Ann. Sci. Math. Québec **21** (1997), no. 2, 103–122.

LANGAGES DE DYCK GÉNÉRALISÉS ET FACTORISATIONS DU MONOÏDE LIBRE

HÉLÈNE JACQUET ET GUY MELANÇON

RÉSUMÉ. Nous étudions un langage de mots D_α qui généralise le langage des mots de Dyck classique sur deux lettres. Nous donnons une grammaire non ambiguë qui engendre ce langage et qui nous permet de donner une équation fonctionnelle satisfaite par sa série génératrice. Nous procédons ensuite, dans la seconde partie, à l'étude d'une factorisation du monoïde libre, appelée la factorisation de Spitzer-Foata. Elle fait intervenir la généralisation D_α du langage de Dyck que nous étudions et utilise la construction des mots de Lyndon. Nous donnons un algorithme qui engendre les mots de la factorisation de Spitzer-Foata, inspiré des idées de Duval [5] pour la factorisation de Lyndon. Le calcul des coefficients de la série génératrice du langage D_α nous permet de donner une estimation satisfaisante de la complexité de notre algorithme.

ABSTRACT. We study a set of words D_α, generalizing the classical Dyck language on two letters. We give a non ambiguous grammar generating this language and find a functional equation for its generating series. We then proceed, in the second part, to the study of a factorization of the free monoid, called the Spitzer-Foata factorization. It is constructed from the sets D_α and the set of Lyndon words. We give an algorithm generating the words of the Spitzer-Foata factorization, inspired from Duval's idea [5] for the Lyndon factorization. The computations of the coefficients of the generating series for D_α lead to a satisfactory estimate for the complexity of our algorithm.

1. Introduction. L'étude des factorisations du monoïde libre est motivée par ses liens avec les calculs dans les algèbres de Lie libres et les groupes libres (*cf.* [19, 22]). Les applications des techniques de la combinatoire des mots lui ont aussi donné une place centrale en théorie du contrôle (*cf.* [10, 18]). De ce point de vue, le calcul effectif des factorisations, et des objets algébriques qui lui sont associés, a pris une relative importance en combinatoire des mots (comme le montrent, par exemple, des travaux récents [1, 2]).

L'une des factorisations du monoïde libre peut-être la plus connue – *la factorisation de Lyndon*, a aussi fait l'objet d'une application dans le monde industriel [20]. Cette factorisation se prête facilement aux manipulations algorithmiques à cause de sa présentation combinatoire relativement simple. Les algorithmes de factorisation des

Reçu le 6 avril 1996 et, sous forme définitive, le 1er octobre 1996.

mots et de génération des mots de Lyndon de Duval [4, 5] montrent bien cet avantage de la factorisation de Lyndon sur les autres factorisations du monoïde libre (*cf.* section 3.3.1).

Nous proposons ici d'étudier une autre factorisation du monoïde libre (sur deux lettres a, b), qui comme la factorisation de Lyndon, se construit directement à partir de mots: la factorisation de Spitzer-Foata (proposition 3.3). Nous reprenons les idées sous-jacentes aux algorithmes de Duval pour les mots de Lyndon pour donner des algorithmes adaptés à cette factorisation (section 3, théorème 3.26).

La construction de cette factorisation fait apparaître des langages D_α ($\alpha \in \mathbf{Q}^+$) qui généralisent le langage des mots de Dyck classique (définition 2.1). Cette généralisation diffère de celle étudiée par Labelle [11, 12, 13, 14] (*cf.* remarque 2.11). Nous étudions le langage D_α et obtenons, pour certaine valeurs de α (*cf.* remarque 2.4) une grammaire algébrique qui l'engendre (théorème 2.3, corollaire 2.5). On met ensuite à profit la méthodologie DSV pour obtenir une relation fonctionnelle (l'équation (4)) satisfaite par la série génératrice du langage, ce qui permet en bout de ligne d'évaluer de façon satisfaisante la complexité des algorithmes que nous proposons.

2. Les langages D_α. Nous nous intéressons ici à des langages D_α ($\alpha \in \mathbf{Q}^+$) qui généralisent le langage de Dyck classique. Ces langages jouent un rôle important dans la définition de la factorisation de Spitzer-Foata sur $\{a, b\}$, que nous étudierons à la section 3. Nous donnons une grammaire algébrique non-ambiguë qui engendre ce langage; nous sommes ainsi en mesure de donner une relation fonctionnelle satisfaite par la série génératrice du langage et d'en calculer les coefficients.

Les définitions et notations de base que nous utilisons sont celles usuelles en combinatoire des mots. Le lecteur peut consulter [17]. Dans toute la suite, nous désignerons l'alphabet sous-jacent par $X = \{a, b\}$.

2.1. Mots de D_α et chemins associés.

Définition 2.1. Soit $\alpha \in \mathbf{Q}^+$ un nombre rationnel positif. L'ensemble D_α est le langage des mots $w \in X^*$ qui satisfont:

$$\frac{|w|_b}{|w|_a} = \alpha, \tag{1}$$

$$\forall\, u, v \in X^+, \quad w = uv \quad \Rightarrow \quad \frac{|u|_b}{|u|_a} < \alpha. \tag{2}$$

Ainsi, on a $D_0 = \{a\}$. On admet aussi $\alpha = \infty$; on a $D_\infty = \{b\}$. Il nous sera utile, pour les autres valeurs possibles de α, de les préciser sous la forme d'un quotient de deux entiers $p, q \geq 1$ relativement premiers. Remarquez que les mots de D_α sont nécessairement non vides.

Soient donc des entiers $p, q \geq 1$ relativement premiers. On peut se représenter les mots de $D_{p/q}$ comme des chemins dans le plan discret. Associons dans le plan $\mathbf{N} \times \mathbf{N}$ à la lettre a un pas *est*, et à la lettre b, un pas *nord*. Le chemin associé au mot $w \in D_{p/q}$ est donc un chemin qui va de l'origine $(0, 0)$ au point de coordonnées (kq, kp), pour un

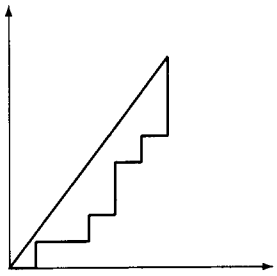

FIGURE 1. Le mot $abaababbababbb \in D_{4/3}$ de pente associée $4/3$.

entier $k \geq 1$. De plus, la condition (2) se lit directement sur le chemin : il ne croise ni ne touche la droite de pente p/q qu'à l'origine et à son extrémité.

Remarque 2.2. Notez que le langage $D_{p/q}$ est un code. C'est-à-dire que le monoïde engendré par $D_{p/q}$ est libre; autrement dit, un mot de $D^*_{p/q}$ se décompose de façon unique en un produit de mots de $D_{p/q}$. On le vérifie facilement à l'aide de la représentation géométrique associée aux mots : une factorisation d'un mot sur $D_{p/q}$ correspond au calcul de ses points de contacts avec la droite de pente p/q. De plus, ce code est *préfixe*, c'est-à-dire que $z, z' \in D_\alpha$, $z \neq z'$, implique que z ne peut être un facteur gauche de z' et vice-versa.

2.2. Fonction génératrice. Nous développons maintenant un argument qui permet de donner le calcul de la série génératrice des langages $D_{p/q}$ et $D^*_{p/q}$, dans le cas où $p = n$ est un entier et $q = 1$ (resp. $p = 1$ et $q = n$ est un entier).

Théorème 2.3. *Soit un mot non vide $w \in X^+$. On a $w \in D^*_n$ (resp. $w \in D^*_{1/n}$) si et seulement si w s'écrit de façon unique comme produit*

$$w = aw_1b \cdots bw_nbw_{n+1} \quad (\textit{resp. } w = aw_1 \cdots aw_nbw_{n+1}) \tag{3}$$

où $w_i \in D^*_n$ $(i = 1, \ldots, n+1)$.

Remarque 2.4. La décomposition prescrite par le théorème 2.3 correspond au découpage habituel des mots de Dyck effectué sur le chemin associé au mot : la recherche du facteur w_i de longueur maximale correspond à balayer le chemin avec une droite parallèle à la droite de pente n, jusqu'à ce que celle-ci coupe la grille en (au moins) deux points compris entre la droite et le chemin. Ces points d'intersection déterminent alors la décomposition (3). C'est ce qu'illustre la figure 2, où $x_1 = a$, $x_2 = x_3 = b$ et $w_1 = abbabb$, $w_2 = ababbb$.

Le cas général ne se soumet pas à un raisonnement aussi simple. En effet, dans le cas général, l'ensemble de points à coordonnées entières compris entre la droite de pente p/q et le chemin associé à un mot est d'une plus grande complexité. En effet, une droite parallèle à la doite de pente p/q, balayée sur la région déterminée par le chemin peut très bien ne rencontrer qu'un seul point de la grille à la fois. L'étude de cet ensemble mène à la formulation de règles de production qui définissent une grammaire pour le langage $D^*_{p/q}$. P. Duchon a récemment établi ce résultat qui fera l'objet d'un travail à venir.

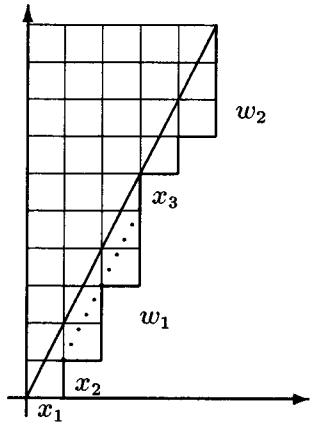

FIGURE 2. Interprétation géométrique de la décomposition des mots de D_n^*.

Démonstration du théorème 2.3

On ne considère que le cas $w \in D_n^*$, le cas $w \in D_{1/n}^*$ étant similaire. Il est clair qu'un mot de la forme (3) est un mot de D_n^*. En effet, soit $u = aw_1b\cdots bu_k$ un facteur gauche du mot $w = aw_1b\cdots w_{n-1}bw_n$, où $k \geq 1$ et où u_k est un facteur gauche de w_k (propre si $k = n$). On a alors

$$\frac{|u|_b}{|u|_a} = \frac{k-1+\sum_{i=1}^{k-1}|w_i|_b + |u_k|_b}{1+\sum_{i=1}^{k-1}|w_i|_a + |u_k|_a} \leq n$$

puisque $k-1 \leq n$, $|w_i|_b = n|w_i|_a$ et $|u_k|_b \leq n|u_k|_a$. Donc $w \in D_n^*$.

Inversement, soit un mot $w \in D_n^*$, de longueur $k(n+1)$. On procède par récurrence sur k pour montrer que w admet une décomposition (3). Dans le cas $k = 1$, on doit avoir $w_i = \epsilon$ (le mot vide) pour tout $i = 1, \ldots, n+1$ puisque tout mot de D_n^* a une longueur qui est un multiple de $n+1$.

Soit donc un mot $w \in D_n^*$ de longueur $k(n+1)$ avec $k \geq 2$. Ce mot commence forcément par la lettre $x_1 = a$. Soit $w_1 \in D_n^*$ de longueur *maximale* tel que x_1w_1 soit un facteur gauche de w. Soit x_2 la lettre qui suit dans le mot w, et $w_2 \in D_{p/q}^*$ de longueur maximale, tel que $x_1w_1x_2w_2$ soit un facteur gauche de w. En continuant ainsi, on obtient une décomposition de $w = x_1w_1\cdots x_mw_m$ avec $w_k \in D_{p/q}^*$. Remarquez qu'au moins l'un des facteurs w_i est non-vide; en effet, sinon on en tirerait que w est de la forme $w = ab^{i_1}ab^{i_2}\cdots ab^{i_k}$ avec $i_j < n$ pour tout $j = 1, \ldots, k$. Et cela entrainerait $w \notin D_n^*$.

On a évidemment

$$\frac{|x_1\cdots x_m|_b}{|x_1\cdots x_m|_a} = n.$$

De plus, pour $k < m$,

$$\frac{|x_1\cdots x_k|_b}{|x_1\cdots x_k|_a} \leq n$$

si et seulement si

$$\frac{|x_1w_1\cdots w_{k-1}x_k|_b}{|x_1w_1\cdots w_{k-1}x_k|_a} \leq n$$

et par conséquent, $x_1 \cdots x_m \in D_n^*$.

On montre maintenant que $x_1 \cdots x_m \in D_n$; cela apparaitra comme conséquence de $|x_1 \cdots x_m| = n+1$. Supposons qu'au contraire, on ait $m > n+1$. Par récurrence, on obtient une décomposition $x_1 \cdots x_m = y_1 t_1 \cdots y_{p+q} t_{p+q}$ avec $t_i \in D_{p/q}^*$. Remarquez qu'on a $x_1 = y_1$ et $x_2 \cdots x_m = t_1 \cdots y_{p+q} t_{p+q}$. On peut, sans perte de généralité, supposer le mot t_1 non vide; soit $t_1 = x_2 \cdots x_k$. On vérifie sans difficulté que $w_1 x_2 w_2 \cdots w_{k-1} x_k \in D_n^*$, ce qui contredit la maximalité de $|w_1|$. On doit donc avoir $m = n+1$ et $x_1 \cdots x_m \in D_n$, de longueur $n+1$.

Supposons qu'un mot $w \in D_n^*$ admette deux décompositions (3), $w = x_1 w_1 \cdots x_m w_m$ et $w = x'_1 w'_1 \cdots x'_m w'_m$. On peut sans perte de généralité supposer que $|w_m| < |w'_m|$. On a donc $x_1 w_1 \cdots w_{m-1} x_m = x'_1 w'_1 \cdots w'_{k-1} x'_k u_k$ où u_k est un facteur gauche non vide de w'_k. Mais alors on obtient une contradiction puisque

$$n = \frac{|x_1 w_1 \cdots w_{m-1} x_m|_b}{|x_1 w_1 \cdots w_{m-1} x_m|_a} = \frac{|x'_1 w'_1 \cdots w'_{m-1} x'_m u_m|_b}{|x'_1 w'_1 \cdots w'_{m-1} x'_m u_m|_a} < n\,.$$

On en tire en corollaire une grammaire algébrique non ambiguë qui engendre le langage D_n^*.

Corollaire 2.5. *Soit S un symbole distinct des lettres de l'alphabet $X = \{a, b\}$. La grammaire algébrique donnée par :*

$$S \to 1 + a \underbrace{Sb \cdots Sb}_{n\,fois} S$$

engendre l'ensemble de mots D_n^*, de façon non ambiguë.

Remarque 2.6. La grammaire correspondant au langage $D_{1/n}^*$ est donnée par :

$$S \to 1 + \underbrace{aS \cdots aS}_{n \text{ fois}} bS.$$

Comme la grammaire est non ambiguë, on en tire une identité dans l'anneau $\mathbf{Q}\langle\langle X \rangle\rangle$ des séries formelles non commutatives. Plus précisément, désignons la série caractéristique du langage D_n^* par $\underline{D_n^*}$, et considérons les lettres $x \in X$ comme des indéterminées non commutatives. On déduit du corollaire 2.5 l'identité :

$$\underline{D_n^*} = 1 + a\underline{D_n^*}b \cdots \underline{D_n^*}b\underline{D_n^*}\,. \tag{4}$$

Ainsi, pour obtenir la fonction génératrice du langage D_n^*, il suffit de calculer l'image commutative de cette identité, via l'homomorphisme $\mathbf{Q}\langle\langle X \rangle\rangle \to \mathbf{Q}[[t]]$ qui à toute lettre $x \in X$ associe l'indéterminée t. Désignons par $D_n^*(t)$ la fonction génératrice du langage D_n^*. Cette série satisfait donc la relation fonctionnelle :

$$D_n^*(t) = 1 + t^{n+1}(D_n^*(t))^{n+1}\,.$$

Une application directe du théorème d'inversion de Lagrange (*cf.* [17, Theorem 11.4.6]) à l'identité (4) satisfaite par $D_n^*(t)$ nous donne :

Corollaire 2.7. *Soit* $D_n^*(t) = \sum_{k\geq 0} a_k t^{k(n+1)}$, *où* a_k *est égal au nombre de mots du langage* D_n^*, *de longueur* $k(n+1)$. *On a :*

$$a_k = \frac{1}{k}\binom{k(n+1)}{k-1}. \tag{5}$$

Une analyse similaire à celle que nous venons de conduire permet de donner la série génératrice du langage D_n. En effet, soit T un nouveau symbole distinct de S. On peut montrer, comme on l'a fait au théorème 2.3 pour le langage D_n^*, en complète analogie avec le cas du langage de Dyck classique, que la grammaire décrite par les deux productions :

$$T \to a\underbrace{Sb\cdots Sb}_{n \text{ fois}}$$

$$S \to 1 + TS$$

engendre bien (à partir de T) le langage D_n de façon non ambiguë. Ainsi, on obtient pour la série génératrice $D_n(t)$ du langage D_n un système d'équations :

$$\begin{aligned} D_n(t) &= t^{n+1}(D_n^*(t))^n, \\ D_n^*(t) &= \frac{1}{1-D_n(t)}. \end{aligned}$$

Corollaire 2.8. *Soit* $D_n(t) = \sum_{k\geq 1} b_k t^{k(n+1)}$, *où* b_k *est égal au nombre de mots du langage* D_n, *de longueur* $k(n+1)$. *On a* $b_1 = 1$ *et, pour* $k \geq 2$ *:*

$$\begin{aligned} b_k &= \frac{kn(kn+1)\cdots(kn+k-2)}{k!} \\ &= \frac{1}{nk+k-1}\binom{nk+k-1}{k}. \end{aligned}$$

Il s'agit à nouveau d'une application directe du théorème d'inversion de Lagrange.

Remarques 2.9. Dans le cas où $p = q = 1$, on retrouve bien la grammaire $D \to 1 + aDbD$ qui engendre les mots de Dyck classiques, et la relation fonctionnelle $D(t) = 1 + t^2 D(t)^2$, qui se résout en $D(t) = (1 - \sqrt{1-4t^2})/2t^2$.

Le cas $p = 1, q = 2$ a déjà été étudié [7] en relation avec l'énumération de polyominos dirigés, diagonalement convexes. Les auteurs concentrent leur étude sur le langage $D_{1/2}$ (plutôt que $D_{1/2}^*$). Ils codent les mots du langage par des arbres ternaires, ce qui est une conséquence du fait qu'il n'existe qu'un seul mot de longueur minimale, soit aab.

Remarques 2.10. Dans [6], les auteurs étudient les mots $w \in \{a, b\}$ qui satisfont $|w|_b = p$, $|w|_a = q$, et qui sont tels que pour tout facteur gauche, le nombre de b est toujours supérieur ou égal au nombre de a. Les auteurs calculent le nombre de tels à l'aide du principe d'André et montre qu'il est $n_{p,q} = \frac{1}{p+q}\binom{p+q}{q}$. Ce langage diffère

du notre puisque par exemple, pour $p = 1$ et $q = 2$ le mot $w = abbbab$ satisfait la condition, mais pourtant $w \notin D_2^*$. De plus, on vérifie facilement que le nombre de mots de D_n (*cf.* (5)) diffère du nombre $n_{p,q}$.

Le langage $D_{p/q}$ apparaît aussi dans les travaux de Borel et Laubie [3], en rapport avec les mots de Christoffel primitifs. Ces mots sont des mots de Lyndon. Pour chaque valeur de $p, q \geq 1$ relativement premiers, on trouve un unique mot de Christoffel; c'est le mot maximal de $D_{p/q}$ pour l'ordre lexicographique (*cf.* corollaire 3.15).

Remarque 2.11. Labelle [11, 12, 13, 14] a étudié une autre généralisation du langage de Dyck classique. On peut présenter le langage de Dyck classique comme l'ensemble des mots associés aux chemins qui vont de l'origine $(0, 0)$ à un point $(2n, 0)$, empruntant des pas nord-est $(i, j) \to (i + 1, j + 1)$ ou sud-est $(i, j) \to (i + 1, j - 1)$. Les pas nord-est font croître l'ordonnée de 1, alors que les pas sud-est font décroître l'ordonnée de 1. Labelle assouplit cette condition; il élargit l'ensemble des pas et permet des pas qui font croître ou décroître l'ordonnée de valeurs entières données (en nombre fini). Il préserve en général la symétrie : les pas viennent en paire; si on peut croître d'une valeur k en ordonnée, on peut aussi décroître de la même valeur. Il permet aussi des pas horizontaux. Ces symétries lui permettent de donner le calcul des coefficients de la série génératrice de ces langages à l'aide de récurrences analogues au cas classique.

L'approche de Labelle ne peut être reprise ici. Les chemins associés aux mots de $D_{p/q}$ et $D_{p/q}^*$ correspondent chez Labelle à des chemins utilisant des pas $(i, j) \to (i, j + p)$ et $(i, j) \to (i, j - q)$.

3. Factorisations de Spitzer-Foata. Nous nous intéressons ici à une factorisation particulière du monoïde libre sur $X = \{a, b\}$, la *factorisation de Spitzer-Foata* (*cf.* [22],[17, Chap. 5.4]). Les langages D_α jouent un rôle important dans la définition de cette factorisation. Nous donnons dans cette section un algorithme qui engendre les mots de cette factorisation, en longueur bornée et nous analysons sa complexité.

3.1. Factorisation de Lyndon. Il nous faut ici rappeler brièvement la factorisation de Lyndon, qui intervient dans la construction de la factorisation de Spitzer-Foata.

Soit donné un ordre total sur un alphabet A (fini ou infini). On étend lexicographiquement cet ordre à l'ensemble de tous les mots. L'ordre lexicographique ainsi obtenu est toujours noté $<$.

Définition 3.1. Un mot $w \in A^+$ est un mot de Lyndon s'il est strictement plus petit que tous ses facteurs droits propres non vides. Nous désignerons par $L(A)$ l'ensemble des mots de Lyndon (sur un alphabet A).

Pour plus de détails sur les mots de Lyndon, le lecteur pourra consulter [17]. Notons que les lettres sont des mots de Lyndon, et que les mots de Lyndon sont primitifs[1]. Nous rappelons une propriété combinatoire des mots de Lyndon qui permet de les engendrer. Soient $u, v \in L(A)$: on a $uv \in L(A) \Longleftrightarrow u < v$.

3.2. Les ensembles $\mathcal{SF}_\alpha$ et $\mathcal{SF}$. Soit un rationnel $\alpha \in \mathbf{Q}^+$. L'ensemble D_α étant un code, on peut le considérer comme un alphabet. On l'ordonne totalement par l'ordre lexicographique induit de X^+, où $X = \{a, b\}$. On peut ainsi étendre cet ordre

[1]Un mot $w \in A^+$ est primitif si $w = z^k$ implique $k = 1$ et $w = z$.

lexicographiquement à tout le monoïde libre D_α^*. Par exemple, on a $abaabbbbb, abb \in D_2$ et $abaabbbbb < abb$ (lexicographiquement). Ainsi, dans D_2^*, on a $abaabbbbbabb < abb$.

On désigne par $\mathcal{SF}_\alpha = L(D_\alpha)$ l'ensemble des mots de Lyndon sur D_α (par rapport à l'ordre lexicographique sur D_α^* induit de l'ordre total sur D_α).

Exemple 3.2. Par exemple, comme $abaabbbbb, abb \in D_2$ et $abaabbbbb < abb$, on a $abaabbbbbabb \in \mathcal{SF}_2$. Remarquez que ni $abaabbbbb$, ni $abaabbbbbabb$ ne sont des mots de Lyndon sur $\{a, b\}$.

L'ensemble

$$\mathcal{SF} = \bigcup_{\alpha \in \mathbf{Q}^+ \cup \infty} \mathcal{SF}_\alpha$$

devient totalement ordonné si on pose, pour $u \in D_\alpha$ et $v \in D_\beta$ $(\alpha \neq \beta)$, $u < v \Leftrightarrow \alpha < \beta$.

Proposition 3.3. *L'ensemble $\mathcal{SF}$ forme une factorisation du monoïde libre X^*, qu'on appelle la factorisation de Spitzer-Foata sur deux lettres. Plus précisément, tout mot $w \in X^*$ s'écrit de façon unique sous la forme :*

$$w = s_1 \cdots s_n \text{ où } s_1, \ldots, s_n \in \mathcal{SF} \text{ et } s_1 \geq \cdots \geq s_n \ (n \geq 0). \tag{6}$$

Notez qu'on a, soit $s_i, s_{i+1} \in \mathcal{SF}_\alpha$ et alors $s_i \geq s_{i+1}$ (par rapport à l'ordre lexicographique induit de l'ordre total sur D_α), soit $s_i \in \mathcal{SF}_{\alpha_i}$ et $s_{i+1} \in \mathcal{SF}_{\alpha_{i+1}}$ avec $\alpha_i > \alpha_{i+1}$.

Remarque 3.4. C'est Spitzer [21] qui le premier a étudié cette factorisation à des fins probabilistes. La description que nous en donnons est plus près des travaux de Foata [8] (et aussi de Viennot [22]).

*Remarque 3.5.*On peut généraliser la factorisation de Spitzer-Foata à un alphabet A de plus de deux lettres. Soit $\mu : A^* \to \mathbf{Q}^+$ un morphisme additif. Soit $\alpha \in \mathbf{Q}^+$. On désigne par Z_α le langage des mots $w \in X^*$ satisfaisant les conditions :

$$\frac{\mu(w)}{|w|} = \alpha, \quad \text{et} \quad \forall u, v \in X^+, \quad w = uv \quad \Longrightarrow \quad \frac{\mu(u)}{|u|} < \alpha.$$

Ensuite, on ordonne totalement ces ensembles (qui sont des codes) par l'ordre lexicographique induit. Puis on forme, pour tout $\alpha \in \mathbf{Q}^+$, l'ensemble $L(Z_\alpha)$ des mots de Lyndon sur Z_α, que l'on note $\mathcal{SF}_\alpha$. La factorisation de Spitzer-Foata (relative à μ) est alors $\mathcal{SF} = \cup_{\alpha \in Q^+} \mathcal{SF}_\alpha$. Il est intéressant de remarquer que sur deux lettres, tout morphisme induit la même factorisation, qu'on peut présenter comme on l'a fait plus haut.

3.3. Génération des mots de $\mathcal{SF}_{p/q}$. Nous souhaitons obtenir un algorithme qui engendre les mots de Spitzer-Foata de longueur au plus m sur l'alphabet de départ $X = \{a, b\}$. Désignons cet ensemble par $\mathcal{SF}^{\leq m}$. Nous pouvons clairement nous

limiter à donner un algorithme qui engendre les mots de $\mathcal{SF}_{p/q}^{\leq m}$, pour p, q donnés. En effet, le plus court mot de ce langage est de longueur $p+q$ et donc

$$\mathcal{SF}^{\leq m} = \bigcup_{p+q \leq m} \mathcal{SF}_{p/q}^{\leq m} \cup \{a, b\},$$

où l'union se fait sur les couples de valeurs entières $p, q \geq 1$ relativement premières.

Duval [5] a donné un algorithme pour engendrer les mots de Lyndon sur un alphabet A fini, en longueur bornée $\leq m$. On peut donc, dans un premier temps, espérer reprendre l'algorithme de Duval et en obtenir l'algorithme souhaité. La difficulté ici vient du fait que nous ne travaillons pas sur un alphabet formé de lettres, dans le sens où une lettre est un mot de longueur unitaire, mais plutôt sur le code préfixe $D_{p/q} \subset X^*$. Une application directe de l'algorithme de Duval engendrerait les mots en longueur sur $D_{p/q}$. Il nous faut contrôler la longueur sur $\{a, b\}$, pour éviter d'engendrer des mots trop longs. Tout le travail de cette section décrit la stratégie que nous envisageons.

3.3.1. Algorithme de génération de Duval. L'algorithme de Duval engendre en ordre lexicographique les mots de Lyndon de longueur bornée $\leq m$, sur un alphabet fini et totalement ordonné. Son principe repose sur une fonction qui en entrée reçoit un mot de Lyndon et retourne son successeur, parmi les mots de Lyndon de longueur au plus m.

L'algorithme de Duval procède en deux étapes. Etant donné un mot de Lyndon de longueur $\leq m$, on calcule d'abord la *sesquipuissance* de w en longueur m, c'est-à-dire le mot u de longueur exactement m facteur gauche d'une puissance de w (appropriée). Puis on parcourt le mot u à partir de la lettre la plus à droite : si une lettre est la lettre maximale elle est effacée et on passe à la suivante; sinon la lettre est remplacée par celle qui suit dans l'alphabet et on s'arrête.

Fixons, par exemple, $m = 8$ et soit acb un mot de Lyndon de longueur 3 sur $\{a < b < c\}$. La première étape nous mène à considérer le mot $acbacbac$; la seconde étape conduit à l'effacement du c le plus à droite, suivi du remplacement du a qui le précède par b, pour donner le mot $acbacbb$. Ce mot est le successeur du mot acb parmi l'ensemble des mots de Lyndon de longueur au plus 8, sur l'alphabet $\{a, b, c\}$. Le lecteur intéressé pourra consulter [4].

Par souci d'économie et de simplicité, l'algorithme de Duval utilise une variable globale `W` qui contient le mot de Lyndon courant, et une variable globale `L` (un entier) telle que ce mot courant soit $W[1] \cdots W[L]$. La valeur entière L permet d'éviter une réinitialisation de la variable W. Ainsi, les entrées $W[L+1]$, $W[L+2]$, ..., contiennent éventuellement une valeur mais sont ignorées. Bien évidemment la valeur de l'entier m est aussi contenue dans une variable globale M. Ainsi, cet algorithme peut s'exprimer simplement comme suit. Le travail des fonctions `sesqui` et `suivant`, qui effectuent les deux étapes successivement, s'expriment par les lignes :

```
sesqui() {
    i ← 1;
    pour i de 1 a M-L faire
        W[L+i] ← W[i];
    L ← M;
}
```

```
suivant() {
    tant que (L > 0 et alors W[L] = lettre_max) faire
        L ← L-1;
    si L <> 0 alors
        W[L] ← lettre_suivante(W[L]);
    sinon rien
}
```

où l'on désigne par `lettre_max` la lettre maximale de l'alphabet et où la fonction `lettre_suivante` effectue ici le travail évident qui consiste à retourner la lettre suivante de l'alphabet. L'algorithme lui-même se ramène donc aux quelques lignes :

```
Lyndon() { /* la variable M a deja ete initialisee */
    L ← 1;
    W[1] ← a; /* la lettre minimale de l'alphabet */
    tant que L > 0 faire
        suivant(sesqui());
}
```

Proposition 3.6. [5, Proposition 2.2]
L'algorithme `Lyndon` *engendre l'ensemble des mots de Lyndon de longueur* $\leq m$ *sur un alphabet fini, en ordre lexicographique. Lors du passage d'un mot de Lyndon à son suivant, à l'aide du calcul* `suivant(sesqui())`*, le nombre de comparaisons de lettres n'excède pas* $m + 1$.

A chacun des passages dans la boucle le mot de Lyndon suivant est engendré. Les instructions permettant le stockage ou l'affichage de ces mots sont ignorées. Nous analysons ici un algorithme qui reprend l'algorithme de Duval sur le code $D_{p/q}$, mais qui n'engendre que les mots de longueur convenable sur $\{a, b\}$. Un travail supplémentaire est nécessaire pour gagner un contrôle de la longueur sur l'alphabet $\{a, b\}$. La clé réside en une représentation bien adaptée des mots.

Soit $a^+b^+ = \{a^ib^j | i, j \geq 1\}$ le langage des mots non vides, non réduit à une lettre et sans facteur ba. Tout mot non vide ne commençant pas par b, ni ne finissant par a se code sur a^+b^+. En effet, un tel mot est de la forme $a^{t_1}b^{r_1} \cdots a^{t_n}b^{r_n}$, avec $n \geq 1$ et $t_i, r_i \geq 1$. Les mots de $D^*_{p/q}$ commencent tous par a et se terminent par b. Ainsi, on peut les coder sur a^+b^+.

3.4. Successeur d'un mot de $D_{p/q}$. Dans l'algorithme de Duval, la fonction `lettre_suivante` effectue un travail évident : on peut supposer que l'alphabet sous-jacent est stocké sous la forme d'une liste finie de lettres. Elle retourne alors le successeur d'une lettre dans cette liste. On ne peut supposer dans notre cas, que le code $D^{\leq m}_{p/q}$ soit connu explicitement. Il nous faut au contraire donner une méthode qui permet de calculer le successeur d'un mot $w \in D_{p/q}$ parmi les mots de $D^{\leq m}_{p/q}$.

Cette section contient une série de lemmes qui établissent comment ce calcul peut être accompli. Plus précisément, pour $m \geq 1$ fixé et $z \in D_{p/q}$, le problème est de décrire le successeur z' de z de longueur au plus m. Deux cas sont à considérer soit $|z| \leq m$, soit $|z| > m$.

Nous utiliserons ici le codage d'un mot sur le code a^+b^+. On écrira un mot $z \in D_{p/q}$ sous la forme $z = a^{t_1}b^{r_1} \cdots a^{t_n}b^{r_n}$.

3.4.1. Successeur de longueur supérieure ou égale. Soient p, q des entiers non nuls, relativement premiers. On détermine facilement le mot le plus petit (lexicographiquement) dans $D^{\leq m}_{p/q}$.

Lemme 3.7. *Soit $m = k(p+q)$ un entier, multiple de $p+q$. Le mot de $D^{\leq m}_{p/q}$ le plus petit est $z = a^{kq}b^{kp}$.*

Soit un mot $z = a^{t_1}b^{r_1} \cdots a^{t_n}b^{r_n} \in D^{\leq m}_{p/q}$. Pour tout $j \in \{1, \ldots, n\}$, on considère les quantités :

$$\nu_z(j) = (\sum_{i=1}^{j-1} r_i) + 1, \quad \delta_z(j) = (\sum_{i=1}^{j} t_i) - 1. \tag{7}$$

Remarquez que la donnée de ces quantités est équivalente à la donnée des exposants t_i, r_i.

Lemme 3.8. *Soit $z = a^{t_1}b^{r_1} \cdots a^{t_n}b^{r_n} \in D^{\leq m}_{p/q}$. Soit $j \geq 1$ maximal tel que $\nu_z(j)/\delta_z(j) < p/q$ et pour tout $i > j$, $\nu_z(i)/\delta_z(i) > p/q$. On considère le mot :*

$$z' = a^{t'_1}b^{r'_1} \cdots a^{t'_j}baa^{t'}b^{r'},$$

où l'on a posé :

$$\begin{array}{l} t'_i = t_i, \quad r'_i = r_i, \quad \textit{pour } i \leq j-1, \\ t'_j = t_j - 1 \geq 0 \end{array}$$

$$\textit{et} \begin{cases} t' = kq - \sum_{i=1}^{j} t_i, \\ r' = kp - [\sum_{i=1}^{j-1} r'_i + 1] = kp - [\sum_{i=1}^{j-1} r_i + 1] \end{cases}$$

Alors z' est le successeur de z, parmi les mots de $D^{\leq m}_{p/q}$.

Remarque 3.9. Le lemme précédent et le suivant, s'interprètent aisément à l'aide de la représentation géométrique associée aux mots. En effet, le mot z' du lemme 3.8 s'obtient en remplaçant dans le mot z une lettre a par une lettre b, le plus à droite possible, en le faisant suivre d'un facteur a^ib^j approprié. Les inégalités à vérifier rendent compte du fait que le mot résultant doit rester sous la droite de pente p/q.

La formulation des lemmes à l'aide des quantités $\nu_z(i)$ et $\delta_z(i)$ est ici nécessaire et sera utilisée dans la formulation des algorithmes qui effectuent le calcul du successeur d'un mot dans $D^{\leq m}_{p/q}$.

Exemples 3.10. Posons $p = 3, q = 2$ et $m = 10$.

Soit $z = a^3b^4ab^2 \in D_{3/2}$. On a $\nu_z(2)/\delta_z(2) = \frac{5}{3} > \frac{3}{2}$, et $\nu_z(1)/\delta_z(1) = \frac{1}{2} < \frac{3}{2}$. Le successeur de z, de taille au plus 10 est donc $z' = a^2ba^2b^5$.

Soit $z = a^2b^3 \in D_{3/2}$. On a $\nu_z(1)/\delta_z(1) = 1 < \frac{3}{2}$, donc dans ce cas le successeur de z, de taille au plus 10, est $z' = aba^3b^5$.

Remarque 3.11. Remarquez que l'écriture de z' à l'aide des exposants $t'_i, r'_i, \ldots$et t', r' ne correspond pas à une décomposition sur a^+b^+ si $t'_j = 0$. Néanmoins, les valeurs $\nu_{z'}(i)$ et $\delta_{z'}(i)$ se calculent directement à partir de leurs homologues pour z. En effet, on a :

$$\begin{array}{l} \nu_{z'}(i) = \nu_z(i), \\ \delta_{z'}(i) = \delta_z(i), \end{array} \quad \text{pour } i < j,$$

$$\nu_{z'}(j) = \nu_z(j), \quad \nu_{z'}(j+1) = \nu_z(j) + 1,$$

$$\text{et} \begin{cases} \delta_{z'}(j) = \delta_z(j) - 1, \\ \delta_{z'}(j+1) = kq - 1. \end{cases} \quad \text{si } t'_j \neq 0,$$

$$\delta_{z'}(j) = kq - 1, \qquad \text{si } t'_j = 0.$$

Démonstration du lemme 3.7.

On vérifie facilement que $z' \in D_{p/q}$ et que $z' > z$. S'il existe un mot $w \in D^{\leq m}_{p/q}$ tel que $z' > w > z$ alors il est de la forme $w = a^{t'_1}b^{r'_1} \cdots a^{t'_j}cw'$, où $c \in \{a, b\}$.

Regardons d'abord le cas $c = a$. On a, en vertu des définitions des exposants t'_i, r'_i : $w = a^{t_1}b^{r_1} \cdots a^{t_j}w'$. Le mot z n'est pas facteur gauche de w et par conséquent, seuls deux cas se présentent.

Ou bien $w' = b^{r_j}a^{t_{j+1}} \cdots b^{r_{j+l}}a^t bw''$ avec $t \leq t_{j+l+1} - 1$ pour un $l \geq 0$. De plus, $j + l = n$ entrainerait $w = zw'''$. On peut donc calculer $\nu_z(j+l+1)/\delta_z(j+l+1)$; la maximalité de j entraine $\nu_z(j+l+1)/\delta_z(j+l+1) > \frac{p}{q}$, c'est à dire :

$$\nu_z(j+l+1)/\delta_z(j+l+1) = \frac{(\sum_{i=1}^{j+l} r_i) + 1}{(\sum_{i=1}^{j+l} t_i) + t_{j+l+1} - 1} > \frac{p}{q}.$$

Or $t \leq t_{j+l+1} - 1$ implique

$$\frac{(\sum_{i=1}^{j+l} r_i) + 1}{(\sum_{i=1}^{j+l} t_i) + t} \geq \nu_z(j+l+1)/\delta_z(j+l+1) > \frac{p}{q}.$$

Par conséquent w admet un facteur gauche propre non vide $w_1 = a^{t_1}b^{r_1} \cdots a^{t_j}b^{r_j} \cdots b^{r_{j+l}}a^t b$, tel que $\frac{|w_1|_b}{|w_1|_a} > \frac{p}{q}$. Donc $w \notin D_{p/q}$ et on a une contradiction.

Ou bien $w' = b^{r_j}a^{t_{j+1}}b^{r_{j+1}} \cdots a^{t_{j+l}}b^r w''$ avec $r > r_{j+l}$, pour un $l \geq 0$. Cette fois encore, $j+l = n$ entrainerait $w = zw'''$. On peut donc calculer $\nu_z(j+l+1)/\delta_z(j+l+1)$:

$$\begin{aligned} \frac{\nu_z(j+l+1)}{\delta_z(j+l+1)} &= \frac{(\sum_{i=1}^{j+l} r_i) + 1}{(\sum_{i=1}^{j+l+1} t_i) - 1} \\ &\leq \frac{(\sum_{i=1}^{j+l} r_i) + (r - r_{j+l})}{(\sum_{i=1}^{j+l} t_i) + t_{j+l+1} - 1} \quad \text{car } r - r_{j+l} \geq 1 \\ &\leq \frac{(\sum_{i=1}^{j+l} r_i) + (r - r_{j+l})}{\sum_{i=1}^{j+l} t_i} \quad \text{car } t_{j+l+1} - 1 \geq 0 \end{aligned}$$

Or, $w \in D_{p/q}$ implique alors

$$\nu_z(j+l+1)/\delta_z(j+l+1) \leq \frac{(\sum_{i=1}^{j+l-1} r_i) + r}{\sum_{i=1}^{j+l} t_i} < p/q,$$

ce qui contredit l'hypothèse selon laquelle $\nu_z(j+l+1)/\delta_z(j+l+1) > p/q$.

Dans le cas où $c = b$, on a $w = a^{t_1}b^{r_1}\cdots a^{t_j-1}bw'$. Encore une fois, $w < z'$ et w non facteur gauche de z' entrainent $w' = a^t w''$ avec $t \geq t'+1$. On a donc $|w|_a \geq \sum_{i=1}^{j} t_i - 1 + t > \sum_{i=1}^{j} t_i + t'$. Comme $t' = kq - \sum_{i=1}^{j} t_i$, on trouve $|w|_a > kq$ et on obtient une contradiction.

Le lemme suivant traite d'un autre cas de figure et se montre de façon analogue au lemme 3.8.

Lemme 3.12. *Soit $z = a^{t_1}b^{r_1}\cdots a^{t_n}b^{r_n} \in D_{p/q}^{\leq m}$ Soit j l'indice maximal pour lequel on ait $\nu_z(j)/\delta_z(j) = p/q$ et pour tout $i > j$, $\nu_z(i)/\delta_z(i) > p/q$. On considère le mot :*

$$z' = a^{t'_1}b^{r'_1}\cdots a^{t'_j}b,$$

où l'on a posé :

$$\begin{array}{l} t'_i = t_i, \quad r'_i = r_i, \quad \textit{pour } i \leq j-1, \\ t'_j = t_j - 1. \end{array}$$

Alors z' est le successeur de z, parmi les mots de $D_{p/q}^{\leq m}$.

Exemple 3.13. Posons $p = 3, q = 2$ et $m = 10$.

Soit $z = a^2b^2ab^2ab^2 \in D_{3/2}$. On a $\nu_z(3)/\delta_z(3) = \frac{5}{3} > \frac{3}{2}$, et $\nu_z(2)/\delta_z(2) = \frac{3}{2}$. Le successeur de z de longueur au plus 10 est donc $z' = a^2b^3$.

Remarque 3.14. Encore une fois, les valeurs $\nu_{z'}(i)$ et $\delta_{z'}(i)$ se calculent directement à partir de leurs homologues pour z. En effet, on a :

$$\begin{array}{l} \nu_{z'}(i) = \nu_z(i), \\ \delta_{z'}(i) = \delta_z(i), \end{array} \quad \text{pour } i < j, \quad \text{et} \quad \begin{array}{l} \nu_{z'}(j) = \nu_z(j), \\ \delta_{z'}(j) = \delta_z(j) - 1. \end{array}$$

Le calcul du mot suivant, dans l'algorithme de Duval, se ramène à détecter la lettre la plus à droite qui n'est pas la lettre maximale de l'alphabet. Il faut donc, dans notre cas, être en mesure de déterminer le plus grand (lexicographiquement), parmi les mots de $D_{p/q}^{\leq m}$. En réalité, nous sommes en mesure de décrire le mot *le plus grand de* $D_{p/q}$.

Lemme 3.15. *Soit $z = a^{t_1}b^{r_1}\cdots a^{t_n}b^{r_n} \in D_{p/q}$. Supposons que pour $j = 1, \ldots, n$, on ait $\nu_z(j)/\delta_z(j) > p/q$, alors z est le mot le plus grand de $D_{p/q}$.*

Remarque 3.16. Géométriquement, le mot maximal est celui qui est *le plus près possible* de la droite de pente p/q. Nous allons voir plus loin, que ce mot est de longueur exactement $p+q$ (corollaire 3.23).

Borel et Laubie [3] ont étudié ces mots maximaux, mieux connus sous le nom de mots de Christoffel primitifs. Ce sont des mots de Lyndon (*cf.* [3, théorème 1]), et leurs propriétés combinatoires les lient fortement au développement en fraction continue des

nombres rationnels. Laubie et Laurier [15, 16] ont donné des algorithmes de calcul sur ces fractions continues qui utilisent la combinatoire de la factorisation de Lyndon.

Démonstration du lemme 3.15

Supposons qu'il existe $z' \in D_{p/q}$ tel que $z' > z$. Deux cas se présentent.

Soit $z' = a^{t'_1} b^{r'_1} \cdots a^{t'_j} b z''$ avec $r'_i = r_i$, $t'_i = t_i$ pour $i < j$, $t'_j < t_j$. On calcule :

$$\begin{aligned}
\frac{p}{q} &\geq \frac{(\sum_{i=1}^{j-1} r'_i) + 1}{(\sum_{i=1}^{j-1} t'_i) + t'_j} \\
&\geq \frac{(\sum_{i=1}^{j-1} r'_i) + 1}{(\sum_{i=1}^{j-1} t'_i) + t_j - 1} \quad \text{car } t'_j \leq t_j - 1 \\
&\geq \frac{(\sum_{i=1}^{j-1} r_i) + 1}{(\sum_{i=1}^{j-1} t_i) + t_j - 1} \quad \text{car } t'_i = t_i, r'_i = r_i \text{ pour } i < j \\
&\geq \frac{(\sum_{i=1}^{j-1} r_i) + 1}{(\sum_{i=1}^{j} t_i) - 1} \\
&= \nu_z(j)/\delta_z(j).
\end{aligned}$$

Ce qui contredit l'hypothèse selon laquelle $\nu_z(j)/\delta_z(j) > p/q$.

Soit $z' = a^{t'_1} b^{r'_1} \cdots a^{t'_j} b^{r'_j} z''$ avec $r'_i = r_i$, $t'_i = t_i$ pour $i < j$, $t'_j = t_j$, $r'_j > r_j$, et $j \neq n$. On calcule :

$$\begin{aligned}
\frac{p}{q} &\geq \frac{\sum_{i=1}^{j} r'_i}{\sum_{i=1}^{j} t'_i} \\
&\geq \frac{(\sum_{i=1}^{j-1} r'_i) + r'_j}{\sum_{i=1}^{j} t'_i + t'_{j+1)-1}} \quad \text{car } j \neq n \text{ et } t_{j+1} \geq 1 \\
&\geq \frac{(\sum_{i=1}^{j-1} r_i) + r'_j}{\sum_{i=1}^{j+1} t_i - 1} \quad \begin{array}{l}\text{car } r'_i = r_i, t'_i = t_i \text{ pour } i < j \\ \text{et } t'_j = t_j\end{array} \\
&\geq \frac{(\sum_{i=1}^{j} r_i) + 1}{\sum_{i=1}^{j+1} t_i - 1} \quad \text{car } r'_j \geq r_j + 1 \\
&\geq \nu_z(j+1)/\delta_z(j+1),
\end{aligned}$$

ce qui encore une fois contredit l'hypothèse $\nu_z(j+1)/\delta_z(j+1) > p/q$. Par conséquent, z est le mot le plus grand de $D_{p/q}$.

3.4.2. Successeur de longueur inférieure. Soit $z \in D_{p/q}$ de longueur $|z| > m$. On s'intéresse au calcul du successeur de z parmi les mots de $D_{p/q}^{\leq m}$.

Définition 3.17. Soit $z \in D_{p/q}^{\leq m}$. On définit $\text{tronc}_m(z)$ comme étant le mot de $D_{p/q}$, de longueur exactement m, qui possède un facteur gauche commun avec z, de longueur maximale.

Exemple 3.18. Soit $z = a^2b^2abab^3$, z appartient à $D_{3/2}$ et z est de longueur 10. On a $\text{tronc}_5(z) = a^2b^3$.

Remarque 3.19. Le mot $\text{tronc}_m(z)$ n'est défini que pour $m = k(p+q)$ un multiple de $p+q$. Ce mot est défini puisque $a^{kq}b^{kp}$ est un mot de $D_{p/q}$ de longueur m qui possède un facteur gauche commun avec z de longueur au moins 1.

Lemme 3.20. *Soit $z \in D_{p/q}$, tel que $|z| = k(p+q) \geq k'(p+q) = m$. Soient $u = a^{k'q}b^{k'p}$ et v et s de longueur maximale tels que :*

$$z = v\bar{z}, u = s\bar{u},$$
$$|v| = |s|,$$
$$\textit{et} \quad |v|_a = |s|_a, |v|_b = |s|_b.$$

Alors $z' = vb^{k'p-|v|_b}$ est le successeur de z, parmi les mots de $D^{\leq m}_{p/q}$ si $|z| > m$ et $z' = z$ sinon. De plus, on a $z' = \text{tronc}_m(z)$.

Exemple 3.21. L'interprétation géométrique est encore une fois éclairante. Le calcul des facteurs gauches v et s correspond au calcul du point d'intersection le plus au nord-est entre les chemins associés aux mots z et $u = a^{k'q}b^{k'p}$. Reste alors, pour obtenir le chemin associé au mot z' cherché, à faire suivre ce point d'intersection d'un segment vertical jusqu'à la droite de pente p/q.

Soit $z = a^3b^2a^2ba^3b^5$ et calculons le successeur z' de z de longueur $m = 12$. On a $z \in D_1$ et $u = a^6b^6$. Les facteurs gauches v et s qui satisfont les conditions sont $v = a^3b^2a^2ba$ et $s = a^6b^3$. Le successeur de z, parmi les mots de $D^{\leq 12}_{p/q}$ est donc $a^3b^2a^2bab^3$.

Remarque 3.22. Notez d'abord que s commence nécessairement par $a^{k'q}$, et que par conséquent le mot z' est exactement de taille m puisqu'il comptera exactement $k'q$ fois la lettre a. De plus, le calcul de v revient à trouver un facteur gauche (le plus long possible) de z qui compte exactement $k'q$ fois la lettre a.

Démonstration du lemme 3.20

Il est clair que z' défini ci-dessus est égal à z si $|z| = m$, et strictement supérieur à z sinon. Supposons qu'il existe $w \in D^{\leq m}_{p/q}$ tel que $z < w < z'$ et $|w| = k''(p+q)$. Les inégalités $z < w < z'$ impliquent $v\bar{z} < w < vb^{k'p-|v|_b}$. Par conséquent, $w = vw_1$ et $\bar{z} < w_1 < b^{k'p-|v|_b}$. Comme w n'est pas facteur gauche de z' deux cas sont possibles : soit $w_1 = aw_2$, soit $w_1 = b^r aw_2$ avec $r < k'p - |v|_b$.

Dans chacun des cas, $|w| = k''(p+q)$ implique que $|w|_a = |v|_a + 1 + |w_2|_a = k''q$ et comme $|v|_a \geq k'q$, alors $k''q \geq k'q + 1 + |w_2|_a$. Ainsi, $k'' \geq k' + \frac{1+|w_2|_a}{q}$ entraine $|w| > m$.

Reste à montrer que $z' = \text{tronc}_m(z)$. Soit $w = \text{tronc}_m(z)$. Par définition, w est le mot de $D_{p/q}$ de longueur exactement m, qui possède un facteur gauche commun avec z de longueur maximale. Or, comme z' possède un facteur gauche commun avec z, on a nécessairement $z < w \leq z'$. Or, le raisonnement précédent montre que $w < z'$ entraine $|w| > m$, ce qui est impossible. Donc $\text{tronc}_m(z) = w = z'$.

Corollaire 3.23. *Soit z le mot le plus grand de $D_{p/q}$, alors $|z| = p + q$.*

Soit $z' = \text{tronc}_{p+q}(z)$, alors $z' \geq z$, en vertu du lemme 3.20. En outre, z étant l'élément maximal de $D_{p/q}$, $z \geq z'$. Par conséquent $z = z' = \text{tronc}_{p+q}(z)$, d'où $|z| = p + q$.

3.5. Algorithme de génération. Nous sommes en mesure de formuler maintenant un algorithme qui engendre les mots de $\mathcal{SF}_{p/q}$. Nous proposons de reprendre l'algorithme de Duval et d'adapter à notre situation les fonctions `sesqui` et `suivant`, qui cette fois feront appel à une fonction `lettre_suivante` plus élaborée, que nous préciserons plus loin.

Nous convenons de représenter un mot de $D^*_{p/q}$ par la suite des mots de $D_{p/q}$ qui le composent. Ces derniers seront codés sur a^+b^+. Ainsi le mot $z = a^{t_1}b^{r_1} \cdots a^{t_n}b^{r_n}$ correspondra à la liste $(t_1, r_1), \ldots, (t_n, r_n)$. L'ordre sur les mots peut alors être ramené à un ordre lexicographique induit de l'ordre total sur les couples :

$$(t, r) < (t', r') \Longleftrightarrow \begin{cases} \text{soit } t > t', \\ \text{soit } t = t' \text{ et } r < r'. \end{cases}$$

Un mot $z_1 z_2 \cdots \in D^*_{p/q}$ correspondra, lui, à une liste

$$(t_1^{(1)}, r_1^{(1)}), \ldots, (t_n^{(1)}, r_n^{(1)}) \; ; \; (t_1^{(2)}, r_1^{(2)}), \ldots, (t_n^{(2)}, r_n^{(2)}) \; ; \; \ldots$$

La différence des longueurs sur $D_{p/q}$ et sur $X = \{a, b\}$ nous oblige à garder un suivi de la longueur sur X de concaténations de mots de $D_{p/q}$. La variable `L` désignera la longueur du mot `W` sur $D_{p/q}$. Il faut remarquer ici que le calcul de la sesquipuissance d'un mot de $D^*_{p/q}$ sur le code $D_{p/q}$, ne peut être fait en longueur *exactement* égale à m (sur $\{a, b\}$). Ainsi, nous introduisons une variable globale `a_combler` qui, à l'entrée de la fonction `sesqui`, indique la différence de longueurs $|\texttt{W[1]} \cdots \texttt{W[L]}| - m$, et qui est mise à jour à chaque concaténation sur $D_{p/q}$.

Le calcul de la sesquipuissance de `W` ne se fait pas en longueur exactement m (sur $\{a, b\}$), mais en approximation par excès : il nous faut concaténer des mots de $D_{p/q}$ jusqu'à ce que le mot obtenu soit de longueur *au moins* m.

```
sesqui() {
    i ← 0;
    tant que a_combler > 0 faire
        i ← i+1;
        W[L+i] ← W[i];
        a_combler ← a_combler - |W[i]|;
    L ← L + i;
}
```

Ainsi, à la sortie de la fonction `sesqui`, la variable `a_combler` est ≤ 0; les variables `W` et `L` sont telles que le mot `W[1]` $\cdots$ `W[L-1]` est de longueur inférieure à m, alors que le mot `W[1]` $\cdots$ `W[L]` est de longueur supérieure ou égale à m. La longueur de `W` est exactement m si et seulement si la valeur de `a_combler` est égale à 0.

Exemple 3.24. Soit $m = 14$ et $w = (aabb)(ab) \in \mathcal{SF}_1^*$. Avant le calcul de la sesquipuissance de w, on aura $L = 2$ et $a\_combler = 8$ puisqu'on souhaite caculer la sesquipuissance en longueur 14 et que $|w| = 6$. Le calcul se fera en plusieurs étapes :

(1) Le mot w passera d'abord de $(aabb)(ab)$ à $(aabb)(ab)(aabb)$, la variable `a_combler` passera, elle, de 8 à $8-4=4$.
(2) Puis on passera au mot $(aabb)(ab)(aabb)(ab)$, la variable `a_combler` passera de 4 à $4-2=2$.
(3) Finalement, on passera au mot $(aabb)(ab)(aabb)(ab)(aabb)$, et alors la variable `a_combler` passera de 4 à $2-4=-2$.

Le calcul de la sesquipuisance sera accompli (par excès), et on incrémentera la variable $L = L + i = 5$ ($i = 3$).

La quantité `a_combler` nous permettra aussi de déterminer comment doit être fait le calcul de la `lettre_suivante`, comme nous le verrons plus loin. Rappelons-nous qu'il nous faut aussi une constante `lettre_max` qui corresponde au mot maximal de $D_{p/q}$, décrit au lemme 3.15. Ainsi, la fonction `suivant` ne nécessite que quelques instructions supplémentaires pour gérer les valeurs des variables globales.

```
suivant() {
    tant que (L > 0 et alors W[L] = lettre_max) faire
        L ← L-1;
        a_combler ← a_combler + |W[L]|;
    si L <> 0 alors
        W[L] ← lettre_suivante(W[L], a_combler + |W[L]|);
    sinon rien
}
```

Exemple 3.25. Reprenons la sesquipuissance $(aabb)(ab)(aabb)(ab)(aabb)$ (en longueur 14) du mot $w = (aabb)(ab) \in \mathcal{SF}_1^*$ (*cf.* exemple 3.24). Les valeurs des variables `a_combler` et `L` sont égales à -2 et 5, respectivement. Dans D_1, le mot maximal est ab. Ainsi, le mot $W[L] = aabb$ n'est pas le mot maximal de D_1. Il faut donc le remplacer par son sucesseur approprié. La sesquipuissance excède la longueur souhaitée de 2, et la longueur du mot $aabb$ à remplacer est 4; il faut donc remplacer ce mot par son successeur dans $D_1^{\leq \texttt{a\_combler}+|\texttt{W[5]}|} = D_1^{\leq 2}$. Or, l'unique mot de $D_1^{\leq 2}$ est ab. Ainsi, le mot qui suit $w = (aabb)(ab)$ dans $D_1^{\leq 14}$ est $(aabb)(ab)(aabb)(ab)(ab)$.

Reste maintenant à préciser le travail de la fonction `lettre_suivante`. En vertu des résultats de la section 3, il faut distinguer deux cas. On doit calculer le successeur d'une lettre $z \in D_{p/q}$ parmi les mots de $D_{p/q}^{\leq m}$. Dans le cas où $z \in D_{p/q}^{\leq m}$ est lui-même un mot de cet ensemble, le successeur se calcule à l'aide des lemmes 3.8, 3.12 et 3.15. Si au contraire, on a $|z| > m$ alors il faut plutôt faire le calcul du mot $\text{tronc}_m(z)$, comme indiqué au lemme 3.20.

```
lettre_suivante(Z, M) {
    si a_combler < 0 alors tronc_M(Z)
        /* dans ce cas, on sait que |Z| > M */
    sinon successeur(Z, M)
        /* dans ce cas, le calcul se fait comme indique */
        /* aux lemmes de la section 3, a l'aide des */
        /* des quantites ν_Z(j) et δ_Z(j) */
}
```

L'algorithme qui engendre les mots de $SF_{p/q}^{\leq m}$ $(m = k(p+q))$ peut donc s'exprimer comme suit :

```
SF(p, q, k) {
    L ← 1;
    W[1] ← (kq, kp); /* le mot minimal de D_{p/q} */
    a_combler ← 0;
    tant que L > 0 faire
        suivant(sesqui());
}
```

Théorème 3.26. *Pour p, q et k donnés, l'algorithme SF calcule la liste des mots du langage $SF_{p/q}^{\leq m}$ $(m = k(p+q))$, par ordre croissant. De plus, le passage d'un mot de $SF_{p/q}^{\leq m}$ au suivant se fait en moyenne à l'aide d'au plus $m(1 + 1/\beta_k)$ comparaisons de lettres de $X = \{a, b\}$, où β_k est égal à la longueur moyenne sur $D_{p/q}$ d'un mot de $D_{p/q}^*$ de longueur au plus m sur X. D'autre part, au cours de l'algorithme le calcul du successeur d'un mot de $D_{p/q}$ se fait en moyenne en au plus $m/4\beta_k$ comparaisons.*

Que l'algorithme engendre bien les mots dans l'ordre croissant résulte directement du fait qu'il engendre une sous-suite des mots de Lyndon sur $D_{p/q}^{\leq m}$ en suivant l'algorithme de Duval. Qu'il engendre bien *tous* les mots de $SF_{p/q}^{\leq m}$ *et seulement ceux-là* devrait apparaître assez clair au lecteur après notre discussion. Une vérification détaillée demande tout de même un certain travail et nous préférons renvoyer le lecteur intéressé à [9].

On désigne par β_k la longueur moyenne sur $D_{p/q}$ d'un mot de $D_{p/q}^*$ de longueur au plus m sur X. C'est-à-dire que β_k est égal au nombre moyen de mots de $D_{p/q}$ dans un mot de $D_{p/q}^*$ de longueur au plus m. Par conséquent, la longueur moyenne (sur X) d'un mot de $D_{p/q}$ est au plus m/β_k. Le passage d'un mot de $SF_{p/q}^{\leq m}$ au suivant se fait à l'aide de l'algorithme de Duval, auquel s'ajoute quelques manipulations propres à notre situation. Ainsi, en vertu de la proposition 3.6, le nombre de comparaisons de mots de $D_{p/q}$ n'excède pas en moyenne $\beta_k + 1$. Comme la comparaison de mots de $D_{p/q}$ se fait lexicographiquement, le nombre total de comparaisons de lettres de X n'excède pas en moyenne $(\beta_k + 1)m/\beta_k$.

Le calcul de la fonction `lettre_suivante` s'effectue à l'aide de comparaisons des quantités (t_i, r_i) ou encore des quantités $\nu_Z(j)$ et $\delta_Z(j)$. Ce nombre de comparaisons n'excède donc pas en moyenne le nombre de facteurs $a^i b^j$ dans un mot de $D_{p/q}$. Or, on montre facilement que le nombre de facteurs $a^t b^r$ d'un mot de longueur n est $n/4$, à partir de la série génératrice $x^2/(1-x)^2$ du langage a^+b^+. Notre estimation suit donc du fait que la longueur moyenne sur X d'un mot $D_{p/q}$ est β_k.

Remarque 3.27. Un calcul plus fin de ces estimations demanderait de donner le calcul des coefficients (ou un équivalent asymptotique) de la dérivée logarithmique de la série $D_{p/q}(t)$ (*cf.* corollaire 2.8). Une alternative serait de donner un calcul satisfaisant des coefficients de la série

$$1 + \sum_{n \geq 1} D_{p/q}(t)^n z^n.$$

Remerciement. Ce travail a bénéficié de l'aide financière de la communauté européenne, sous le contrat CHRX-CT93-0400.

English extended abstract. The paper is twofold. First we describe a family of languages D_α ($\alpha \in \mathbf{Q}^+ \cup \infty$) generalizing the classical Dyck languages on two letters (*cf.* Definition 2.1). This generalization is different from the one proposed by Labelle [14]. We give a non ambiguous grammar generating this language. We write $\alpha = p/q$ with $p, q \geq 1$ relatively prime integers.

Theorem 2.3. *Let $w \in X^+$ be a non empty word and $n \geq 1$. We have $w \in D_n^*$ (resp. $w \in D_{1/n}^*$) if and only if w may be written as a unique product $w = aw_1b \cdots bw_nbw_{n+1}$ (resp. $w = aw_1 \cdots aw_nbw_{n+1}$), where $w_i \in D_n^*$ $(i = 1, \ldots, n+1)$.*

Corollary 2.5. *Let S be distinct from the letters of $X = \{a, b\}$. The algebraic grammar given by the rules:*

$$S \to 1 + a\underbrace{Sb \cdots Sb}_{n \text{ times}}S$$

is non ambiguous and generates the set D_n^*.

This enables us to give a functional equation satisfied by the generating function $D_n^*(t)$. This functional equation may then be used to get expressions for the coefficients, with the help of Lagrange's inversion formula (Eq. 5).

The second part is devoted to the study of a particular factorization of the free monoid on a two letters alphabet $\{a, b\}$: the *Spitzer-Foata factorization* (Definition 3.3). We give results in the same spirit than those by Duval [4, 5] for the Lyndon factorization. Recall that the set of Lyndon words over a totally ordered alphabet A is the set of words that are strictly less than their non empty proper right factors, with respect to the lexicographical order on A^*. The Spitzer-Foata factorization is constructed using both the Lyndon factorization and the languages D_α. More precisely, we totally order the set D_α with the induced lexicographical order and we then form the set of Lyndon words over D_α, which is denoted by $\mathcal{SF}_\alpha$. The Spitzer-Foata factorization over $\{a, b\}$ is the set $\mathcal{SF} = \bigcup_{\alpha \in \mathbf{Q}^+ \cup \infty} \mathcal{SF}_\alpha$.

Following the ideas of Duval, we give an algorithm generating the words of Spitzer-Foata of bounded length (Section 3). Duval's idea is quite simple to state: in order to compute the successor (of length $\leq m$) of a given Lyndon word u, first compute the greatest power of u of length immediately exceeding m and then cut it to a word of length m; secondly, read the word from the right hence visiting each letter, and either erase it if it is the maximal letter of the alphabet, or replace it by the letter immediately following it in A. What you get is a Lyndon word, immediately following u in the set of Lyndon words of length $\leq m$.

The difficulty here arises from the fact that the underlying alphabet is the set of words in D_α of length at most m, denoted $D_\alpha^{\leq m}$. We need to develop a procedure to compute the successor of a word in D_α^m (Subsection 3). Furthermore, we need to be able to describe explicitly the maximal word in D_α^m; this word turns out to be the maximal word in D_α. The result on the generating series for the languages D_α lead to a satisfactory measure of the complexity of our algorithm (Theorem 3.26), at least as satisfactory as the estimation given by Duval in the Lyndon case.

Bibliographie

1. P. Andary, *Finely homogeneous computations in free Lie algebras*. Technical Report LIR95.13, LIR, Université de Rouen, 1995.
2. P. Andary et D. Krob, *Decomposition of Lie polynomials over* $\lambda(\mathcal{L})$. Technical Report LIR95.12, LIR, Université de Rouen, 1995.
3. J. P. Borel et F. Laubie, *Quelques mots sur la droite projective réelle*, J. Théor. Nombres Bordeaux **5** (1993), 23–51.
4. J. P. Duval, *Factorizing Words over an Ordered Alphabet*, J. Algorithms **4** (1983), 363–381.
5. J. P. Duval, *Génération d'une section des classes de conjugaison et arbre de mots de Lyndon de longueur bornée*, Theoret. Comput. Sci. **60** (1988), 255–283.
6. A. Dvoretzky et I. Motzkin, *A Problem of Arrangements*, Duke Math. J. **14** (1947), 305–313.
7. S. Feretić et D. Svrtan, *Combinatorics of Diagonally Convex Directed Polyominoes*, Proceedings of the 6th Conference on Formal Power Series and Algebraic Combinatorics (New Brunswick, NJ, 1994), Discrete Math. **157** (1996), 147–168.
8. D. Foata, *Étude algébrique de certains problèmes d'analyse combinatoire et du calcul des probabilités*, Publ. Inst. Statist. Univ. Paris **14** (1965), 81–241.
9. H. Jacquet, *Factorisations de Spitzer-Foata*. Technical Report 1133-96, LaBRI, Université Bordeaux I, 1996.
10. M. Kawski, *Nonlinear Control and Combinatorics on Words*, Geometry of Nonlinear Feedback and Optimal Control (B. Jacubczyk et W. Respondek, eds.), 1995.
11. J. Labelle, *Langages de Dyck généralisés*, Ann. Sci. Math. Québec **17** (1993), 53–64.
12. J. Labelle, *On pairs of non-crossing generalized Dyck paths*, J. Statist. Plann. Inference **34** (1993), 209–217.
13. J. Labelle, *Lattice paths with or without horizontal steps*, Congr. Numer. **101** (1994), 233–241.
14. J. Labelle et N. Y. Yeh, *Generalized Dyck Paths*, Discrete Math. **82** (1990), 1–6.
15. F. Laubie et E. Laurier, *Calcul de multiples de mots de Christoffel*. C. R. Acad. Sci. Paris Sér. I Math. **320** (1995), 765–768.
16. E. Laurier, *Opérations sur les mots de Christoffel*, Thèse de doctorat, Université de Limoges, Limoges, 1995.
17. M. Lothaire, *Combinatorics on Words*. Addison-Wesley, Reading, Mass., 1983.
18. G. Melançon, *Combinatoire des bases des algèbres de Lie libres*, Journées thématiques Medicis «Calculs de Lie» (D. Krob, G. Jacob et P. V. Koseleff, eds.), à paraître.
19. C. Reutenauer, *Free Lie Algebras*, Oxford University Press, Oxford, UK, 1993.
20. R. Siromoney et L. Matthew, *A public key cryptosystem based on Lyndon words*, Inform. Process. Lett. **35** (1990), 33–36.
21. F. Spitzer, *A combinatorial lemma and its application to probability theory*, Trans. Amer. Math. Soc. **82** (1956), 441–452.
22. X. G. Viennot, *Algèbres de Lie libres et monoïdes libres*, Lecture Notes in Math., vol. 691, Springer-Verlag, 1978.

H. Jacquet et G. Melançon
LaBRI, Université Bordeaux I
351, Cours de la Libération
33405 Talence Cedex
France