INFORMATIQUE THÉORIQUE ET APPLICATIONS

JEANNE DEVOLDER

Codes générateurs minimaux de langages de mots bi-infinis

Informatique théorique et applications, tome 34, n° 6 (2000), p. 585-596

<http://www.numdam.org/item?id=ITA_2000__34_6_585_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **34** (2000) 585–596

CODES GÉNÉRATEURS MINIMAUX DE LANGAGES DE MOTS BI-INFINIS

JEANNE DEVOLDER[1]

Abstract. In this paper we give two families of codes which are minimal generators of biinfinite languages: the family of very thin codes (which contains the rational codes) and another family containing the circular codes. We propose the conjecture that all codes are minimal generators.

Résumé. Dans cet article, nous exhibons deux familles de codes qui sont générateurs minimaux du langage de mots bi-infinis qu'ils engendrent : la famille des codes très minces (et parmi eux, les codes rationnels) et une autre famille qui contient les codes circulaires. Nous avançons la conjecture que tous les codes sont des générateurs minimaux.

Classification mathématique. 94A45.

1. INTRODUCTION

Contrairement au cadre de la dynamique symbolique sur un alphabet A, où l'on étudie essentiellement les fermés de A^Z invariants par le «shift»(les sous-shifts), nous nous intéressons ici aux ensembles de la forme ${}^{\omega}G^{\omega}$, qui, en général, ne sont pas fermés. Les sous-shifts (S) correspondent en fait à ce que nous appelons ici les bi-adhérences (*cf.* Sect. 5). Ils sont caractérisés par leurs facteurs finis : $S = Biadh(F(S))$, et les problèmes relatifs aux sous-shifts peuvent souvent être reformulés en termes de mots finis. Les ensembles de la forme ${}^{\omega}G^{\omega}$ ne sont pas, sauf s'ils sont fermés, caractérisés par leurs facteurs finis.

Mots clés : Mots bi-infinis, biω-générateur, code, code très mince, code rationnel, code circulaire, code précirculaire, code synchrone.

[1] Laboratoire de Statistique et Probabilités, F.R.E. CNRS 2222, Université des Sciences et Technologies de Lille, bâtiment M2, 59655 Villeneuve-d'Ascq, France ;
e-mail : Jeanne.Devolder@univ-lille1.fr

Le problème qui nous occupe est la recherche et l'étude des générateurs d'un langage de mots bi-infinis L donné, c'est-à-dire des langages $G \subseteq A^+$ tels que $L = {}^{\omega}G^{\omega}$. Des résultats ont été montrés pour les langages rationnels de mots bi-infinis [5]. En particulier, on sait décider si un langage rationnel L a des générateurs et si L a des générateurs finis ; dans le cas où le langage rationnel L a des générateurs, il a un nombre fini et calculable de générateurs maximaux ; ces générateurs maximaux sont rationnels et constructibles, et tout générateur est inclus dans un générateur maximal.

Si $L \subset {}^{\omega}A^{\omega}$ a des générateurs, certains d'entre eux sont plus intéressants. C'est le cas des générateurs rationnels, puisqu'il suffit d'un automate fini pour les mémoriser. C'est le cas des codes générateurs, en particulier ceux qui permettent de coder de façon unique les mots bi-infinis (les bi-ω codes [6]). C'est aussi le cas des générateurs minimaux, puisqu'il n'y en a pas de plus «petits ».

Hélas, comme dans le cas infinitaire, nous savons peu de choses sur les générateurs minimaux. Nous ne savons pas s'il en existe, ni si leur existence est décidable, ni *a fortiori* si tout générateur contient un générateur minimal.

Dans le cas infinitaire, étant donné un générateur rationnel, on sait décider s'il est minimal. On sait aussi que les codes générateurs sont des générateurs minimaux. Dans le cas bi-infinitaire, nous ne savons rien de tel. C'est la raison pour laquelle, dans cet article, nous nous intéressons aux codes vus comme générateurs de langages de mots bi-infinis.

Dans le cas infinitaire, un code C, vu comme générateur de C^{ω}, est toujours un générateur minimal. Cela est dû au fait que tout mot infini u^{ω} a une C-factorisation unique si $u \in C^+$ [9]. Cette propriété (P) n'a pas d'équivalent dans le cadre bi-infinitaire, et nous ne savons pas si tout code C est générateur minimal de ${}^{\omega}C^{\omega}$.

Nous avons déterminé deux familles de codes qui sont des générateurs minimaux. La première famille est celle des codes C pour lesquels il existe un mot primitif $u_0 \in C^+$ tel que ${}^{\omega}u_0{}^{\omega}$ ait une seule C-factorisation; cette famille contient celle des codes précirculaires, lesquels vérifient une propriété analogue à (P). Les codes circulaires sont des cas particuliers de codes précirculaires.

La deuxième famille est plus inattendue : c'est celle des codes très minces, famille importante puisqu'elle contient tous les codes minces et maximaux, et tous les codes rationnels.

Nous appliquons les résultats précédents aux codes générateurs de ${}^{\omega}A^{\omega}$ (pour un alphabet A quelconque) et donnons quelques résultats complémentaires dans ce cadre. Nous comparons la famille des générateurs de ${}^{\omega}A^{\omega}$ avec celle des générateurs de A^{ω}, et constatons, comme pour d'autres problèmes [2,5,8,10,13], la plus grande complexité du cas bi-infinitaire. Pour terminer, nous concluons sur deux conjectures, à savoir que tout code C est générateur minimal de ${}^{\omega}C^{\omega}$, et que tout code générateur de ${}^{\omega}A^{\omega}$ est un code maximal.

2. DÉFINITIONS ET NOTATIONS

Les définitions et notations utilisées ici sont en général celles de la référence [1].

Soit $\boldsymbol{A}$ un alphabet, $\boldsymbol{A^*}$ l'ensemble des **mots finis** sur A, ε le mot vide et $\boldsymbol{A^+}$ le langage $A^* \setminus \varepsilon$, $\boldsymbol{A^\omega}$ l'ensemble des **mots infinis (à droite)** sur A. Les **mots bi-infinis** [2, 13] sur A sont les classes d'équivalence pour la relation de décalage sur A^Z :

$$(u_n)_{n\in Z} \sim (v_n)_{n\in Z} \iff \exists p \in Z\ \forall n \in Z\ u_n = v_{n+p}.$$

On peut considérer les mots bi-infinis comme les orbites des éléments de A^Z dans le système dynamique (A^Z, σ), où σ est l'opérateur de décalage (le «shift»).

Le mot bi-infini défini par $(u_n)_{n\in Z}$ sera noté $\ldots u_{-1}u_0u_1\ldots$ L'ensemble des mots bi-infinis sur A est noté ${}^\omega\boldsymbol{A}^\omega$. Un mot bi-infini w est dit **périodique** (resp. **ultimement périodique**) s'il peut s'écrire $w = {}^\omega v^\omega$ (resp. ${}^\omega vuw^\omega$) où v (resp. u, v, w) $\in A^+$.

Soit C un sous-ensemble de A^+. Un mot bi-infini w est dit **C-périodique** (resp. **C-ultimement périodique**) s'il peut s'écrire $w = {}^\omega v^\omega$ (resp. ${}^\omega vuw^\omega$) où v (resp. u, v, w) $\in C^+$.

Un mot $u \in A^+$ est dit **primitif** si $z = u^n$ seulement pour $n = 1$.

Soit $B \subseteq A^+$, on définit : $\boldsymbol{B^*} = \{v_1 \ldots v_n \mid n \geq 0\ \ v_i \in B \text{ pour } 1 \leq i \leq n\}$, $\boldsymbol{B^\omega} = \{v_0v_1v_2\ldots \mid v_i \in B \text{ pour } i \geq 0\}$, ${}^\omega\boldsymbol{B^\omega} = \{\ldots v_{-1}v_0v_1v_2\ldots \mid v_i \in B, i \in Z\}$.

Les définitions classiques relatives aux langages de mots finis (codes, générateurs, bases, ...) ont été pour la plupart étendues à des langages de mots infinis, et à des langages de mots bi-infinis [2–6, 8, 12, 13] ... Nous ne définirons ici que les notions utiles dans la suite.

Un **générateur** de ${}^\omega B^\omega$ est un langage G de mots finis tel que ${}^\omega G^\omega = {}^\omega B^\omega$. Un générateur G de ${}^\omega B^\omega$ est dit **minimal** s'il ne contient pas strictement d'autres générateurs de ${}^\omega B^\omega$.

Une **B-factorisation** d'un mot $v \in B^*$ est une suite finie de mots de B : $(v_1, \ldots, v_n)$ telle que $v = v_1 \ldots v_n$. Une **B-factorisation** d'un mot $v \in {}^\omega B^\omega$ est une classe d'équivalence, pour la relation de décalage $\sim$, définie cette fois sur B^Z, d'une suite $(v_i)_{i\in Z}$ de mots de B, vérifiant $v = \ldots v_{-1}v_0v_1v_2\ldots$

Un langage $C \subseteq A^+$ est un **code** si tout mot de C^+ a une seule C-factorisation. Étant donné un code C et $u \in C$, le code $C \setminus \{u\}$ est noté $\boldsymbol{C_u}$.

La longueur d'un mot u est notée $|\boldsymbol{u}|$. Le nombre de lettres a d'un mot u est noté $|\boldsymbol{u}|_{\boldsymbol{a}}$. Un mot $u \in A^*$ est dit **facteur** d'un mot $v \in A^*$ (ou d'un mot v infini ou bi-infini) s'il existe des mots x et y tels que $v = xuy$. L'ensemble des facteurs des mots d'un langage L est noté $\boldsymbol{F(L)}$.

Nous nous intéresserons ici plus particulièrement à certaines familles de codes. Les codes **minces** sont les codes C tels que A^* n'est pas inclus dans $F(C)$. Les codes **très minces** sont les codes C tels que C^* n'est pas inclus dans $F(C)$. Les codes **complets** sont les codes C tels que A^* est inclus dans $F(C^*)$.

3. Codes précirculaires

Un langage $C \subseteq A^+$ est un **code précirculaire** [7] si

$$\forall\ n,p \geq 1\ \forall u_0,\ldots,u_{n-1},v_0,\ldots,v_{p-1} \in C\ \forall\ t,s \in A^* \text{ tels que } v_0 = ts$$

$$(u_0 \ldots u_{n-1} = sv_1 \ldots v_{p-1}t \implies n = p \text{ et } \exists h\ \forall i\ u_i = v_{(i+h)\ mod\ p}).$$

Les codes suivants sont précirculaires :

- les **biω-codes**, c'est-à-dire les codes C tels que tout mot de ${}^{\omega}C^{\omega}$ a une unique C-factorisation (définition p. 368 et preuve p. 372 dans [6]).
- les **codes circulaires** : un langage C est un **code circulaire** si

$$\forall\ n,p \geq 1\ \ \forall u_0,\ldots,u_{n-1},v_0,\ldots,v_{p-1} \in C$$

$$\forall t \in A^*\ \ \forall s \in A^+ \text{ tels que } v_0 = ts$$

$$(u_0 \ldots u_{n-1} = sv_1 \ldots v_{p-1}t\ \ \Rightarrow\ \ n = p,\ t = \varepsilon \text{ et } \forall i\ u_i = v_i)$$

(définition p. 322 dans [1] et preuve immédiate).

Citons quelques codes circulaires intéressants :

- les codes **à délai de synchronisation borné** : un code C est un code à délai de synchronisation borné si

$$\exists m > 0\ \ \forall f,f' \in A^*\ \ \forall v,v' \in C^m\ (fvv'f' \in C^*\ \Rightarrow\ fv,v'f' \in C^*)$$

(définition et preuve dans [11] ou dans [8]) ;

- les codes **semi-Dyck** D'_n à n couples de lettres. Rappelons que D'_n est la base de l'ensemble des mots bien parenthésés pour la famille des couples de lettres considérée.

Proposition 3.1. *Les codes* **semi-Dyck** D'_n *à n couples de lettres sont circulaires.*

Preuve. Considérons d'abord le cas de D'_1. Appelons (a,a') le couple de lettres définissant D'_1. Considérons la fonction ν définie par : $\nu(x) = |x|_a - |x|_{a'}$. Cette fonction vérifie :

$\forall x \in {D'_1}^*\ \ \nu(x) = 0$;
$\forall x\ \forall y\ \ \nu(xy) = \nu(x) + \nu(y)$;
$\forall x\ \forall y$ tels que $xy \in D'_1$ on a $\nu(x) \geq 0$;
$\forall x\ \forall y$ tels que $xy \in D'_1$ on a $\nu(x) > 0$ si $x \neq \epsilon$ et $x \notin D'_1$.

Considérons $n,p \geq 1\ \ u_0,\ldots,u_{n-1},v_0,\ldots,v_{p-1} \in D'_1\ \ t \in A^*\ \ s \in A^+$ tels que $v_0 = ts$ et $u_0 \ldots u_{n-1} = sv_1 \ldots v_{p-1}t$. On a alors $\nu(s) + \nu(t) = 0$, $\nu(s) \geq 0$ et $\nu(t) \geq 0$, donc $\nu(s) = \nu(t) = 0$ et finalement $t = \varepsilon$ et $s = v_0$. On conclut en remarquant que D'_1 est un code (préfixe).

Dans le cas de D'_n, on définit les fonctions ν_l, analogues à ν, associées aux couples l de lettres définissant D'_n. Ces fonctions ont les mêmes propriétés que ν, sauf la dernière qui est à remplacer par : $\forall x\ \forall y$ tels que $xy \in D'_1$ on a : $\sum_l \nu_l(x)$

$= 0 \Rightarrow x = \epsilon$ ou $x \in D'_1$. Il suffit de modifier très légèrement la démonstration faite pour D'_1 pour l'adapter à D'_n. □

La proposition suivante, démontrée dans [7] (Th. 5.5) va nous permettre d'obtenir une première famille de codes qui sont des générateurs minimaux.

Proposition 3.2. [7] *Un langage C inclus dans A^+ est un code précirculaire si et seulement si tout mot bi-infini périodique a au plus une C-factorisation.*

Théorème 3.3. *Tout code précirculaire C est un générateur minimal de ${}^{\omega}C^{\omega}$.*

Preuve. Pour tout u de C, le mot périodique ${}^{\omega}u^{\omega}$ a une C-factorisation unique : ${}^{\omega}(u)^{\omega}$, donc n'appartient pas à ${}^{\omega}C_u{}^{\omega}$, et C_u n'engendre pas ${}^{\omega}C^{\omega}$. □

On peut généraliser ce théorème de la façon suivante :

Théorème 3.4. *Tout code C pour lequel il existe un mot primitif u_0 de C^+ tel que ${}^{\omega}u_0{}^{\omega}$ ait une seule C-factorisation est un générateur minimal de ${}^{\omega}C^{\omega}$.*

Preuve. Soit $u \in C$. On va montrer que ${}^{\omega}C_u{}^{\omega} \neq {}^{\omega}C^{\omega}$. Supposons le contraire. Le mot : ${}^{\omega}u_0 u u_0^{\omega}$ de ${}^{\omega}C^{\omega}$ a alors une C_u-factorisation. Puisque C_u est un code, cette C_u-factorisation est C_u-ultimement périodique [7]. Donc ${}^{\omega}u_0 u u_0^{\omega} = {}^{\omega}v x w^{\omega}$ où $v, x, w \in C_u^*$. On peut choisir x tel que $x = s' u_0^n u u_0^m p$ où $n, m \in N$, s' est un suffixe de u_0 et p un préfixe de u_0, tous deux différents de u_0. Montrons que $s' = p = \epsilon$. Soit s tel que $ps = u_0$. On a : $s u_0^{\omega} = (sp)^{\omega} = w^{\omega}$. Puisque u_0 est primitif, sp l'est aussi et donc il existe k tel que $w = (sp)^k$. Donc ${}^{\omega}u_0{}^{\omega} = {}^{\omega}(sp)^{\omega} = {}^{\omega}w^{\omega}$ où $w \in C_u^*$. Il s'ensuit que ${}^{\omega}u_0{}^{\omega}$ a deux C-factorisations : celle n'utilisant que des u_0, et celle utilisant la C_u-factorisation de w. Puisque u_0 est primitif, ces C-factorisations sont distinctes, sauf si $p = \epsilon$. Donc $p = \epsilon$; de même, on montre que $s' = \epsilon$. On a donc $x = u_0^n u u_0^m$; ce mot x de ${C_u}^*$ a donc aussi une C-factorisation utilisant le mot u, ce qui est absurde puisque C est un code. Donc ${}^{\omega}C_u{}^{\omega} \neq {}^{\omega}C^{\omega}$.

Dans le cas où $u = u_0$ la démonstration précédente est encore valable (mais peut être simplifiée). Donc ${}^{\omega}C_u{}^{\omega} \neq {}^{\omega}C^{\omega}$ pour tout u de C et C est un générateur minimal de ${}^{\omega}C^{\omega}$. □

4. Codes très minces

Un code C est **très mince** si C^* n'est pas inclus dans $F(C)$ [1].

Avant de démontrer le théorème principal de cette section (Th. 4.8), nous allons donner quelques exemples de codes très minces, et d'abord une propriété de ces codes.

4.1. Exemples de codes très minces

Proposition 4.1. *Un code C est très mince si et seulement si il existe $u \in C$ tel que $C^* \not\subseteq F(C_u)$.*

Preuve. Supposons que C ne soit pas très mince ; pour tous $v \in C^+$ et $u \in C$, on a $vu \in F(C)$ donc $vu \in F(C_u)$ et finalement $v \in F(C_u)$; donc, s'il existe $u \in C$, $C^* \not\subseteq F(C_u)$, le code C est très mince. La réciproque est évidente. □

Corollaire 4.2. *Les codes C tels qu'il existe $u \in C$, $C^* \not\subseteq F(C_u^*)$ sont des codes très minces.*

Dans ce dernier cas, il est clair que C_u n'engendre pas ${}^\omega C^\omega$ (car ${}^\omega C^\omega = {}^\omega C_u{}^\omega$ entraîne $C^* \subseteq F(C_u^*)$) ; mais rien ne laisse présager le résultat du théorème 4.8, à savoir que les autres éléments de C sont aussi indispensables que u.

Proposition 4.3. *(p. 227 dans [1]) Les codes minces et maximaux (ou minces et complets) sont très minces.*

Dans ce cas, le théorème 4.8 peut se démontrer de façon simple [8].

Proposition 4.4. *(p. 224 dans [1]) Les codes reconnaissables, en particulier les codes rationnels sont très minces.*

Pour le prochain exemple, nous avons besoin d'une définition. On dit que (x, y) est une **paire synchronisante** pour un code C si x et y appartiennent à C^* et vérifient :

$$\forall\, u, v \in A^* \quad (uxyv \in C^* \implies ux, yv \in C^*)$$

(définition p. 240 dans [1]).

Clairement, tout code à délai de synchronisation borné possède des paires synchronisantes. Mais la réciproque est fausse :

Exemple 4.1. *Le code $C = a + b + ab^*d$ possède une paire synchronisante : (a, a), mais, pour tout m, le couple (b^m, b^m) n'en est pas une puisque le mot $ab^m b^m d$ appartient à C.*

Proposition 4.5. *Les codes ayant une* **paire synchronisante** *sont très minces.*

Preuve. Montrons d'abord qu'on peut supposer que la paire synchronisante (x, y) est constituée d'éléments de C^+.

Si $x = \epsilon$ et $y \in C^+$, le couple (y, y) est une paire synchronisante ; en effet, si $uyyv \in C^*$, alors u et yyv appartiennent à C^* (puisque (ϵ, y) est une paire synchronisante), et aussi y et yv (pour la même raison) ; donc uy et yv appartiennent à C^*. Le raisonnement est analogue pour $y = \epsilon$ et $x \in C^+$.

Si $x = y = \epsilon$, on voit clairement que tout mot de C ne peut avoir qu'une lettre ; et alors, tout couple de mots de C est une paire synchronisante.

Considérons un code C ayant une paire synchronisante (x, y) constituée d'éléments de C^+. Pour montrer que C est très mince, il suffit de prouver que xy, qui appartient à C^+, n'est pas dans $F(C)$. Supposons le contraire, alors il existe u et v tels que $uxyv \in C$; mais alors ux et yv appartiennent à C^* et sont différents de ϵ, ce qui contredit le fait que C est un code. □

Remarque. Dans [1], on définit p. 239 un code **synchrone** comme un code très mince de degré 1 ; ce qui veut dire qu'un code C très mince est synchrone s'il existe un mot de C^* n'appartenant pas à $F(C)$ et dont toutes les C-interprétations sont adjacentes. La proposition 6.2 dans [1] dit que, pour les codes très minces, il y a équivalence entre être synchrone et avoir une paire synchronisante ; finalement, d'après ce que nous venons de voir, nous avons :

Proposition 4.6. *Les notions de code synchrone et de code ayant une paire synchronisante coïncident.*

Dans le cas d'un code ayant une paire synchronisante, le théorème 4.8 peut aussi se démontrer de façon simple [8].

4.2. Codes très minces vus comme générateurs

Revenons maintenant au problème qui motive cette section. La proposition suivante est essentielle pour démontrer le résultat principal. Elle nous assure qu'un code très mince B est maximal parmi les codes C vérifiant $F(C^*) \subseteq F(B^*)$.

Proposition 4.7. *Soient B et C deux codes tels que $B \subseteq C$ et $F(B^*) = F(C^*)$. Si B ou C est très mince alors $B = C$.*

Preuve. Si B n'est pas très mince, B^* est inclus dans $F(B)$. On a alors : $C^* \subseteq F(C^*) = F(B^*) \subseteq F(B) \subseteq F(C)$, et donc C n'est pas très mince. Par conséquent, si C est très mince, B l'est aussi.

Si le code B est complet et très mince, il est maximal et donc $B = C$.

Supposons maintenant que le code B soit très mince mais ne soit pas complet. La suite de la démonstration s'appuie sur l'étude des monoïdes de relations non ambiguës qui peuvent être construits à partir des codes ([1], Chap. IV : Automata). À partir d'un automate (non nécessairement fini) émondé non ambigu reconnaissant B, on peut construire un automate émondé, non ambigu, de la forme $(Q, 1, 1, R)$ reconnaissant B^*, et considérer le morphisme de représentation φ associé : pour $a \in A$, $\varphi(a)$ est la matrice m de $\{0,1\}^{Q\times Q}$ telle que $m_{p,q} = 1$ si et seulement si $q \in R(p, a)$. Le stabilisateur $Stab(q)$ d'un état q est l'image par φ de l'ensemble des mots finis qui admettent une lecture de q à q. Puisque B est un code très mince non complet, $M = \varphi(A^*)$ est un monoïde transitif, non ambigu, de rang minimal $r(M)$ fini, contenant la matrice nulle ([1], pp. 189 et 224). Il existe dans $Stab(1)$ un idempotent e de rang $r(M)$ et $e = \varphi(x)$ où $x \in B^*$. Si $B \neq C$, il existe $y \in C \setminus B$. Le mot xyx appartient à $F(C^*) = F(B^*)$ donc $\varphi(xyx)$ n'est pas nul. $\varphi(xyx) = e\varphi(y)e$ est dans le groupe des unités de eMe. Ce groupe est fini ([1], p. 221), donc il existe $n > 0$ tel que $(e\varphi(y)e)^n = e$. Donc $\varphi((xyx)^n) = e$. Comme e appartient à $Stab(1)$ le mot $(xyx)^n$ est dans B^*. Comme ce mot a une C-factorisation utilisant y, $B \cup \{y\}$, et, *a fortiori*, C ne sont pas des codes. □

Remarquons que la proposition précédente devient fausse quand on remplace l'hypothèse : «B ou C très mince »par l'hypothèse : «B ou C mince».

Exemple 4.2. *Considérons* $C = D'_2 = aD'^*_2a' + bD'^*_2b'$ *le code semi-Dyck sur deux couples de lettres* (a, a') *et* (b, b'), *et le code* $B = aD'^*_2a'$. *Les codes* B *et* C *sont minces et vérifient :* $B \subseteq C$, $B \neq C$ *et* $F(B^*) = F(C^*)$.

Preuve. D'une part, on a : $C^* \subseteq F(B)$ donc $F(B^*) = F(C^*)$. D'autre part, on vérifie facilement que B et C sont minces mais pas très minces : tout u de C^* est facteur du mot de B : aua', mais le mot ab' n'est facteur d'aucun mot de C. □

Théorème 4.8. *Tout code* C *très mince est un générateur minimal de* ${}^\omega C^\omega$.

Preuve. Soit C un code très mince. Pour tout $u \in C$, le code C_u ne peut vérifier $F({C_u}^*) = F(C^*)$ puisque $C_u \subseteq C$, $C_u \neq C$ (proposition 4.7). Donc il existe un mot v de $F(C^*)$ n'appartenant pas à $F({C_u}^*)$. On peut prolonger v en un mot de ${}^\omega C^\omega$. Ce dernier mot ne se factorise pas sur C_u, donc C_u n'engendre pas ${}^\omega C^\omega$. Le code C est donc un générateur minimal de ${}^\omega C^\omega$. □

En corollaire du théorème précédent nous obtenons le résultat important suivant :

Théorème 4.9. *Tout code* C *rationnel est un générateur minimal de* ${}^\omega C^\omega$.

5. Cas particulier des codes générateurs de ${}^\omega A^\omega$

Avant d'énoncer la proposition suivante qui va nous inciter à étudier la maximalité des codes générateurs de ${}^\omega A^\omega$, nous rappelons quelques définitions et propriétés. Un langage $G \subseteq A^*$ est dit **complet** si $A^* \subseteq F(G^*)$ (ou, ce qui est équivalent, $A^* = F(G^*)$). On appelle **biadhérence** de G, pour $G \subseteq A^*$, l'ensemble $Biadh(G) = \{w \in {}^\omega A^\omega \mid F(w) \subseteq F(G)\}$ [10]. On dit qu'un langage $L \subseteq {}^\omega A^\omega$ est une **biadhérence** s'il existe $G \subseteq A^*$ tel que $L = Biadh(G)$. On démontre que $L \subseteq {}^\omega A^\omega$ est une biadhérence si et seulement si $L = Biadh(F(L))$ (Cor. 3.2.3 dans [5]). Il s'ensuit, pour le langage ${}^\omega G^\omega$ engendré par $G \subseteq A^+$, que ${}^\omega G^\omega$ est une biadhérence si et seulement si ${}^\omega G^\omega = Biadh(F(G^*))$.

Proposition 5.1. *Tout langage générateur de* ${}^\omega A^\omega$ *est un langage complet. Tout langage complet qui engendre une biadhérence engendre* ${}^\omega A^\omega$.

Preuve. Clairement, si ${}^\omega G^\omega = {}^\omega A^\omega$, on a : $A^* \subseteq F(G^*)$, donc G est complet.
Si G est un langage qui engendre une biadhérence, on a : ${}^\omega G^\omega = Biadh(F(G^*))$ d'après le rappel ci-dessus. Si G est de plus complet, on a : ${}^\omega G^\omega = Biadh(F(G^*))$ $= Biadh(A^*) = {}^\omega A^\omega$. □

Si l'alphabet A est fini, il existe des codes maximaux finis, l'alphabet A lui-même, par exemple. Si l'alphabet A est infini, il n'existe pas de codes maximaux finis. En effet, l'ensemble des lettres figurant dans les mots d'un code fini C (l'«alphabet» du code) est fini. L'ensemble $C \cup \{a\}$, où a est une lettre de A n'appartenant pas à l'alphabet de C, est encore un code. Donc le code C n'est pas maximal.

Théorème 5.2 (Cas où A est fini). *Tout code maximal fini sur un alphabet fini A est un générateur minimal de ${}^{\omega}A^{\omega}$.*

Preuve. D'une part, un langage fini engendre une biadhérence (Cor. 3.2.8 dans [5]) ; d'autre part, un code maximal fini est complet et très mince [1] ; on applique alors le théorème 4.8 et la proposition 5.1. □

Remarques. Cependant, il existe des codes maximaux (infinis) qui n'engendrent pas ${}^{\omega}A^{\omega}$; c'est par exemple le cas (pour $A = \{a, b\}$) du code de Dyck D_1, base du monoïde des mots ayant autant de a que de b, ou encore celui du code rationnel $b + ab^*a$, dont aucun n'engendre le mot ${}^{\omega}bab^{\omega}$.

Mais il existe aussi des codes maximaux infinis qui engendrent ${}^{\omega}A^{\omega}$. L'existence de codes infinis non maximaux qui engendrent ${}^{\omega}A^{\omega}$ reste une question ouverte.

Exemple 5.1. *Pour $A = \{a, b\}$, le code rationnel : $C = a + b^2 + ab(b^2)^*a + ab(b^2)^*ab$ est un code maximal qui engendre ${}^{\omega}A^{\omega}$.*

Preuve. Montrons que C est un code. On voit clairement qu'un mot de A^+ qui a deux C-factorisations commençant par des mots de C distincts appartient à $ab(b^2)^*aA^*$. En comptant le nombre de b, on voit alors qu'il est de la forme $ab(b^2)^nab^p$ où p est à la fois pair et impair, ce qui est impossible.

Le code C engendre ${}^{\omega}A^{\omega}$; en effet, dans un mot de ${}^{\omega}A^{\omega}$, seuls les b^{2n+1} pourraient poser problème, et il n'en est rien, car : $ab^{2n+1}a.b^{2p}.a$ et $ab^{2n+1}ab.b^{2p}.a$ appartiennent à C^*. □

De plus, ${}^{\omega}A^{\omega}$ a aussi des générateurs minimaux qui ne sont pas des codes.

Exemple 5.2. *Pour $A = \{a, b\}$, le langage fini $G_1 = a + aba + bab + b^2 + b^3$ est un générateur minimal de ${}^{\omega}A^{\omega}$ et ce n'est pas un code.*

Preuve. Clairement, G_1 n'est pas un code. Cependant, il engendre ${}^{\omega}A^{\omega}$; en effet, dans un mot de ${}^{\omega}A^{\omega}$, seuls les b isolés pourraient poser problème, et il n'en est rien, car : $aba, a.bab.a, aba.bab.a, \ldots$ appartiennent à G_1^*. Le langage G_1 est un générateur minimal de ${}^{\omega}A^{\omega}$; en effet,

- a ne peut être ôté, car ${}^{\omega}a^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- aba ne peut être ôté, car ${}^{\omega}aba^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- bab ne peut être ôté, car ${}^{\omega}ababa^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- b^2 ne peut être ôté, car ${}^{\omega}ab^2a^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- b^3 ne peut être ôté, car ${}^{\omega}ab^3a^{\omega}$ appartient à ${}^{\omega}A^{\omega}$. □

Ces remarques montrent l'énorme différence existant entre ${}^{\omega}A^{\omega}$ et A^{ω}. En effet, dans le cas d'un alphabet fini A, tout générateur de A^{ω} contient un générateur minimal, et les générateurs minimaux de A^{ω} sont les codes préfixes maximaux finis [12].

Les théorèmes 3.4 et 4.8 nous donnent quelques compléments intéressants :

Théorème 5.3. *Tout code mince qui engendre ${}^{\omega}A^{\omega}$ est un générateur minimal de ${}^{\omega}A^{\omega}$, et est un code maximal.*

Preuve. Tout code qui engendre ${}^{\omega}A^{\omega}$ est complet (Prop. 5.1). Un code mince et complet est maximal et très mince [1], et le théorème 4.8 s'applique. □

Théorème 5.4. *Tout code C qui engendre ${}^{\omega}A^{\omega}$ et pour lequel il existe un mot primitif u_0 de C^+ tel que ${}^{\omega}u_0{}^{\omega}$ ait une seule C-factorisation est un générateur minimal de ${}^{\omega}A^{\omega}$, et est un code maximal.*

Preuve. Le théorème 3.4 s'applique ; il reste à montrer que C est un code maximal.

Pour tout y n'appartenant pas à C, posons $B = C \cup \{y\}$ et montrons que B n'est pas un code. Le mot : ${}^{\omega}u_0 y u_0^{\omega}$ de ${}^{\omega}A^{\omega} = {}^{\omega}C^{\omega}$ a une C-factorisation. Nous développons maintenant le même argument que dans la preuve du théorème 3.4. Comme C est un code, toute C-factorisation de ${}^{\omega}u_0 y u_0^{\omega}$ est ultimement périodique. Le mot u_0 étant primitif et ${}^{\omega}u_0{}^{\omega}$ n'ayant qu'une seule C-factorisation, il existe p et q tels que $u_0^p y u_0^q \in C^*$. Ce mot $u_0^p y u_0^q$ de C^* a une C-factorisation où n'apparaît pas le mot y et une B-factorisation: $(u_0, \ldots, y, \ldots, u_0)$ où apparaît y. Donc B n'est pas un code. □

6. Conclusion

Nous avons étudié deux familles de codes. Pour $Card(A) > 1$, les exemples suivants montrent qu'aucune de ces familles n'est incluse dans l'autre.

Exemple 6.1. *Le code semi-Dyck à un couple de lettres D_1' est circulaire mais n'est pas très mince.*

Preuve. Nous avons déjà vu que le code D_1' est circulaire. Le code D_1' n'est pas très mince ; en effet, si (a, a') désigne le couple de lettres de D_1', tout mot u de $D_1'^*$ est facteur du mot de D_1' : aua'. □

Exemple 6.2. *Pour le code très mince A^2, il n'existe pas de mot primitif u_0 de C^+ tel que ${}^{\omega}u_0{}^{\omega}$ ait une seule C-factorisation.*

Preuve. On voit facilement, à l'aide de la proposition 3.2, que le code très mince A^2 n'est pas précirculaire, car le mot bi-infini périodique ${}^{\omega}(ab)^{\omega}$ a deux A^2-factorisations : $\ldots ab.ab.ab \ldots$ et $\ldots ba.ba.ba \ldots$ Démontrons qu'il ne vérifie pas non plus les hypothèses du théorème 3.4. Soit $u = a_1 \ldots a_n$ un mot tel que ${}^{\omega}u^{\omega}$ a une seule A^2-factorisation. Cela signifie que les A^2-factorisations qui proviennent de $(\ldots, a_1a_2, a_3a_4, \ldots)$ et $(\ldots, a_2a_3, a_4a_5, \ldots)$ sont identiques. Notons ${}^{\omega}u^{\omega}$ sous la forme $\ldots a_{-1}a_0a_1 \ldots a_n a_{n+1} \ldots$ Il existe $p \in Z$, p impair tel que $a_1a_2 = a_{p+1}a_{p+2}$, $a_3a_4 = a_{p+3}a_{p+4}$, $\ldots$, $a_ka_{k+1} = a_{p+k}a_{p+k+1}$ pour tout k. Donc ${}^{\omega}u^{\omega} = {}^{\omega}(a_1 \ldots a_p)^{\omega}$. Si le mot $u = a_1 \ldots a_n$ est primitif, $a_1 \ldots a_p$ est une puissance de u. Donc la longueur de u est impaire et u n'appartient pas à $(A^2)^+$. □

Nous avons aussi :

Proposition 6.1. *Il existe un code C, bipréfixe et maximal, générateur minimal de ${}^{\omega}C^{\omega}$, tel que ${}^{\omega}C^{\omega} \neq {}^{\omega}A^{\omega}$, pour lequel tout mot périodique de ${}^{\omega}C^{\omega}$ a plusieurs C-factorisations, et qui n'est pas très mince.*

Exemple 6.3. *Le* **code de Dyck** D_1 *sur* $A = \{a, b\}$, *base du monoïde des mots ayant autant de* a *que de* b, *en est un exemple.*

Preuve. Le code D_1 est bipréfixe. Rappelons qu'il est aussi maximal (p. 65 dans [1]) : pour tout mot $v \notin D_1$ tel que $|v|_a > |v|_b$, le mot $vb^{(|v|_a - |v|_b)}v$ a plusieurs $(D_1 + v)$-factorisations.

Le code D_1 n'est pas mince, car tout mot $v \in A^*$ est facteur du mot de D_1 : $b^{|v|_a+1}va^{|v|_b+1}$.

Étudions les mots périodiques de ${}^{\omega}D_1{}^{\omega}$. Comme D_1 est un code, tout mot périodique de ${}^{\omega}D_1{}^{\omega}$ est de la forme ${}^{\omega}u^{\omega}$ où $u \in D_1^+$ (Th. 5.2 dans [7]). Pour toute écriture de u sous la forme $u = xy$ où $x, y \in A^*$, considérons $\nu_x = |x|_b - |x|_a$. Posons $x = x'y' = x''y''$ pour x' et x'' tels que $\nu_{x'}$ soit le minimum et $\nu_{x''}$ soit le maximum des ν_x. Considérons les conjugués de u : $u' = y'x'$ et $u'' = y''x''$. Tous les mots de la D_1-factorisation de u' (resp. u'') commencent par b (resp. a) et se terminent par a (resp. b). Ces D_1-factorisations donnent donc deux D_1-factorisations distinctes de ${}^{\omega}u^{\omega} = {}^{\omega}u'^{\omega} = {}^{\omega}u''^{\omega}$.

Clairement, ${}^{\omega}A^{\omega} \neq {}^{\omega}D_1{}^{\omega}$. Montrons que D_1 est un générateur minimal de ${}^{\omega}D_1{}^{\omega}$. Pour tout mot $u \in D_1$ on peut considérer le mot ${}^{\omega}(ab)u(aub)(ab)^{\omega}$ de ${}^{\omega}D_1{}^{\omega}$, dont toutes les D_1-factorisations utilisent u . En effet, comme D_1 est préfixe, le mot ${}^{\omega}(ab)u(aub)(ab)^{\omega}$ n'a que les deux D_1-factorisations suivantes : $\ldots ab.ab.u.$ (D_1-factorisation de $aub).ab.ab\ldots$ et $\ldots ba.ba.$ (D_1-factorisation de $bua).u.ba.ba\ldots$ On en déduit que D_1 est un générateur minimal de ${}^{\omega}D_1{}^{\omega}$. □

Les théorèmes démontrés, l'exemple précédent et l'étude des codes générateurs de ${}^{\omega}A^{\omega}$ nous suggèrent la conjecture suivante :

Conjecture 1. *Tout code* C *est générateur minimal de* ${}^{\omega}C^{\omega}$.

Si cette conjecture est vraie, la suivante le sera aussi :

Conjecture 2. *Tout code générateur de* ${}^{\omega}A^{\omega}$ *est un code maximal.*

Preuve. Soit C un code qui engendre ${}^{\omega}A^{\omega}$ et B un code contenant C. On a ${}^{\omega}B^{\omega} = {}^{\omega}A^{\omega} = {}^{\omega}C^{\omega}$. D'après la précédente conjecture, B est un générateur minimal de ${}^{\omega}B^{\omega}$ et donc $C = B$.

Par conséquent, C est un code maximal. □

RÉFÉRENCES

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

[2] D. Beauquier, *Automates sur les mots bi-infinis.* Thesis, University of Paris VII, France (1986).

[3] V. Bruyère, Codes, Chapter 7, *Algebraic Combinatorics on words*, edited by M. Lothaire (to appear).

[4] J. Devolder, Comportement des codes vis-à-vis des mots infinis et bi-infinis. *Théorie des Automates et Applications*, edited by D. Krob. Rouen, France (1991) 75-90.

[5] J. Devolder et I. Litovsky, Finitely generated biω-langages. *Theoret. Comput. Sci.* **85** (1991) 33-52.

[6] J. Devolder et E. Timmerman, Finitary codes for biinfinite words. *RAIRO: Theoret. Informatics Appl.* **26** (1992) 363-386.
[7] J. Devolder, Precircular codes and periodic bi-infinite words. *Inform. and Comput.* **107** (1993) 185-201.
[8] J. Devolder, *Codes, mots infinis et bi-infinis.* Ph.D. Thesis, University of Lille I, France (1993).
[9] J. Devolder, M. Latteux, I. Litovsky et L. Staiger, Codes and infinite words. *Acta Cybernet.* **11** (1994) 241-256.
[10] F. Gire et M. Nivat, Langages algébriques de mots bi-infinis. *Theoret. Comput. Sci.* **86** (1991) 277-323.
[11] J.-L. Lassez, Circular codes and synchronisation. *Internat. J. Comput. Inform. Sci.* **5** (1976) 201-208.
[12] I. Litovsky, Prefix-free languages as ω-generators. *Inform. Process. Lett.* **37** (1991) 61-65.
[13] M. Nivat et D. Perrin, Ensembles reconnaissables de mots bi-infinis, in *Proc. 14e ACM Symp. on Theory of Computing*, Vol. 005 (1982) 47-59.

Communiqué par J.E. Pin.
Reçu le 31 juillet 2000. Accepté le 19 mars 2001.