INFORMATIQUE THÉORIQUE ET APPLICATIONS

MARCELLA ANSELMO

Automates et codes zigzag

Informatique théorique et applications, tome 25, n° 1 (1991), p. 49-66

<http://www.numdam.org/item?id=ITA_1991__25_1_49_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 25, n° 1, 1991, p. 49 à 66)

AUTOMATES ET CODES ZIGZAG (*)

par Marcella ANSELMO (1)

Communiqué par J. BERSTEL

Résumé. – *Le point de départ de cette étude est un théorème d'Isbell : le théorème du zigzag. L'on introduit ici la notion de « zigzag » sur les mots et les langages, ainsi que la notion de code zigzag. Il sera fait un usage fondamental des automates boustrophédons (ou two-way, en anglais) et l'on remarquera l'équivalence de ces derniers avec les automates usuels. Le fait de pouvoir associer à tout langage reconnaissable un automate boustrophédon particulier nous permettra de prouver la décidabilité des codes zigzag finis et de certains autres.*

Abstract. – *Starting from the Isbell's "zigzag's theorem", we introduce here the notion of "zigzag" over words and languages as well as that of zigzag code.*

We associate to each language a special two-way automaton representing the set of "zigzag" over it. This allows us to decide whether or not a finite language is a zigzag code. The result can be extended to some infinite languages.

1. FACTORISATIONS ET CODES ZIGZAG

Le théorème du zigzag d'Isbell, présenté en 1965 [4] concerne la notion de « dominion » d'un semi-groupe, notion que l'on va introduire selon la notation d'Isbell.

Soit S un semi-groupe et X un sous-semi-groupe. On dit que X *domine* un élément $d \in S$ si pour tout semi-groupe T et pour tout couple de morphismes $\beta, \gamma : S \to T$

$$(\forall x \in X,\ x\beta = x\gamma) \quad \Rightarrow \quad d\beta = d\gamma.$$

(*) Reçu en avril 1989, révisé en juin 1990.
Cette recherche a été soutenue en partie par le C.N.R. italien.
(1) Laboratoire d'Informatique Théorique et Programmation, Université Paris-VII, 2, place Jussieu, 75251 Paris Cedex 05.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/91/01 49 18/$3.80/

Plus simplement, X domine d si, dès que deux morphismes quelconques dans S coïncident sur les éléments de X, alors ils coïncident aussi sur d.

L'ensemble des éléments dominés par X, est appelé le *dominion* de X dans S et noté $\mathrm{Dom}_s(X)$.

Étant donnés un semi-groupe S et un sous-semi-groupe X de S, soient $S' = S \cup \{1\}$, $X' = X \cup \{1\}$, où 1 est une identité adjointe et soit τ la relation d'équivalence dans $S' \times S'$ engendrée par :

$$T = \{(ux, v), (u, xv)/u, v \in S', x \in X'\}.$$

L'ensemble $S' \times S'/\tau$ sera noté par $S' \odot_{X'} S'$ et appelé le *produit tensoriel* de S' sur X'.

La classe de τ-équivalence de (u, v) sera notée par $u \odot v$.

Le théorème d'Isbell est alors le suivant [4].

THÉORÈME DU ZIGZAG : *Si X est un sous-semi-groupe d'un semi-groupe S et si $d \in S$, alors $d \in \mathrm{Dom}_s(X)$ ssi $d \odot 1 = 1 \odot d$ dans le produit tensoriel $A = S' \odot_{X'} S'$.*

En analogie avec les idées précédentes, on va maintenant introduire les notions de factorisations et codes zigzag.

Étant donné un langage X sur un alphabet fini A, soit τ la relation d'équivalence sur $A^* \times A^*$, introduite ci-dessus, avec $S' = A^*$ et $X' = X$.

Nous écrirons $(u, v) \to (u', v')$ si et seulement si $((u, v), (u', v')) \in T$ ou $((u', v'), (u, v)) \in T$: nous appellerons $(u, v) \to (u', v')$ un *pas*. Nous ne pouvons donc avoir que des pas de la forme $(ux, v) \to (u, xv)$, que nous appellerons « vers la gauche » ou « en arrière » ; ou de la forme $(u, xv) \to (ux, v)$, que nous appellerons « vers la droite » ou « en avant ». On désignera encore par $u \odot v$ la classe de τ-équivalence du couple (u, v) et par ε le mot vide de A^*. Nous pouvons alors introduire l'ensemble des mots obtenus par zigzag sur les mots de X, ce qu'on notera $X^\uparrow$.

DÉFINITION 1 : Pour tout sous-ensemble X de A^* le langage zigzag sur X est

$$X^\uparrow = \{w \in A^*/\varepsilon \odot w = w \odot \varepsilon\}.$$

Donc un mot w dans A^* appartient à $X^\uparrow$ s'il existe une suite (finie) de pas reliant (ε, w) à (w, ε) ; mais X n'étant pas un semi-groupe, nous ne pouvons plus exiger que dans la suite les pas vers la gauche et vers la droite alternent ; cependant le premier et le dernier pas doivent encore être vers la droite.

DÉFINITION 2 : Pour tout mot $w \in X^{\uparrow}$, une *factorisation zigzag* de w sur X, est une suite de pas $(u_i, v_i) \to (u_{i+1}, v_{i+1})$, $i = 1, 2, \ldots, m$ telle que

(1) $u_1 = v_{m+1} = \varepsilon$ et $v_1 = u_{m+1} = w$;

(2) il n'y a pas de sous-suites

$$(u_j, v_j) \to (u_{j+1}, v_{k+1}) \to \ldots \to (u_{j+k}, v_{j+k})$$

où

$$1 \leqq j < j+k \leqq m+1$$

avec

$$u_j = u_{j+k} \qquad \text{et} \qquad v_j = v_{j+k}.$$

Plus intuitivement, nous pouvons regarder la virgule dans (u, xv) comme la position d'un curseur dans le mot uxv ; le fait qu'on peut passer de (u, xv) à (ux, v) nous dit alors que le curseur peut se déplacer vers la droite et s'arrêter à chaque fois qu'il a lu un mot de X. De même vers la gauche.

Il faut remarquer que la condition (2) de la définition, exclut dans une factorisation zigzag la présence de « boucles » sur une même position : en effet, on pourrait répéter ces boucles autant de fois que l'on veut dans la factorisation et ceci donnerait lieu à un nombre infini de factorisations qui ne différeraient que pour le nombre de boucles.

Exemple : $A = a + b$, $X = a + aba$.

Le mot *ababa* appartient à $X^{\uparrow}$, même s'il n'appartient pas à X^*.

Pour $w = ababa$ une factorisation zigzag sur X est :

$$(\varepsilon, w) = (\varepsilon, ababa) \to (aba, ba) \to (ab, aba) \to (ababa, \varepsilon)$$

qu'on pourra aussi visualiser par

On verra ensuite, que $X^{\uparrow} = a(a + ba)^* + \varepsilon$.

Remarque : On a pour tout sous-ensemble X de A^* l'inclusion $X^{\uparrow} \supseteq X^*$, mais pas toujours l'inclusion inverse, comme on le voit dans l'exemple. En

fait, si $w \in X^*$, alors $w = x_1 x_2 \ldots x_m$ avec $x_i \in X$, $\forall i = 1, 2, \ldots, m$, et la suite

$$(\varepsilon, w) = (\varepsilon, x_1 x_2 \ldots x_m) \rightarrow (x_1, x_2 \ldots x_m) \rightarrow \ldots \rightarrow (x_1 \ldots x_j, x_{j+1} \ldots x_m) \rightarrow \ldots \rightarrow (w, \varepsilon)$$

est une factorisation zigzag de w comme mot de $X^\uparrow$.

Étant donné un mot $w \in X^\uparrow$, nous pouvons maintenant nous demander combien de factorisations zigzag sur X, il admet et introduire la notion de « code zigzag ».

DÉFINITION 3 : Un sous-ensemble X de A^* est un *code zigzag* ssi tout mot de A^* admet au plus une factorisation zigzag sur X.

On remarque que tout code zigzag est un code dans le sens classique.

Exemples de codes zigzag et non : (1) Tout code X préfixe ou suffixe (ou bipréfixe) est trivialement zigzag.

Soit l'alphabet $A = a + b$.

(2) $X_1 = a + aba$ est un code et il est aussi un code zigzag. En effets la seule façon, dans une factorisation zigzag d'un mot de $(X_1)^\uparrow$, de faire des pas vers la gauche et de repartir après vers la droite sans faire de boucles, est la suivante :

$$(u, abav) \rightarrow (uaba, v) \rightarrow (uab, av) = (uab, abaz) \rightarrow (ababa, z)$$

c'est-à-dire :

Il résulte que $X^\uparrow = a(a + ba)^* + \varepsilon$; nous pouvons donc établir la factorisation zigzag d'un mot de $(X_1)^\uparrow$ de manière univoque, en regardant le nombre de « a » qui séparent deux « b » successifs.

Nous pouvons généraliser cet exemple au code

$$X_2 = a + x \qquad \text{si } x \text{ est un mot avec au moins un « } b \text{ »}$$

et au code infini

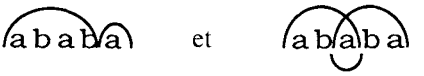
$$X_3 = a + ab^+ a$$

qui sont donc encore des codes zigzag.

Si à X_1 on rajoute le mot bb, $X_4 = a + aba + bb$ reste encore un code zigzag, car on peut univoquement décoder en observant aussi le nombre de « b » consécutifs. Par contre

$$X_5 = a + aba + abab$$

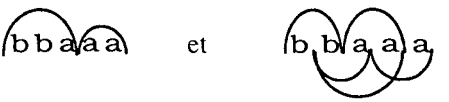
est un code, mais il n'est pas zigzag car par exemple, le mot $ababa$ admet deux factorisations zigzag distinctes :



(3) Un autre exemple de code non zigzag est

$$X = aa + ba + bb + baa + bba$$

qui n'est pas zigzag car $bbaaa$ admet deux factorisations :



(4) $X_1 = ab + abb + baab$ est un code zigzag ; nous pouvons nous en convaincre en faisant des tentatives de double factorisation zigzag pour un mot de $X_1^{\uparrow}$.

Si le mot commence par « a », nous pourrons essayer avec

mais à ce point-là, nous ne pourrons plus faire aucun pas ni vers la gauche, ni vers la droite. Une tentative analogue pour un mot commençant par « b » nous conduirait à cette situation :

d'où nous ne pourrons plus bouger.

Une procédure analogue nous conduit à dire que

$$X_2 = ab + abb + baab + bbab$$

est aussi un code zigzag.

2. LES AUTOMATES BOUSTROPHÉDONS

Pour l'étude des « zigzag » sur les mots, il est naturel et en plus très utile, d'utiliser des automates boustrophédons, ou « 2-way » selon la terminologie anglaise.

On appelle *automate boustrophédon* sur l'alphabet fini A tout quadruplet $\mathscr{A} = (Q, I, T, F)$ constitué d'un ensemble fini d'états Q disjoint de A, de deux parties de Q : I, ensemble des états initiaux et T, ensemble des états terminaux, et d'une partie F de $Q \times (A \cup A') \times Q$ contenant les flèches, où $A' = \{ a'/a \in A \}$ est une copie de A, disjointe de A et de Q.

Nous suivrons la terminologie de [5].

On appelle *section* de $w \in A^*$, tout couple $(u, v) \in A^* \times A^*$ vérifiant $uv = w$ et *description instantanée* (d'un calcul) de w, tout mot $uqv \in A^* Q A^*$ dans lequel (u, v) est une section de w. On définit sur les descriptions instantanées une relation binaire notée $\mapsto$ par les conditions (*) :

$$uq_1 av \mapsto uaq_2 v \qquad \text{si} \quad (q_1, a, q_2) \in F$$
$$uaq_1 v \mapsto uq_2 av \qquad \text{si} \quad (q_1, a', q_2) \in F.$$

Soit $\mapsto^*$ la fermeture réflexive et transitive de $\mapsto$. La relation $\mapsto^*$ est semblable à la relation τ introduite dans le chapitre précédent, mais en général elle n'est pas symétrique.

On appelle *chemin* dans $\mathscr{A}$, tout mot de F^* de la forme

$$c = (q_0, x_0, q_1)(q_1, x_1, q_2) \ldots (q_{n-1}, x_{n-1}, q_n).$$

Son étiquette est $|c| = x_0 x_1 \ldots x_{n-1} \in (A \cup A')^*$ et son étiquette réduite $\|c\|$ est le mot $|c|$ réduit par rapport à la congruence $\sim$ engendrée par l'ensemble $\{ (aa', \varepsilon), (a'a, \varepsilon)/a \in A \}$.

On appelle *calcul* de $w \in A^*$ dans $\mathscr{A}$, toute suite $w_i q_i w_i'$, $i = 0, 1, \ldots, n$ de descriptions instantanées de w vérifiant

$$w_i q_i w_i' \mapsto w_{i+1} q_{i+1} w_{i+1}' \qquad \text{pour} \quad i = 0, 1, \ldots, n-1.$$

A un tel calcul les relations (*) permettent d'associer un unique chemin dont l'étiquette est appellée une *lecture* de w. On dit que le calcul (ou la lecture associée) est *réussi* si $q_0 \in I$ et $q_n \in T$ et qu'il est *couvrant* s'il existe $i, j \in \{0, 1, \ldots, n\}$ tels que $w_i = w'_j = \varepsilon$.

On a donc $uqv \mapsto^* u' q' v'$ ssi il existe un calcul de $w = uv = u' v'$ de premier terme uqv et de dernier terme $u' q' v'$.

Si $c : q \mapsto q'$ est un chemin associé à un tel calcul on notera :

$$uqv \mapsto^{*c} u' q' v'$$

ou bien

$$(u, v) \mid c \mid = (u', v')$$

si on regarde le chemin c comme un opérateur sur les sections.

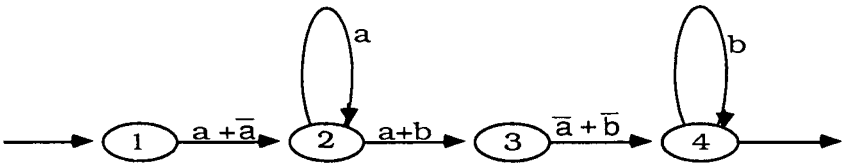
On peut définir différents modes de reconnaissance pour l'automate $\mathscr{A}$, selon le type de calculs admis [5].

Le mode classique, et le seul que nous allons considérer, consiste à accepter les mots $w \in A^*$ admettant un calcul réussi dans $\mathscr{A}$, de premier terme εiw et dernier $wt\varepsilon$, où $i \in I$, $t \in T$. Le langage des mots reconnus par $\mathscr{A}$ est donc

$$\mathscr{L}(\mathscr{A}) = \{ w \in A^* / \exists\, i \in I,\ t \in T,\ c : i \to t$$

dans $\mathscr{A}$ tel que $(\varepsilon, w) \mid c \mid = (w, \varepsilon) \}$.

Exemple : Soit $\mathscr{A}$ l'automate boustrophédon



Le mot $w = aabb$ appartient à $\mathscr{L}(\mathscr{A})$ car

$$\varepsilon\, 1\, aabb \mapsto a\, 2\, abb \mapsto aa\, 2\, bb \mapsto aab\, 3\, b \mapsto aa\, 4\, bb \mapsto aab\, 4\, b \mapsto aabb\, 4\, \varepsilon\,;$$

par contre $aaa \notin \mathscr{L}(\mathscr{A})$ et on peut se convaincre que $\mathscr{L}(\mathscr{A}) = a^+ b^+$.

Les automates boustrophédons sont aussi appelés « automates 2-way » car ils peuvent être vus comme des machines à une tête de lecture pouvant se déplacer à gauche et à droite et qui acceptent un mot si une lecture commencée sur le bord gauche dans un état initial se termine sur le bord droit dans un

état terminal. Les automates 2-way ont donc plus de liberté que les automates « 1-way ». Cependant, leur puissance n'est pas supérieure, dans le sens où les langages reconnus par les automates boustrophédons sont encore exactement les langages rationnels [2, 3, 6].

La preuve donnée par Shepherdson comporte l'introduction de deux relations entre les états de l'automate boustrophédon.

DÉFINITION 4 : Pour tout $s \in A^*$ et pour tout $p \in Q$:

$$s\rho = \{ q \in Q / \exists\, i \in I \quad \text{et} \quad c : i \to q$$

tel que $(\varepsilon, s)|c| = (s, \varepsilon)\}$;

$$p(s\varphi) = \{ q \in Q / \exists\, c : p \to q$$

tel que $(s, \varepsilon)|c| = (s, \varepsilon)\}$.

Si l'on définit en outre la relation d'équivalence ϑ dans A^* :

$$s \vartheta t \text{ ssi } s\rho = t\rho \quad \text{et} \quad s\varphi = t\varphi \quad \text{pour tout} \quad s, t \in A^*$$

on a

$$(*) \qquad s \vartheta t \;\Rightarrow\; s^{-1}L = t^{-1}L.$$

Étant donné que le nombre de couples possibles $(s\rho, s\varphi)$ est $2^{m(m+1)}$, ceci constitue une borne pour le nombre de contextes et permet de construire l'automate déterministe minimale.

Nous pouvons aussi considérer combien de fois un mot est reconnu par l'automate. Pour établir ce nombre nous ne pouvons pas considérer tous les chemins réussis de l'automate dont l'étiquette est une lecture du mot, car nous risquerions de compter aussi les chemins où il y a des « boucles » ; comme nous l'avions déjà vu pour les factorisations zigzag, ces boucles répétées autant de fois que l'on veut nous fourniraient un nombre infini de chemins en réalité tous égaux.

DÉFINITION 5 : Une *boucle* dans un chemin $c = (q_1, x_1, q_2) \ldots (q_n, x_n, q_{n+1})$ d'un automate boustrophédon, est un sous-chemin $b = c_1 c_2$ avec

$$c_1 = (q_i, x_i, q_{i+1}) \ldots (q_{j-1}, x_{j-1}, q_j)$$
$$c_2 = (q_j, x_j, q_{j+1}) \ldots (q_k, x_k, q_{k+1}), \qquad 1 \leqq i \leqq j \leqq k \leqq n$$

dans lequel $q_{k+1} = q_i \in I \cap T$ et $\| c_1 \| \,.\, \| c_2 \| \sim \varepsilon$.

DÉFINITION 6 : Pour tout automate boustrophédon $\mathscr{A}$, et tout mot $x \in A^*$ la *multiplicité* de x dans $\mathscr{L}(\mathscr{A})$, que l'on notera par $\mu_{\mathscr{A}}(x)$, est le nombre de chemins réussis et sans boucles c, avec $(\varepsilon, x) \mid c \mid = (x, \varepsilon)$.

Et finalement

DÉFINITION 7 : La série associée à un automate boustrophédon $\mathscr{A}$ sur un alphabet fini A est

$$\underline{\mathscr{A}} = \sum_{x \in A^*} \mu_{\mathscr{A}}(x)\, x$$

3. SUR LA DÉCIDABILITÉ DES CODES ZIGZAG

Afin de décider si un langage reconnaissable X est un code zigzag ou non, on commence par associer à X un automate boustrophédon $\mathscr{A}^{\uparrow}$ qui reconnaît $X^{\uparrow}$ avec la multiplicité exacte.

Si on pose

$$\mu_X(w) = \text{nombre de factorisations zigzag distinctes de } w \text{ sur } X$$

on a la

PROPOSITION 1 : *Pour tout langage reconnaissable X sur un alphabet fini A, il existe un automate boustrophédon $\mathscr{A}^{\uparrow} = (Q, 1, 1, F)$ tel que pour tout $w \in A^*$*

$$\mu_{\mathscr{A}^{\uparrow}}(w) = \mu_X(w).$$

DÉFINITION 8 : Pour tout langage reconnaissable X sur un alphabet fini A, la série associée à $X^{\uparrow}$ est

$$(\underline{X})^{\uparrow} = \sum_{w \in A^*} \mu_X(w)\, w.$$

Avec cette définition, la proposition 1 devient

PROPOSITION 1 *bis : Pour tout langage reconnaissable X sur un alphabet fini A, il existe un automate boustrophédon $\mathscr{A}^{\uparrow} = (Q, 1, 1, F)$* tel que

$$\underline{\mathscr{A}}^{\uparrow} = (\underline{X})^{\uparrow}.$$

Pour construire l'automate désiré, on commence par considérer un automate 1-way, $\mathscr{A}^* = (Q^*, 1, 1, F^*)$ dont le comportement est la série $\underline{\mathscr{A}}^* = (\underline{X})^*$, où $\underline{X}$ est la série caractéristique de X. Cet automate peut être construit en prenant par exemple l'automate déterministe minimal $\mathscr{A} = (Q', I', T', F')$ pour

lequel $\underline{\mathscr{A}} = \underline{X}$. A partir de $\mathscr{A}$, on peut construire $\mathscr{A}^*$ en posant $Q^* = Q' \cup \{1\}$, et $F^* = F' \cup S \cup E \cup B$ où

$$S = \{(1, a, q) / \exists\, i \in I' \quad \text{tel que} \quad (i, a, q) \in F'\}$$
$$E = \{(q, a, 1) / \exists\, t \in T' \quad \text{tel que} \quad (q, a, t) \in F'\}$$
$$B = \{(1, a, 1) / \exists\, i \in I, \quad t \in T \quad \text{tel que} \quad (i, a, t) \in F'\}$$

et en émondant l'automate ainsi obtenu. (Une autre technique, est présentée dans le deuxième chapitre de [1].)

Pour tout automate à un seul état initial 1, confondu avec le seul état final, on peut parler de « chemins simples ».

DÉFINITION 9 : Un chemin $c : p \to q$ dans un automate $\mathscr{A} = (Q, 1, 1, F)$ est *simple* ssi pour toute factorisation $c : p \to r \to q$ de c en deux chemins non nuls on a $r \neq 1$.

Dans le cas où X est fini, nous pouvons aussi prendre comme automate $\mathscr{A}^*$ l'automate en pétales, ([1], IV.2) qui, dans ce cas, est fini. Si l'on construit $\mathscr{A}^*$ par l'une de ces deux méthodes, il possède la propriété suivante.

PROPRIÉTÉ 1 : Pour tout mot $w \in A^*$, le nombre de chemins simples de 1 à 1 d'étiquette w, est égal à la valeur de la série caractéristique de X en w.

Pour tout mot $x \in X$, il n'y a donc qu'un seul chemin simple de 1 à 1 d'étiquette x, même si X n'est pas un code (ce qui est le cas lorsque son automate $\mathscr{A}^*$ est ambigu).

Nous pouvons maintenant construire l'automate boustrophédon cherché $\mathscr{A}^{\uparrow} = (Q, I, T, F)$.

L'ensemble de ses états est $Q = Q^+ \cup Q^- \cup \{1\}$ où $Q^+ = Q^* - \{1\}$ et Q^- en est une copie : $Q^- = \{q^- / q \in Q^+\}$; 1 est encore le seul état initial et le seul état final. Les transitions sont données par $F = F^* \cup F^-$, où F^- est un ensemble de flèches dont les étiquettes sont toutes dans A' et $(p^-, a', q^-) \in F^-$ ssi $(q, a, p) \in F^*$.

En pratique l'automate $\mathscr{A} = (Q^- \cup \{1\}, 1, 1, F^-)$ n'est qu'une copie renversée et barrée de $\mathscr{A}^*$, dans le sens où $\underline{\mathscr{A}}^- = \underline{X}'$, où $\underline{X}'$ est la série caractéristique de $X' = \{x' = x'_n x'_{n-1} \ldots x'_2 x'_1 / x = x_1 x_2 \ldots x_n \in X\}$. Des exemples sont donnés dans la suite.

Pour un tel automate $\mathscr{A}^{\uparrow}$ et pour tout $w \in A^*$, on a :

LEMME 1 : *A tout chemin c réussi sans boucles tel que* $(\varepsilon, w) \mid c \mid = (w, \varepsilon)$ *on peut faire correspondre de façon unique une factorisation zigzag sur X de w, et vice versa.*

Preuve : Soit c un chemin vérifiant les conditions du lemme. c se décompose en un produit $d_1 d_2 \ldots d_n$, de chemins simples réussis d_i. Puisque tout chemin d_i est simple, ou bien il ne traverse que des états de Q^+, et son étiquette est un mot x de X, ou bien il ne traverse que des états de Q^- et son étiquette est un mot $x' \in X'$. Dans le premier cas on pourra associer à d_i un pas (et un seul) vers la droite du genre $(u, xv) \to (ux, v)$ avec $uxv = w$; et dans le deuxième, un pas vers la gauche $(ux, v) \to (u, xv)$. Étant donné que $\|c\| = w$, qui appartient à A^*, le produit de ces pas se recolle bien pour nous donner une factorisation zigzag de w sur X.

Réciproquement, à cause de la propriété 1, à tout pas $(u, xv) \to (ux, v)$ on peut associer le seul chemin simple $1 \to 1$ d'étiquette x; et à tout pas $(ux, v) \to (u, xv)$ le seul d'étiquette x'. Il n'y a pas de problèmes pour les boucles, car on les a éliminées du comptage de $\mu_{\mathscr{A}^\uparrow}(x)$, mais aussi dans la définition de factorisation zigzag. ■

Ce lemme donne la preuve de la proposition 1.

Maintenant, nous allons voir quels rapports existent entre le nombre de factorisations zigzag pour un mot de $X^\uparrow$ et la nature de l'automate boustrophédon $\mathscr{A}^\uparrow$, que l'on vient de lui associer. On va établir ce lien, en considérant pour toute factorisation zigzag de w sur X

l'unique chemin c, qui lui correspond dans $\mathscr{A}^\uparrow$, et les états du boustrophédon auxquels on a abouti en suivant le chemin c de $\mathscr{A}^\uparrow$:

Étant donné que dans toute factorisation zigzag le dernier pas est toujours vers la droite, il existe des suffixes de w qui sont parcourus entièrement vers la droite dans la factorisation; soit U l'ensemble de ces suffixes. Dire que u est un tel suffixe, signifie que l'on peut factoriser le chemin c qui calcule w, en $c = ab$ où $b = b(u)$ est un chemin $p \to 1$, d'étiquette $u \in A^*$ qui ne passe que par des états de $Q^+ \cup \{1\}$ et où $(\varepsilon, wu^{-1})|a| = (wu^{-1}, \varepsilon)$; on fait aussi correspondre à tout $u \in U$, l'état $e(u) = p$.

Si w admet deux factorisations zigzag sur X distinctes, on peut considérer les états $e_1(u)$ et $e_2(u)$ qui correspondent dans les deux factorisations à tout suffixe u de U; ces états appartiennent à $(wu^{-1})\rho$ où ρ est la fonction relative

à $\mathscr{A}^{\uparrow}$, définie dans le chapitre 2. C'est l'étude de ces états $e_1(u)$ et $e_2(u)$, avec la connaissance de la fonction ρ relative à $\mathscr{A}^{\uparrow}$, qui nous permettra de décider si un langage fini est un code zigzag ou non.

Soit X un langage reconnaissable et soit $\mathscr{A}^{\uparrow} = (Q, 1, 1, F)$ un automate boustrophédon tel que $\mathscr{A}^{\uparrow} = (\underline{X})^{\uparrow}$, construit comme montré précédemment. Pour tout mot s de A^*, considérons le multi-ensemble $s\rho'$, qui n'est autre que $s\rho$ (où ρ est relatif à $\mathscr{A}^{\uparrow}$), où chaque état est considéré avec sa multiplicité : un état p de $s\rho'$ a multiplicité n, s'il y a « n » façons différentes d'arriver à p sur le bord droit de s, étant parti de 1 sur le bord gauche de s.

On donne aussi une définition :

DÉFINITION 10 : Étant donné un automate boustrophédon $\mathscr{A} = (Q, I, T, F)$ avec $I = T = 1$, on dit que deux états p et q sont *s-équivalents*, s'il existe deux chemins simples dans $\mathscr{A}$, $c_1 : p \to 1$, $c_2 : q \to 1$ de même étiquette.

On remarque que n'ayant pas exigé que les chemins c_1 et c_2 soient différents, un état est toujours s-équivalent à lui-même.

On a alors la

PROPOSITION 2 : *Il existe un mot $w \in A^*$ ayant deux factorisations zigzag distinctes sur X ssi il existe un mot $s \in A^*$ tel que $s\rho'$ contient deux états s-équivalents.*

(Ici « deux états » signifie deux états distincts ou bien un état avec multiplicité 2.)

Preuve : Si w a deux factorisations zigzag, soit u le plus long suffixe de w qui est visité entièrement vers la droite dans chacune des deux factorisations ; alors $e_1(u)$ et $e_2(u)$ sont deux états s-équivalents qui appartiennent tous les deux à $s\rho'$, où $s = wu^{-1}$.

Réciproquement, si p et q sont deux états (éventuellement égaux) de $s\rho'$, s-équivalents, il existe deux chemins distincts $a : 1 \to p$ et $a' : 1 \to q$ qui visitent s du bord gauche au bord droit ; et deux chemins simples, mais éventuellement égaux, $b : p \to 1$ et $b' : q \to 1$ de même étiquette, disons u. Le mot su admet alors les deux factorisations zigzag distinctes données par les chemins ab et $a'b'$. ■

Par contraposition de la proposition 2 on obtient :

PROPOSITION 2 *bis : X est un code zigzag ssi pour tout $s \in A^*$, $s\rho'$ ne contient pas deux états s-équivalents.*

Mais cette proposition ne nous donne pas une procédure effective pour décider si X est un code zigzag ou non, à cause de « pour tout $s \in A^*$ » et du

fait qu'il y a une infinité de classes $s\rho'$ possibles pour s dans A^*. Par contre on peut obtenir une procédure effective dans le cas où X est fini.

En notant ρ la fonction relative à un automate $\mathscr{A}^\uparrow$ construit comme en précédence et R, l'ensemble des représentants de longueur minimale des classes de ρ-équivalence : $s\,\rho\,t$ ssi $s\rho = t\rho$, nous avons

PROPOSITION 3 : *Soit X un langage reconnaissable fini. Alors X est un code zigzag ssi pour tout $s \in R$, $s\rho$ ne contient pas deux états distincts s-équivalents.*

Preuve : Si X est un code zigzag, on a prouvé dans la proposition 2 que, pour tout s dans A^*, $s\rho'$ ne contient pas deux états s-équivalents ; et *a fortiori* $s\rho$ non plus.

La réciproque est démontrée par contraposition : s'il existe un mot w avec plus d'une factorisation zigzag sur X, alors on trouve dans R un mot dont l'image par ρ contient deux états s-équivalents distincts. Pour cela on choisit comme $\mathscr{A}^\uparrow$, l'automate boustrophédon obtenu en ajoutant une copie renversée et barrée à l'automate en pétales (comme vu dans la proposition 1) : celui-ci est fini car X est fini.

On se sert d'un tel automate car il jouit de cette propriété.

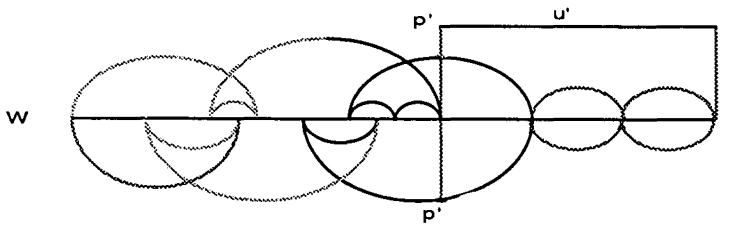
PROPRIÉTÉ 2 : Pour tout $p \in Q$, $p \neq 1$, il existe un seul chemin simple $d : p \to 1$ et un seul chemin simple $g : 1 \to p$.

Soit alors w un mot avec deux factorisations zigzag sur X; on considère l'ensemble U des suffixes parcourus seulement par des pas vers la droite dans chacune des deux factorisations.

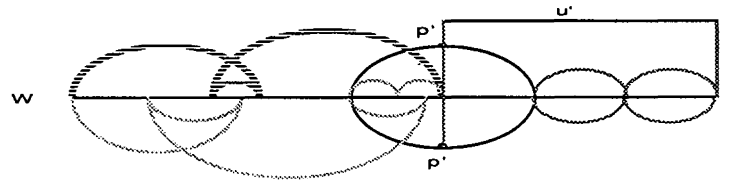
S'il existe $u \in U$ tel que $e_1(u) \neq e_2(u)$, on a fini, car ces deux états sont s-équivalents et différents, et ils appartiennent aussi au mot de R, ρ-équivalent à wu^{-1}.

Si par contre pour tout $u \in U$, $e_1(u) = e_2(u)$, ceci entraîne l'existence d'un facteur propre de w avec une double factorisation zigzag. Ceci est vrai car $p' = e_1(u') = e_2(u') = q'$ aussi pour le mot u' de longueur maximale dans U. Alors, par la propriété 2 de $\mathscr{A}^\uparrow$, le chemin simple $s_1(u') : 1 \to 1$ qui dans la première factorisation passe par p' chevauchant la position $(w(u')^{-1}, u')$, coïncide avec le chemin simple $s_2(u')$ relatif à la deuxième factorisation.

On aura une situation du genre schématisé par



où la coïncidence des deux chemins $1 \to p'$ implique



Cette situation comporte l'existence de deux factorisations zigzag pour $w(u')^{-1}$, mises en évidence dans la figure. Elles sont distinctes, sauf dans le cas où le recollement des pas d'une factorisation à l'autre a formé une boucle non triviale, laquelle donne lieu à deux factorisations zigzag distinctes pour le mot qu'elle visite. ■

La proposition 3 nous fournit donc une façon effective d'établir si un langage X fini est un code zigzag ou non, étant donné que l'analyse peut se limiter aux classes $s\rho$, avec s dans R, et ces classes sont en nombre fini.

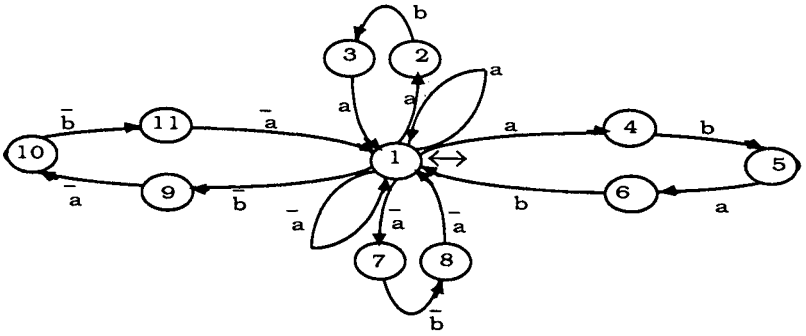
COROLLAIRE : *Si X est un langage fini qui n'est pas un code zigzag, on peut trouver le mot le plus court qui admet deux factorisations zigzag distinctes sur X.*

Preuve : Il suffit de regarder tous les mots s dans R tels que $s\rho$ contient deux états distincts p et q, s-équivalents, avec $p \xrightarrow{u} 1$ et $q \xrightarrow{u} 1$; et comparer en longueur les mots su, avec double factorisation, ainsi obtenus. ■

Exemple : Soit X le langage fini $X = a + aba + abab$.

X n'est pas un code zigzag car, par exemple, le mot *ababa* présente deux factorisations zigzag distinctes.

On construit l'automate $\mathscr{A}^{\uparrow}$ à partir de l'automate en pétales :



Les états 1 et 3 sont s-équivalents car $\mathscr{A}^{\uparrow}$ contient les chemins simples $1 \xrightarrow{a} 1$ et $3 \xrightarrow{a} 1$; si on trouve un mot s de R tel que $1,3 \in s\rho$, alors le mot sa a une double factorisation zigzag et X n'est pas un code zigzag. Nous allons calculer la fonction ρ pour cet automate, ainsi que la relation φ pour obtenir aussi l'automate 1-way qui calcule $X^{\uparrow}$.

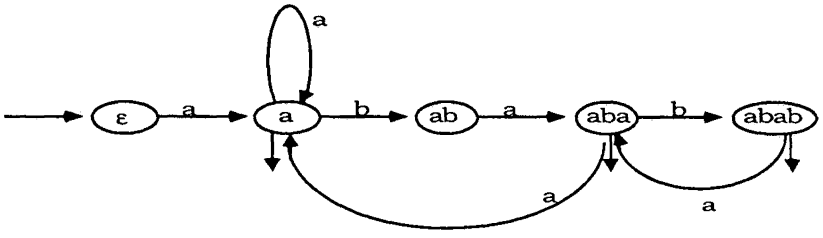
Pour tout $n>0$,

$a^n\rho=\{1,2,4\}$	$a^n\varphi$:	$1,8,11$	$\to\{1,2,4\}$
		les autres	$\to\varnothing$
$ab\,\rho=\{3,5\}$	$ab\,\varphi$:	$7,10$	$\to\{3,5\}$
		les autres	$\to\varnothing$
$aba\,\rho=\{1,2,4,6\}$	$aba\,\varphi$:	$1,8,9,11$	$\to\{1,2,4,6\}$
		les autres	$\to\varnothing$
$abaa\,\rho=\{1,2,4\}$	$abaa\,\varphi$:	$1,8,11$	$\to\{1,2,4\}$
		les autres	$\to\varnothing$
d'où $\vartheta(abaa)=\vartheta(a)$;			
$abab\,\rho=\{1,3,5\}$	$abab\,\rho$:	$1,7,10$	$\to\{3,5\}$
		les autres	$\to\varnothing$

Puisque les états 1 et 3 appartiennent tous deux à $abab\,\rho$, nous pouvons dire que *ababa* a deux factorisations zigzag différentes ; on peut donc arrêter le test et affirmer que X n'est pas un code zigzag.

De plus, *ababa* est le mot de longueur minimale ayant deux factorisations. Comme $\vartheta(ababa)=\vartheta(aba)$, l'analyse est terminée.

On a $R=\{\varepsilon, a, ab, aba, abab\}$ et l'automate 1-way équivalent au boustrophédon ci-dessus est :



3.1. Le cas infini

Dans le cas où X est un langage infini, nous ne pouvons pas répéter la même preuve, car il nous manque l'existence d'un automate dont la série associée est $(\underline{X})^{\uparrow}$, et qui jouit de la propriété 2 des automates en pétales.

En fait il nous faut une condition un peu moins forte pour $\mathscr{A}^{\uparrow}=(Q, 1, 1, F)$, qui est la suivante.

PROPRIÉTÉ 3 : Pour tout $p\in Q^{+}$, $p\neq 1$, et pour tout couple de chemins simples $c_1: 1\to p$, $c_2: 1\to p$ si $|c_1|$ est suffixe de $|c_2|$, alors $c_1=c_2$ et $|c_1|=|c_2|$.

Cette propriété suffit car on pourra conclure, comme dans la preuve de la proposition 3, que les chemins que l'on avait appellé $s_1(u')$ et $s_2(u')$, coïncident.

On a donc la

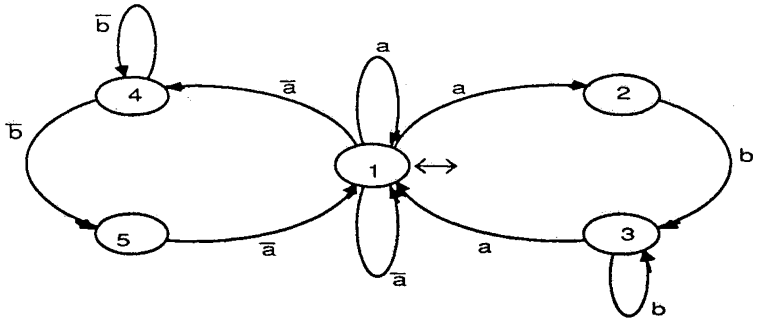
PROPOSITION 4 : *Soit X un langage reconnaissable sur un alphabet A ; s'il existe un automate boustrophédon $\mathscr{A}^{\uparrow}=(Q, 1, 1, F)$ tel que $\mathscr{A}^{\uparrow}=(\underline{X})^{\uparrow}$, qui vérifie la propriété* 3, *alors X est un code zigzag ssi pour tout $s\in R$, $s\rho$ ne contient pas deux états s-équivalents distincts.*

La proposition 4 admet comme cas particulier la proposition 3.

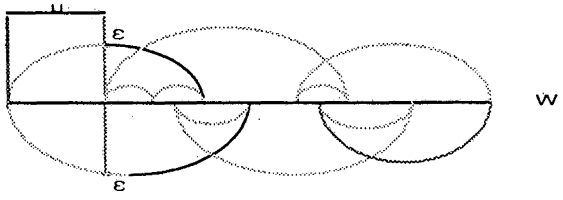
Exemple : Soit $X=a+ab^{+}a$.

X est un code infini, mais qui vérifie les conditions de la proposition 4. Nous pouvons construire l'automate boustrophédon suivant qui calcule bien

$X^{\uparrow}$:



Pour l'état 3, il existe plusieurs chemins simples de $1 \to 3$: ce sont les chemins $1 \xrightarrow{ab^n} 3$, pour tout $n>0$. Mais si ab^n est suffixe de ab^m, alors $n=m$. La propriété 3 est donc valable pour cet automate. En fait, si on se trouve avec une double factorisation d'un mot w causée par une situation de ce genre



on peut conclure, comme dans le cas fini, que les deux chemins $1 \to 3$, marqués dans la figure, coïncident et donc que la double factorisation de w, découle de celle d'un de ses facteurs.

On trouve pour tout $n>0$

$$a^n\rho = \{1,2\},\ ab^n\rho = \{3\} \qquad \text{et} \qquad aba\,\rho = a\,\rho.$$

Donc $R=\{\varepsilon, a, ab\}$ et aucune de ces classes ne contient à la fois deux états s-équivalents : 1 et 3 ou bien 2 et 3. Nous pouvons conclure que X est un code zigzag.

4. CONCLUSION

Dans cette étude, nous avons établi les bases du domaine des « zigzag ». Nous avons considéré le cas des langages finis et de certains langages infinis. Il nous reste encore à étudier plus en détail le cas des langages infinis.

Un autre problème très intéressant est le suivant. On se souvient que, lorsque nous avons construit l'automate boustrophédon $\mathscr{A}^{\uparrow}$, nous avons dit qu'il reconnaît $X^{\uparrow}$ avec la multiplicité exacte, dans le sens où un mot qui est un zigzag sur X, est produit par cet automate autant de fois que son nombre de factorisations zigzag. Par contre, l'automate 1-way correspondant, étant déterministe, produit chaque mot de $X^{\uparrow}$ une seule fois. Nous avons alors « perdu » toutes les informations concernant les multiplicités; sauf dans le cas où X est un code zigzag, lorsque l'automate 1-way respecte aussi la multiplicité (celle-ci étant égale à 1 !).

Il serait alors intéressant de trouver une façon d'associer, aussi aux ensembles qui ne sont pas des codes zigzag, un automate 1-way avec les bonnes multiplicités.

5. CONCLUSION

Je tiens à remercier le Professeur Dominique Perrin, qui a dirigé mon stage de D.E.A., et le Professeur Antonio Restivo qui m'a permis de venir en France.

BIBLIOGRAPHIE

1. J. Berstel et D. Perrin, Theory of Codes, *Academic Press*, 1985.
2. S. Eilenberg, Automata, Languages and Machines, *Academic Press*, *A*, 1974.
3. J. E. Hopcroft et J. D. Ullman, Introduction to Automata Theory, Languages and Computation, *Addison-Wesley*, 1979.
4. J. M. Howie, An introduction to Semigroup Theory, *Academic Press*, 1976.
5. J. P. Pécuchet, Automates boustrophédons, langages reconnaissables de mots infinis et variétés de semi-groupes, (*Thèse d'État*), L.I.T.P., mai 1986, I; ou bien, Automates boustrophédons, semi-groupe de Birget et monoïde inversif libre, *R.A.I.R.O. Inform. Théor.*, 1985, *19*, n° 1 p. 17-100.
6. J. C. Shepherdson, The reduction of Two-Way Automata to One-Way Automata, *I.B.M. J. Res.*, 1959, *3*, 2, p. 198-200; et dans E. F. Moore éd., Sequential Machines: Selected Papers, *Addison-Wesley*, 1964.