varepsilon\}$ et, pour tout $n \geq 0$, $U_{n+1} = W^{-1}U_n \cup U_n^{-1}W$.

L'algorithme SP est basé sur la propriété selon laquelle W est un code ssi, pour tout $n \geq 0$, $\varepsilon \notin U_n$ (Berstel & Perrin, 1985). Son itération est celle des ensembles U_n tandis que celle issue de la propriété (4.2d) est celle des ensembles FD_n. Le lemme ci-après fournit une définition directe et récursive des ensembles FD_n, qui constitue un élément pour comparer les deux suites.

[5] Un état q d'un automate fini est *utile* s'il est *accessible* (il existe un chemin d'un état initial à q) et *coaccessible* (il existe un chemin de q à un état final).

LEMME 4.B − Soit un dictionnaire W. On a :

(4.4a) $FD_0 = (W^{-1}W - \{\varepsilon\}) \cup (WW^+)^{-1}W \cup W^{*-1}((W^{+-1}W)^{-1}W)$;

(4.4b) $FD_{n+1} = W^{*-1}((W^{*-1}({FD_n}^{-1}W))^{-1}W)$, pour tout $n \geq 0$.

indications. Dans l'expression (4.4a), le premier terme de l'union est l'ensemble des feuillures à droite f des 1-recouvrements de la forme $((a_1),(b_1),\varepsilon,f)$, $a_1 \in W$, $b_1 \in W$, le second celui de ceux de la forme $((a_1),(b_1,\ldots,b_n),\varepsilon,f)$, $n \geq 2$, et le dernier celui de ceux de la forme $((a_1,\ldots,a_m),(b_1,\ldots,b_n),\varepsilon,f)$, $m \geq 2$, $n \geq 2$.

4.3 *Eléments de comparaison avec l'algorithme de Spehner.* Un quadruplet (w, s, u, v), dans lequel $w \in W$, s est une suite non vide sur W, $u \in V^*$ et $v \in V^*$, est un *S-quadruplet* si $w = u\,\gamma(s)\,v$. Soient la relation binaire $B = \{(u,v)|(w,s,u,v)$ est un S-quadruplet$\}$ sur V^* et l'ensemble C des éléments $c \neq \varepsilon$ de V^* tels qu'il existe, dans B, un chemin de ε à c et un chemin de c à ε. L'algorithme de Spehner est basé sur la propriété selon laquelle W est un code ssi l'ensemble C est vide (Lallement, 1979). Moyennant les transformations entre S-quadruplets et 1-recouvrements que décrit le lemme ci-après et que montre la figure 5, il est aisé d'établir que la vacuité de l'ensemble C est équivalente à la propriété (4.2e).

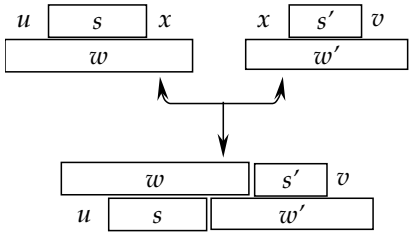

Figure 5 : S-quadruplets et 1-recouvrements

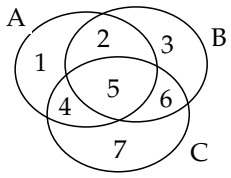

Figure 6 : partition Π

LEMME 4.C - Etant donné un dictionnaire W,

(A) si deux S-quadruplets sur W sont de la forme (w, s, u, x) et (w', s', x, v), alors le quadruplet $(w + s', s + w', u, v)$, dans lequel $+$ désigne l'adjonction à gauche ou à droite d'un élément à une suite, est un 1-recouvrement sur W ;

(B) si (a, b, u, v) est un 1-recouvrement avec $a = (a_i\,; 1 \leq i \leq m)$ et $b = (b_j\,; 1 \leq j \leq n)$, et si x désigne la chaîne sur laquelle a_1 et b_n se chevauchent, alors les quadruplets $(a_1, (b_j\,; 1 \leq j \leq n-1), u, x)$ et $(b_n, (a_i\,; 2 \leq i \leq m), x, v)$ sont des S-quadruplets. ■

5 Calcul d'une expression régulière du langage AI

La méthode de calcul présentée est en plusieurs étapes. Certaines utilisent des techniques avérées, d'autres sont plus spécifiques. A partir de l'expérience déjà menée sur la langue laotienne, ce chapitre expose la méthode puis ses aspects spécifiques.

5.1 *La méthode.* Les règles (3.1) et la propriété (3.3a) conduisent à la succession des étapes ci-dessous pour calculer une expression régulière du langage AI :

(5.1a) calcul de l'ensemble R^1 des 1-recouvrements du dictionnaire W ;

(5.1b) calcul de l'automate AUT_L du langage L à partir des règles (3.1) ;

(5.1c) calcul de l'automate AUT_K du langage K à partir de l'automate AUT_L ;

(5.1d) calcul de l'automate AUT_{AI} de AI à partir de l'automate AUT_K et de l'expression (3.3a) de AI ;

(5.1e) calcul d'une expression régulière de AI à partir de l'automate AUT_{AI}.

Avant d'être implantée en vraie grandeur, la chaîne de calculs (5.1) fut testée "à la main" sur des exemples réduits grâce au logiciel interactif (*Automate*, 2000) dédié aux automates finis. Elle fut alors entièrement réalisée sur la langue laotienne dont le dictionnaire des syllabes compte environ 56 000 entrées. En fonction des conclusions de cette expérience la chaîne est en cours de reprogrammation. La première leçon est que les temps d'exécution et l'espace-mémoire qu'elle requiert peuvent devenir "redoutables" avec la taille du dictionnaire W. Or, le dictionnaire du thaï compte plus d'un million de syllabes, celui du khmer plus d'un milliard ! Ces dictionnaires sont décrits par des grammaires de forme hors contexte sans symbole non terminal récursif de façon qu'ils restent finis. Pour chacune des langues étudiées, sa grammaire recense non seulement les syllabes attestées dans les mots mais aussi − et cela explique la taille volumineuse des dictionnaires − les syllabes "théoriques", autrement dit celles qui, bien qu'en accord avec leur langue, sont peu ou pas attestées mais dont l'occurrence dans un texte ne peut être exclue. Ainsi en français, de la syllabe inhabituelle *xieng* pour l'écriture de la ville laotienne *Xieng-Khouang*. Compte tenu de la taille des dictionnaires, il est essentiel de pouvoir la réduire. Aussi commence-t-on par présenter un prétraitement qui substitue à tout dictionnaire un dictionnaire de cardinalité plus petite sur lequel opérera, sans perte d'information, la chaîne (5.1).

Dans la suite, on suppose que chaque dictionnaire est stocké sous la forme de l'automate d'états finis minimal qui le reconnaît. Le récent travail de Yeu (2003) a montré qu'en pratique cette hypothèse était réaliste. Le calcul des automates minimaux s'est en effet révélé rapide − moins d'une seconde pour les syllabes du laotien, moins de quatre pour celles du khmer [6]. Partant de la grammaire du dictionnaire, l'algorithme en calcule une expression régulière, puis un automate non déterministe, finalement l'automate minimal. On souligne que l'algorithme "profite" de la forme grammaire originelle des dictionnaires : il ne réclame pas de stocker *in extenso* les dictionnaires ou de les énumérer syllabe après syllabe. Dans l'exemple du khmer, ces solutions seraient difficiles, voire impossibles, à mettre en œuvre compte tenu du grand nombre de syllabes, même à partir des techniques proposées par Daciuk,

[6] Les temps d'exécution cités sont tous relatifs à une machine cadencée à 1,5 GHz.

Watson & Watson (1998) pour la construction des automates minimaux acycliques. On précise enfin que l'algorithme qui réalise toutes les déterminisations d'automates mentionnées dans l'article a été implanté à partir des techniques décrites par Leslie (1995) et que l'algorithme de Brzozowski (Watson, 1993) est celui utilisé pour toutes les minimisations.

5.2 *Réduction de la taille des dictionnaires.* Etant donné un dictionnaire W, on considère la relation d'équivalence $\equiv$ sur l'alphabet V définie par :

$$\forall a \in V,\, \forall b \in V,\, a \equiv b \overset{déf}{\Longleftrightarrow} \forall m \in V^*,\, \forall m' \in V^*,\, m\,a\,m' \in W \Leftrightarrow m\,b\,m' \in W.$$

Intuitivement, $a \equiv b$ si a et b "jouent le même rôle" dans le dictionnaire W. Soient $V/_{\equiv}$ l'ensemble quotient et π la surjection canonique qui associe à tout $v \in V$ sa classe d'équivalence $\pi\, v$. Considérant alors $V/_{\equiv}$ comme un nouvel alphabet, la surjection π s'étend canoniquement en un morphisme de monoïdes libres de V^* vers $(V/_{\equiv})^*$: $\pi\, v_1 v_2 \ldots v_n = \pi\, v_1\, \pi\, v_2 \ldots \pi\, v_n$ et $\pi\, \varepsilon = \varepsilon$. Par ailleurs, on note $\pi\, W$ le dictionnaire $\{\pi\, w | w \in W\}$ sur $V/_{\equiv}$. Prenant $W = \{aa, ab, ba, bb, c\}$ et posant $C^1 = \{a, b\}$ et $C^2 = \{c\}$, alors $V/_{\equiv} = \{C^1, C^2\}$, $\pi\, a = \pi\, b = C^1$, $\pi\, c = C^2$ et $\pi\, W = \{C^1C^1,\, C^2\}$.

LEMME 5.A – Pour tout dictionnaire W,

$$\text{(5.2a)} \quad W = \pi^{-1}\pi\, W\,; \qquad \text{(5.2b)} \quad AI(W) = \pi^{-1} AI(\pi\, W)\,. \qquad \blacksquare$$

Eu égard à la propriété (5.2b), l'évaluation du langage $AI(W)$ sur V se ramène à celle du langage $AI(\pi\, W)$ sur $V/_{\equiv}$. Or le cardinal de $V/_{\equiv}$ est, par construction, au plus égal à celui de V et, par conséquent, celui du dictionnaire $\pi\, W$ au plus égal à celui du dictionnaire W. Selon les dictionnaires, le gain de cardinalité peut se révéler important. Il l'est pour le dictionnaire laotien dont la taille est divisée par 5 (56 000 → 11 000), bien plus pour celui du khmer dont la taille est divisée par 10^4 (1 milliard → 100 000).

On montre maintenant comment évaluer la partition $V/_{\equiv}$ à partir de l'automate minimal AUT_W du dictionnaire W. Pour cette circonstance, on considérera une version complète de l'automate [7]. Soit T_W l'ensemble de ses transitions. Pour tout couple (q_i, q_j) d'états, on note $V_{i,j}$ le sous-ensemble $\{c \in V | (q_i, c, q_j) \in T_W\}$ de V. L'union des sous-ensembles $V_{i,j}$ est égale à V. On note Π la moins fine [8] des partitions de V compatibles avec la famille des sous-ensembles $V_{i,j}$ [9]. Sur la figure 6, les ovales étiquetés par des lettres représentent les ensembles $V_{i,j}$ et la partition Π est constituée des "tommettes" numérotées.

[7] Un automate est *complet* si sa relation de transition est le graphe d'une application du produit $Q \times V$ dans Q.

[8] Si Π et Π' sont des partitions d'un même ensemble, Π est plus *fine* que Π' si chaque classe de Π est incluse dans une classe de Π'.

[9] Une partition Π d'un ensemble E est *compatible* avec une famille F de sous-ensembles non vides de E si pour chaque élément p de Π et chaque élément f de F, soit $p \subseteq f$, soit p et f sont disjoints.

LEMME 5.B − Pour tout dictionnaire W, les partitions Π et $V/_{\equiv}$ de V sont égales.

dem. [Π **plus fine que** $V/_{\equiv}$] Soient c^1 et c^2 dans la même classe C de Π. Puisque l'automate est déterministe et complet, pour tout état q_i, il existe un unique état q_j tel que $(q_i, c^1, q_j) \in T_W$, autrement dit $c^1 \in V_{i,j}$. Or, de par la définition de la relation Π, $C \subseteq V_{i,j}$, donc $(q_i, c^2, q_j) \in T_W$. Il en résulte que $c^1 \equiv c^2$. [$V/_{\equiv}$ **plus fine que** Π] Soit $c^1 \equiv c^2$ et soit $(q_i, c^1, q_j) \in T_W$. Puisque l'automate est déterministe et complet, il existe un unique état q_k tel que $(q_i, c^2, q_k) \in T_W$. Soit $m \in V^*$ tel que q_i soit l'état d'arrivée de l'automate opérant sur m à partir de l'état initial. Puisque $c^1 \equiv c^2$, pour tout $m' \in V^*$, $mc^1m' \in W \Leftrightarrow mc^2m' \in W$. Le langage reconnu par l'automate à partir de l'état q_j et celui à partir de l'état q_k sont donc égaux. Or l'automate est minimal. Il en résulte que $q_j = q_k$ d'après le théorème de Myhill-Nérode (Aho & Ullman, 1972). Il s'ensuit que c^1 et c^2 appartiennent à la même classe de Π.

5.3 *Calcul* (5.1a) *de l'ensemble des* 1-*recouvrements.* Dans ce parapraphe, on note, pour tout $v = c_1c_2 \ldots c_l$, dans V^*, $\widetilde{v} = c_l \ldots c_2c_1$ son image miroir et on désigne par $\widetilde{W}$ le dictionnaire $\{\widetilde{w} | w \in W\}$. Par ailleurs, si $v \in V^+$ est un chemin dans un automate *aut*, on dit que v *se prolonge* dans *aut* s'il existe c dans V tel que la concaténation vc est un chemin dans *aut*.

Un 1-recouvrement est *minimal* s'il est de la forme $((a_1), (b_1), x, y)$ dans laquelle les suites a et b sont de longueur 1. Le calcul des 1-recouvrements minimaux est la base pour celui de tous les 1-recouvrements puisque ceux-ci s'obtiennent en "comblant" de toutes les manières possibles, avec des éléments de W, les feuillures de ceux-là. L'algorithme "spontané" pour calculer l'ensemble des 1-recouvrements minimaux consiste à examiner tous les couples (w_1, w_2) d'éléments du dictionnaire W et, pour chacun d'eux, à calculer toutes les manières qu'ont w_1 et w_2 de se chevaucher, sans se confondre lorsqu'ils sont égaux. Cet examen est une itération de $|W|^2$ tours si $|W|$ est le cardinal de W. Sa complexité peut être significativement réduite en considérant les automates minimaux AUT_W de W et $AUT_{\widetilde{W}}$ de $\widetilde{W}$, qu'on supposera, pour cette circonstance, tous deux émondés [10]. Tout 1-recouvrement minimal est de la forme $((x\,\mu), (\mu\,y), x, y)$, $\mu \in V^+$, $x, y \in V^*$ et $xy \neq \varepsilon$. On dit de μ qu'il est *le milieu* du 1-recouvrement.

LEMME 5.C − Pour tout dictionnaire W, un élément μ de V^+ est un milieu ssi la conjonction des conditions (5.3) est vérifiée :

(5.3a) μ est un chemin dans l'automate AUT_W en partant de son état initial ;

(5.3b) $\widetilde{\mu}$ est un chemin dans l'automate $AUT_{\widetilde{W}}$ en partant de son état initial ;

(5.3c) $(\mu \notin W) \vee (\mu$ se prolonge dans $AUT_W) \vee (\widetilde{\mu}$ se prolonge dans $AUT_{\widetilde{W}})$.

dem. $\mu \in V^+$ est un milieu ssi $F(\mu) \stackrel{déf}{\equiv} (\exists x \in V^*, \exists y \in V^*, \widetilde{\mu}\,\widetilde{x} \in \widetilde{W}, \mu\,y \in W, xy \neq \varepsilon)$ vaut *vrai*. D'une part, on a $F(\mu) \Rightarrow$ (5.3a) $\wedge$ (5.3b). D'autre part, si

[10] Un automate est *émondé* (*trim automata*) si tous ses états sont utiles (Bellot & Sakarovitch, 1998).

la conjonction (5.3a) $\wedge$ (5.3b) est vérifiée, alors lorsque $\mu \notin W$, $F(\mu)$ vaut *vrai* puisque les automates AUT_W et $AUT_{\widetilde{W}}$ sont émondés et, lorsque $\mu \in W$, $F(\mu)$ est équivalente à la disjonction (μ se prolonge dans AUT_W) $\vee$ ($\widetilde{\mu}$ se prolonge dans $AUT_{\widetilde{W}}$). Donc, $F(\mu) \Leftrightarrow$ (5.3a) $\wedge$ (5.3b) $\wedge$ (5.3c) [11].

Ce lemme conduit à un algorithme basé sur l'énumération de tous les chemins non vides μ de AUT_W. Pour chacun d'eux, il vérifie que $\widetilde{\mu}$ est un chemin dans $AUT_{\widetilde{W}}$, s'assure que la condition (5.3c) est satisfaite, calcule alors tous les 1-recouvrements qui ont μ comme milieu. Il existe autant de chemins non vides dans AUT_W à partir de son état initial que de préfixes non vides dans W. L'algorithme réalise donc au plus $\overline{lg}(W) \times |W|$ tours, si $\overline{lg}(W)$ désigne la longueur moyenne des éléments de W. Pour le laotien, dont la version réduite du dictionnaire compte environ 11 000 syllabes de longueur moyenne 3,76 caractères, l'algorithme "spontané" réaliserait $11\,000^2$ itérations tandis que celui issu du lemme (5.C) en réalisera au plus $3,76 \times 11\,000$.

5.4 *Calculs* (5.1b) *et* (5.1c) ; *réduction du nombre de transitions de* AUT_K. Les calculs (5.1b) et (5.1c) se résument à appliquer les règles du chapitre 3, puis à émonder les automates visés. Ces étapes introduisent une nouvelle réduction des données, moins importante que celle du paragraphe 5.2 mais non négligeable. En effet, le nombre de transitions de l'automate AUT_L, partant celui de l'automate AUT_K, peut être réduit. A partir de l'automate AUT_L du langage L, on considère l'automate obtenu en supprimant toutes les transitions qui "partent" de l'état final ε_d, i.e. les transitions de la forme $(\varepsilon_d, (a, b, \varepsilon, y), y)$. Cet automate reconnaît le langage $L^{\sim}$ sur l'ensemble R^1 des 1-recouvrements. Le même cheminement que celui qui, à partir du langage L, aboutit au langage K (chapitre 3) conduit, à partir de $L^{\sim}$, au langage $K^{\sim}$ et à son automate $AUT_{K^{\sim}}$.

LEMME 5.D – Pour tout dictionnaire W,

$$AI = K^{\sim} - (W^+K^{\sim} \cup K^{\sim}W^+ \cup W^+K^{\sim}W^+). \tag{5.4}$$

dem. Par construction, $K = K^{\sim} \cup KK^{\sim}$. Donc, à partir de la propriété (3.3a), $AI = (K^{\sim} \cup K\,K^{\sim}) - (W^+K \cup KW^+ \cup W + KW^+)$. Or, $KK^{\sim} \subseteq W^+K$, donc, $AI = K^{\sim} - (W^+K \cup KW^+ \cup W^+KW^+)$. Par ailleurs, $W^+K = W^+K^{\sim} \cup W^+KK^{\sim} = W^+K^{\sim}$ puisque $W^+K \subseteq W^+$. De même, $W^+K = W^+K^{\sim}$ et $W^+KW^+ = W^+K^{\sim}W^+$.

Ce lemme optimise l'étape (5.1d) en remplaçant le langage K par le langage $K^{\sim}$ dont l'automate a moins de transitions et l'égalité (3.3a) par l'égalité (5.4).

[11] En appliquant la règle : si $p \Rightarrow q$ et $q \Rightarrow ((\neg r \wedge p) \vee (r \wedge (p \Leftrightarrow s)))$, alors $p \Leftrightarrow (q \wedge (\neg r \vee s))$.

5.5 *Calcul* (5.1d) *de l'automate* AUT_{AI}. Ce calcul est probablement le plus délicat. En effet, la présence dans la formule (5.4) d'une différence ensembliste réclame que les automates de ses opérandes soient déterministes, savoir celui de $K^{\sim}$ et celui de l'union ($W^+K^{\sim} \cup K^{\sim}W^+ \cup W^+K^{\sim}W^+$). Or l'algorithme pour rendre déterministe un automate fini est, dans le pire des cas, de l'ordre de 2^q, q étant le nombre d'états. Le tableau de la figure 7 est lié au laotien. Il indique, pour certains des langages intermédiaires intervenant dans la formule (5.4), le nombre d'états (en gras) et le nombre de transitions (en italique) des versions non déterministe et minimale de leur automate. Il montre le fort "peuplement" des automates non déterministes opposé à celui de leur correspondant minimal. Ce constat a conduit à systématiquement minimiser chacun des automates intermédiaires. Le temps de calcul de l'étape (5.1d), appliquée au laotien, fut de sept minutes.

	non déterministe	minimal
$K^{\sim}$	**13829** ; *88881*	**41** ; *262*
$W^+ K^{\sim}$	**14403** ; *95574*	**85** ; *1486*
$K^{\sim} W^+$	**15550** ; *96147*	**75** ; *913*
$W^+ K^{\sim} W^+$	**15550** ; *102840*	**126** ; *2133*
AI		**38 ;** *253*

Figure 7 : taille des automates

5.6 *Calcul* (5.1e) *d'une expression régulière du langage AI*. Une expression régulière est obtenue, à partir de l'automate AUT_{AI}, par la méthode basée sur l'algorithme de résolution d'équations linéaires par élimination gaussienne (Aho & Ullman, 1972), (Bellot & Sakarovitch, 1998). En développant cette expression, on aboutit à la liste des motifs irréductibles. Le laotien en compte quelque 240 000. Certains contiennent l'itérateur * de Kleene. Ainsi de :

ແກ ອ (ອ ອ ນ ອ)* ງ ັ ງ et de ເ ກ ็ ອ (ອ)* ນ ອ (ອ ອ ນ ອ)* ງ ັ ງ .

Le langage des ambiguïtés irréductibles issu de la grammaire des syllabes laotiennes utilisée est donc infini. Cependant, on fait remarquer qu'aucun des motifs infinis ne correspond à des suites de mots de la langue laotienne. Ceci ne doit pas surprendre puisque toutes les concaténations de syllabes, i.e. tous les éléments de W^+, ne font pas partie de la langue laotienne.

Références

[1] *Automate* , logiciel, Caylux, B., (2000).
Téléchargeable à partir de `brassens.upmf-grenoble.fr/IMSS/logiciels/`

[2] Aho A. V. & Ullman J. D., (1972). *The theory of parsing, translation and compiling.* Prentice Hall.

[3] Bellot P. & Sakarovitch J., (1998). *Logique et automates.* Ellipses.

[4] Berment V., (1997). *Traitement automatique du laotien. Quelques aspects morphologiques.* Mémoire de DREA, Institut National des Langues et Civilisations Orientales (INALCO), 96 p.

[5] Berment V., (1998). *Prolégomènes graphotaxiques du laotien.* Mémoire de DEA, Institut National des Langues et Civilisations Orientales (INALCO), 160 p.

[6] Berment V., (2002). *Several Technical Issues for Building New Lexical Bases.* Papillon Seminar, July 2002. 5 p.

[7] Berstel J. & Perrin D., (1985). *Theory of codes.* Academic Press, Orlando.

[8] Daciuk J., Watson B. W. & Watson R. E., (1998). *Incremental construction of minimal acyclic finite state automata and transducers.* Actes de *Finite State Methods in Natural Language Processing*, Université de Bilkent, Ankara, Turquie, juin-juillet 1998.

[9] Head T. & Weber A., (1995). *Deciding multiset decipherability.* IEEE transactions on information theory, vol. 41, n, pp291-297.

[10] Lallement G., (1979). *Semigroups and combinatorial applications.* John Wiley & sons, New York.

[11] *LaoWord*, logiciel, Berment V., (2000).
Téléchargeable à partir de `www.LaoSoftware.com`

[12] Leslie T., (1995). *Efficient Approaches to Subset Construction.* Masters thesis, Computer Science, University of Waterloo. Téléchargeable à partir de `www.csd.uwo.ca/research/grail/.papers/subset.ps`

[13] Lothaire M., (2002). *Algebraic combinatorics on words.* Cambridge University Press.

[14] Watson B. W., (1993). *A taxonomy of finite automata minimization algorithms.* Université de Technologie d'Eindhoven, Pays-Bas. Téléchargeable à partir de `www.cs.up.ac.za/cs/bwatson/publications.html#1993`

[15] Weber A. & Head T., (1994). *The finest homophonic partition and related code concepts.* Actes de *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, n841, pp618-628, Springer-Verlag, ed. Privara, Rovan & Ruzicka.

[16] Yeu T., (2003). *Découpage syllabique des langues du Sud-Est Asiatique.* Rapport de stage IUT, Université René Descartes, Paris 5, 28 pages.

Centre d'Analyse et de Mathématique Sociales, CNRS,
44, rue de l'Amiral Mouchez, 75014 Paris,
email : delvigna@ivry.cnrs.fr

Groupe d'Etude pour la Traduction Automatique, CLIPS
385, rue de la Bibliothèque, BP 53
38040 Grenoble CEDEX 9
email : Vincent.Berment@imag.fr