Mémoire

Auteur : Dallemagne, Joseph
Promoteur(s) : Charlier, Emilie
Faculté : Faculté des Sciences
Diplôme : Master en sciences mathématiques, à finalité didactique
Année académique : 2023-2024
URI/URL : http://hdl.handle.net/2268.2/19950

Faculté des Sciences
Département de Mathématique

Algorithmes de factorisation de monoïdes libres

Mémoire de fin d'études présenté en vue de l'obtention du titre de
Master en Sciences Mathématiques, à finalité didactique

Année académique 2023-2024

Auteur :
Joseph Dallemagne

Promoteur :
Émilie Charlier

Remerciements

Tout d'abord, je tiens à remercier ma promotrice de mémoire Émilie Charlier. Merci pour m'avoir proposé ce sujet qui fait partie de mes préférés dans les mathématiques. Un grand merci pour les discussions et les remarques qui m'ont permis d'arriver au bout de ce travail.

Merci à M.Schneiders, M.Leroy, Md. Haesbroeck et Md.Stipulanti de faire partie de mon jury de défense de mémoire avec Md.Charlier. Vous m'avez tous et toutes fait découvrir des sujets mathématiques d'horizons différents ce qui m'a mené à la découverte de certains pans des mathématiques.

Je remercie également tous les professeur(e)s et assistant(e)s du département de Mathématique de l'Université de Liège pour les différents cours que vous avez dispensés et qui m'ont aidé dans la prise de conscience de mes forces et faiblesses en la matière. Cela m'a aidé à m'améliorer et à être plus rigoureux dans les définitions de notions et les justifications de résultats. Je remercie également M.Boigelot, M.Debruyne, M.Louppe, M.Louveaux et M.Geurts pour m'avoir ouvert l'esprit à différents domaines de l'informatique, ce qui a pu m'aider dans la réalisation du dernier chapitre de ce mémoire.

Je tiens à remercier les différentes personnes que j'ai rencontrées tout au long de mes études à l'université. C'est grâce à chacun de vous que je suis devenu la personne que je suis aujourd'hui. Merci tout particulièrement à Thelma pour l'aide qu'elle m'a fournie tout au long de ces cinq années. Merci aussi à Marie pour tous les projets que l'on a réalisés ensemble ainsi que pour les discussions et remarques durant les cours.

Je tiens particulièrement à remercier ma famille qui m'a soutenu durant mon parcours universitaire de par leurs présences et/ou leurs conseils. Merci à ma maman pour toutes les relectures des travaux afin de corriger les fautes d'orthographe ainsi que les remarques sur les soi-disant "hiéroglyphes" apparaissant dans les différents cours et travaux. Je remercie aussi ma petite amie pour le soutien psychologique apporté durant ma formation. Grâce à celui-ci, j'ai pu tenir le coup tout au long de mon cursus. Merci à toi !

Enfin, ce mémoire est pour toi papa ...

Table des matières

Chapitre 1

Introduction

En arithmétique, il existe un théorème assurant qu'il est possible de factoriser un nombre entier $n \geq 2$ en un produit de nombres premiers : le Théorème Fondamental de l'Arithmétique (TFA). En mathématiques discrètes, plus particulièrement en combinatoire des mots, il est possible de retrouver un équivalent des nombres premiers pour le TFA : les factorisations complètes. Ces dernières sont des sous-ensembles F de $\mathcal{A}^*$ totalement ordonné par $\prec$ qui permettent de factoriser de manière unique n'importe quel mot de w de $\mathcal{A}^*$ en respectant l'ordre $\prec$. Dans ce travail, on considère le monoïde libre $\mathcal{A}^*$ où $\mathcal{A}$ est un alphabet. Tout l'étude présentée ci-après a donc été réalisée sur l'ensemble des mots finis.

Le premier chapitre de ce mémoire permettra de se familiariser avec une famille de mots introduite en 1958 : les mots de Lyndon. En effet, cette année-là, les mathématiciens Roger Lyndon, Ralph Fox et Kuo-Tsai Chen ont publié leur article *Free différential calculus IV : The quotient groups of the lower central series* (cf. [3]). Dans cet article, les auteurs cherchaient à développer un algorithme de calcul du groupe formé par le quotient de deux termes consécutifs d'une série centrale décroissante (*lower central series* en anglais). Lors de cette recherche, ils ont défini les suites standards (*standard sequences*) qui sont, de nos jours, appelées les mots de Lyndon. Ces mots de Lyndon possèdent de nombreuses propriétés qui sont parfois utilisées comme définition de ces derniers. Ce chapitre a pour but de présenter cette famille de mots ainsi que plusieurs propriétés associées à l'aide des articles [2], [3] et [9]. Parmi ces dernières, la cardinalité et le lien avec les suites de l'OEIS seront étudiés. Le résultat central de cette partie est le théorème de Chen-Fox-Lyndon qui stipule que tout mot w de $\mathcal{A}^*$ peut se factoriser en un produit décroissant de mots de Lyndon. Ce théorème permettra donc d'affirmer que l'ensemble des mots de Lyndon muni de l'ordre $\leq_{\text{lex}}$ forme une factorisation complète de $\mathcal{A}^*$.

De ce théorème de Chen-Fox-Lyndon, il est donc possible de définir les mots de Lyndon. Si un mot w possède une factorisation respectant le théorème de Chen-Fox-Lyndon et que cette factorisation est de longueur 1, alors w est un mot de Lyndon. Partant de cette constatation, un mathématicien allemand du nom de Darij Grinberg se posa la question suivante (cf. [7]) : que se passerait-il si l'on changeait le mot "décroissant" en "croissant" ? Grâce au travail d'Émilie Charlier, Manon Stipulanti et Manon Philibert (cf. [2]), la question de Grinberg a pu être résolue. En effet, cette nouvelle définition récursive permet d'introduire la famille de mots de Nyldon. Ce sont ces mots qui sont au centre de ce mémoire. Ce deuxième chapitre permet de

présenter cette nouvelle famille de mots en les définissant et en donnant plusieurs de leurs propriétés. De plus, il est possible de comparer la famille des mots de Lyndon et celle des mots de Nyldon. Effectivement, une des questions qui se posent est de savoir si le changement de définition que Darij Grinberg proposait a complètement modifié les propriétés qu'avaient les mots de Lyndon. Par exemple, une des propriétés invariantes est le fait que les mots de Nyldon muni de l'ordre lexicographique $>_{\text{lex}}$ forme également une factorisation complète de $\mathcal{A}^*$.

Les ensembles des mots de Lyndon et Nyldon sont inclus dans des ensembles plus grands : les ensembles de Lazard et les ensembles de Hall. Dans une première partie de ce troisième chapitre, les ensembles de Hall seront présentés en suivant les définitions de G.Mélançon (cf. [13]). En particulier, il existe un théorème de factorisation unique permettant la factorisation en mots de Hall de tout mot w dans $\mathcal{A}^*$. Dans un second temps, ce sont les ensembles de Lazard qui seront abordés. En particulier, ce sont les deux types d'ensembles de Lazard (gauche et droite) qui seront étudiés avant de les mettre en corrélation avec les mots de Lyndon et ceux de Nyldon. De plus, il est possible de lier les ensembles de Lazard à droite et les ensembles de Hall de G.Mélançon. Nous verrons que cette précision a toute son importance car il existe une seconde définition des ensembles de Hall qu'à décider de suivre G.Viennot dans son travail (cf. [18]). Enfin, une petite discussion sur l'ordre $\prec$ d'un ensemble de Lazard sera faite afin de montrer l'importance de l'ordre dans le choix de l'élément à extraire dans la procédure permettant de construire les différents ensembles.

Enfin, tout au long de ce travail, plusieurs algorithmes de factorisation seront présentés explicitement ou implicitement. Le but du dernier chapitre de ce mémoire est de les rassembler, de les coder et d'étudier leur complexité dans le pire cas possible. Cela inclut les algorithmes de Mélançon, de Duval et un algorithme découlant du théorème de Chen-Fox-Lyndon. Un autre but de ce chapitre est de voir s'il est possible d'optimiser ces algorithmes à l'aide des différents résultats développés dans les chapitres précédents ou de structures de données connues en informatique.

Chapitre 2

Contextualisation : les mots de Lyndon

En 1958, un article nommé *Free differential calculus IV. The quotient groups of the lower central series* écrit par les mathématiciens Kuo-Tsai Chen, Ralph Fox et Roger Lyndon est publié dans *The Annals of Mathematics*. L'objectif de cet article était de trouver un algorithme permettant de calculer les quotients G_n/G_{n+1} d'une *lower central series* $\ldots \subset G_3 \subset G_2 \subset G_1 = G$ tel que $G_{n+1} = [G_n, G], \forall n \geq 1$ où $[.,.]$ est le commutateur. L'opération $[.,.]$ est définie sur G de la manière suivante : $[x, y] = x^{-1}y^{-1}xy, \forall x, y \in G$. Pour ce faire, les auteurs ont, entre autres, défini les *standard sequences* afin de les aider à atteindre cet objectif. De nos jours, ces *standard sequences* portent le nom de **mots de Lyndon** et possèdent plusieurs définitions équivalentes. Pour le lecteur curieux, de simples recherches permetttent d'avoir les définitions suivantes :

Définition 2.1.

1. Un mot de Lyndon est un mot qui est strictement minimal (au sens de l'ordre lexicographique) parmi tout ses conjugués (ou permutés circulaires). Strictement minimal signifie ici qu'il n'appartient qu'une seule fois dans la liste de ses permutés ; il est donc forcément primitif, c'est-à-dire n'est pas puissance entière d'un autre mot.
2. De manière équivalente, un mot de Lyndon w est caractérisé par la propriété suivante : w est toujours lexicographiquement plus petit que tout ses suffixes non triviaux. En d'autres termes, w est un mot de Lyndon si et seulement si, pour toute factorisation $w = uv$ en deux mots, avec u et v non vide, on a $w < v$.
3. Une variante de cette caractérisation est la suivante : un mot w de longueur n est un mot de Lyndon si et seulement si, pour tout i avec $0 < i < n$, le préfixe de longueur i de w est strictement inférieur au suffixe de longueur i de w.
4. Ces définitions impliquent que si w n'est pas une lettre, alors w est un mot de Lyndon si et seulement s'il existe deux mots de Lyndon u et v tels que $u < v$ et $w = uv$.

De nos jours, les mots de Lyndon sont connus, entre autres, pour le théorème de Chen-Fox-Lyndon. Ce théorème démontre qu'il est possible de factoriser n'importe quel mot d'un monoïde en un produit décroissant de mots de Lyndon plus courts. Cependant, l'origine de ce théorème est inconnue. En effet, d'après [1], il a fallu

attendre 1965 et l'article *On a Factorisation of Free Monoids* du mathématicien Marcel-Paul Schutzenberger pour montrer que les mots de Lyndon formaient une factorisation de monoïde. Cependant, d'autres mathématiciens remarquèrent qu'il était possible de déduire une telle propriété à l'aide des résultats de l'article [3] sans que cette dernière soit explicitement notée.

2.1 Définitions préliminaires

Avant de continuer ce chapitre, un rappel de certaines notions semble opportun. Une des utilités de ce dernier est de fixer les notations qui vont être utiles dans la suite de ce mémoire. De ce fait, les définitions d'alphabet, de mot, d'étoile de Kleene, de concaténation de mots, de préfixe et de suffixe sont les suivantes :

Définition 2.2.

1. Un **alphabet** $\mathcal{A}$ est un ensemble fini et non vide dont les éléments sont appelés lettres ou symboles.
2. Un **mot** fini sur $\mathcal{A}$ est une suite finie $(a_i)_{1\leq i\leq n}$ où a_i est un élément de $\mathcal{A}$. On notera ϵ le mot vide ou mot de longueur nulle.
3. Soit $\mathcal{A}$ un alphabet. On définit $\mathcal{A}^* = \bigcup_{n\geq 0} \mathcal{A}^n$ où $\mathcal{A}^n$ est l'ensemble des mots de longueur n. On appelle $\mathcal{A}^*$ **l'étoile de Kleene de** $\mathcal{A}$.
4. On définit la longueur d'un mot w, notée $|w|$, de la manière suivante : si $w = a_1 \cdots a_n$, alors $|w| = n$. On notera ϵ le mot de longueur nulle.
5. Soit $\mathcal{A}$ un alphabet. On définit le **produit de concaténation** de la manière suivante :
$$\cdot : \mathcal{A}^* \times \mathcal{A}^* \to \mathcal{A}^* : (u, v) \mapsto uv.$$
6. Soit w un mot sur un alphabet $\mathcal{A}$ donné. On dira que u est un **préfixe** (resp. **suffixe**) de ω si $\omega = uv$ (resp. $\omega = vu$) avec $v \in \mathcal{A}^*$. On note $\mathrm{Pref}(w)$ l'ensemble des préfixes de w (resp. $\mathrm{Suff}(w)$ l'ensemble des suffixes de w). De plus, si u est différent de w, alors u est un préfixe propre de w (resp. suffixe propre de w).

Les éléments de $\mathcal{A}$ peuvent être ordonnés via une relation d'ordre $\leq$. Pour rappel, une relation d'ordre est définie comme suit :

Définition 2.3. Une **relation d'ordre** $\leq$ sur un ensemble E est une relation

- réflexive : $\forall x \in E, x \leq x$
- antisymétrique : $\forall x, y \in E, x \leq y$ et $y \leq x \Rightarrow x = y$
- transitive : $\forall x, y, z \in E$, si $x \leq y$ et $y \leq z \Rightarrow x \leq z$

Donc, quitte à permuter les éléments, l'alphabet $\mathcal{A}$ de taille n est de la forme $\mathcal{A} = \{a_1, \ldots, a_n\}$ avec $a_1 < a_2 < \ldots < a_n$.

Remarque 2.4. Il est possible de raffiner cette notion d'ordre en ordre total. Un ordre $\leq$ est un ordre total sur un ensemble E si, pour tout $x, y \in E$, $x \leq y$ ou $y \leq x$.

Ensuite, pour définir les mots de Lyndon, les notions d'ordre lexicographique, de classe de conjugaison et de mots primitifs sont importantes.

Définition 2.5. Soient $\mathcal{A}$ un alphabet et u, v deux mots sur $\mathcal{A}^*$.

1. On dira que u est **plus petit lexicographiquement** que v si et seulement si

   u est un préfixe de v,
   ou $u = pas$ et $v = pbt$ où $p, s, t \in \mathcal{A}^*, a, b \in \mathcal{A}$ et $a < b$

   Dans ce cas, on notera $u \leq_{\text{lex}} v$ où $\leq_{\text{lex}}$ est appelé **ordre lexicographique**.
2. On définit la **relation binaire de conjugaison** $\sim$ de la manière suivante :

   $$u \sim v \text{ s'il existe } x, y \in \mathcal{A}^* \text{ tels que } u = xy \text{ et } v = yx.$$

   On dira que u et v sont **conjugués**. La relation $\sim$ est une relation d'équivalence et on note $[u]_\sim$ la **classe de conjugaison de** u.
3. On dit que u est **primitif** s'il n'est pas de la forme $u = w^k$ où $w \in \mathcal{A}^*, u \neq \epsilon, k \geq 2$.

Exemple 2.6. Considérons $\mathcal{A} = \{a, b\}$ où $a < b$.

— $011 \leq_{\text{lex}} 011010100$ et $011 \leq_{\text{lex}} 10$

— Les mots 01 et 10 sont conjugués.

— Le mot 0110 est primitif mais 010101 ne l'est pas

Grâce aux différentes définitions, il est possible de montrer que l'ordre lexicographique $\leq_{\text{lex}}$ est un ordre total.

Proposition 2.7. *L'ordre lexicographique $\leq_{lex}$ est un ordre total.*

Démonstration. Montrons tout d'abord que $\leq_{\text{lex}}$ est une relation d'ordre.

1. Réflexivité
   Il est trivial que $x \leq_{\text{lex}} x$ car x est un préfixe de x.
2. Antisymétrie
   Supposons avoir $x \leq_{\text{lex}} y$ et $y \leq_{\text{lex}} x$. On a donc deux cas à traiter.
   a) Supposons que x soit un préfixe de y. Comme x est un préfixe de y, la seule manière de satisfaire $y \leq_{\text{lex}} x$ est que y soit un préfixe de x. Donc, on a bien $x = y$.
   b) Supposons maintenant qu'il existe $p, s, t \in \mathcal{A}^*$ tels que $x = p0s$ et $y = p1s$. De plus, comme $y \leq_{\text{lex}} x$, on remarque aisément que $x = y$.
3. Transitivité
   Soient $x, y, z \in \mathcal{A}^*$. Supposons $x \leq_{\text{lex}} y$ et $y \leq_{\text{lex}} z$. Considérons différents cas.
   a) Si y est un préfixe de z, on aura toujours $x \leq_{\text{lex}} z$. En effet, si x est un préfixe de y, alors x est un préfixe de z et $x \leq_{\text{lex}} z$. De plus, si $|y| = n$, $|z| = m$ et $x = p0s$ et $y = p1t$ avec $p, s, t \in \mathcal{A}^*$, $x \leq_{\text{lex}} z$ car $z = yv = p1tv$ où $v = z_{n+1} \dots z_m$.
   b) Sinon, il existe $p, s, t, p', s', t' \in \mathcal{A}$ tels que $x = p0s$, $y = p1t = p'0s'$ et $z = p'1t'$. Raisonnons sur les longueurs de p et p'.
   *b*1) Si $|p| < |p'|$, alors on remarque que x est un préfixe de z donc $x \leq_{\text{lex}} z$.

*b*2) Le cas où $|p| = |p'|$ est impossible car y ne peut pas s'écrire avec deux lettres différentes à la même position.

*b*3) Enfin, traitons le cas $|p| > |p'|$. Il existe donc $u \in \mathcal{A}^n$ pour un certain $n \in \mathbb{N}$ tel que $p = p'u$. Dès lors, on peut réécrire x, y et z de la manière suivante : $x = p'u0s$, $y = p'u1t$ et $z = p'1t'$. Cependant, comme y s'écrit également $p'0t'$, le mot u commence forcément par 0. Donc, $x = p'0S$ où $S = u_2 \cdots u_n 0s$. Au final, on obtient $x \leq_{\text{lex}} z$.

4. Totalité
Étant toujours possible de comparer des mots entre eux (comparaison lettre par lettre), cette relation d'ordre est trivialement totale.

□

Proposition 2.8. *La relation de conjugaison $\sim$ est une relation d'équivalence.*

Démonstration. Pour montrer que $\sim$ est une relation d'équivalence, montrons que $\sim$ est réflexif, symétrique et transitif.

1. Réflexivité
On a effectivement que $w \sim w$ pour tout $w \in \mathcal{A}^*$ en prenant $u = w$ et $v = \epsilon$.
2. Symétrie
Au vu du caractère symétrique de la définition, on obtient bien que $w \sim w' \Rightarrow w' \sim w$ pour tout $w, w' \in \mathcal{A}^*$.
3. Transitivité
Si, pour tout $w, w', z \in \mathcal{A}^*$, $w \sim w'$ et $w' \sim z$, montrons que $w \sim z$. En conséquence de la définition, il existe $u, v, u', v' \in \mathcal{A}^*$ tel que $w = uv$, $w' = vu = u'v'$ et $z = v'u'$. Afin de démontrer cette propriété, considérons cinq cas.
   a) Si $|u'| = |v|$, on obtient les égalités $u = v'$ et $v = u'$. Dès lors, $w = uv = z$ d'où $w \sim z$.
   b) Si $|u'| > |v|$, alors il existe $x \in \mathcal{A}^*$ tel que $u' = vx$. Par conséquent, le mot w' peut être réécrit $w' = vu = vxv'$. Dès lors, on obtient que $u = xv'$. Donc, en réécrivant les mots w et z avec cette nouvelle égalité, on obtient $w = uv = xv'v$ et $z = v'u' = v'vx$. Enfin, on obtient $w \sim z$.
   c) Si $|u'| < |v|$, par un raisonnement similaire au cas *b*), on obtient aisément $w \sim z$

□

Grâce aux différentes notions définies ci-avant, les mots de Lyndon sont définis de la manière suivante :

Définition 2.9. Soit w un mot défini sur un alphabet $\mathcal{A}$. On dira que w est un **mot de Lyndon** s'il est primitif et minimal (pour l'ordre lexicographique) dans sa classe de conjugaison.

2.2 Équivalences des définitions

Maintenant que les mots de Lyndon ont été définis proprement, une question est de savoir le lien avec les autres définitions données par la définition 2.1. Pour ce faire, un raisonnement similaire à celui présenté dans l'article [3] permet de répondre

à la question. Pour commencer, les auteurs rappellent quelques propriétés de l'ordre lexicographique bien utiles dans plusieurs démonstrations.

Proposition 2.10. *Soient $\mathcal{A}$ un alphabet et $x, y, z, t \in \mathcal{A}^*$. L'ordre lexicographique respecte les propriétés suivantes :*

(L1) $xy <_{lex} xz \Leftrightarrow y <_{lex} z$

(L2) Si $y <_{lex} x <_{lex} yz$, *alors* $x = yt$ *où* $t <_{lex} z$.

(L3) Si $x <_{lex} y$ *mais que* $|x| \geq |y|$, *alors* $xz <_{lex} yt$.

L'équivalence entre les définitions découle du lien entre les différents ensembles définis ci-dessous.

Définition 2.11. Soit $\mathcal{A}$ un alphabet ordonné par $<$. On définit les ensembles $\mathcal{A}', \mathcal{A}'', \mathcal{A}'''$ et $\mathcal{A}''''$ de la manière suivante :

($\mathcal{A}'$) $w \in \mathcal{A}'$ si $w \in \mathcal{A}$ ou si $w = xy$ où $x, y \in \mathcal{A}'$ et $x <_{\text{lex}} y$.

($\mathcal{A}''$) $w \in \mathcal{A}''$ si w est plus petit lexicographiquement que tous ses suffixes propres ($w < s$, $\forall s$ suffixe propre de w). Les éléments de $\mathcal{A}''$ sont appelés *standard sequences*.

($\mathcal{A}'''$) $w \in \mathcal{A}'''$ si w est plus petit lexicographiquement que toutes ses permutations cycliques non-triviales.

Remarque 2.12. Pour tout mot w sur $\mathcal{A}^{\geq 2}$, il existe au moins un suffixe propre s de w qui appartient à $\mathcal{A}''$. Par exemple, si $w = w_1 \cdots w_n$, alors w_n est un suffixe propre de w appartenant à $\mathcal{A}''$. Donc, il existe un suffixe propre y de w appartenant à $\mathcal{A}''$ qui est de longueur maximale. En conséquence, la **factorisation standard** de w est la factorisation $w = xy$ où y est le suffixe propre de w appartenant à $\mathcal{A}''$ de longueur maximale.

Avant de montrer le lien entre ces trois ensembles (propositon 2.15), les deux lemmes ci-dessous sont nécessaires.

Lemme 2.13. *Si $x \in \mathcal{A}^*$, $y \in \mathcal{A}''$ et s est un suffixe propre de xy tel que $x <_{lex} s$, alors $xy <_{lex} s$.*

Démonstration. Supposons qu'il existe s un suffixe propre de xy tel que $s \leq_{\text{lex}} xy$. Dès lors, comme $x <_{\text{lex}} s$ et par L2, on a $s = xt$ avec $t \leq_{\text{lex}} y$. De plus, il est trivial de montrer que $|t| \leq |y|$. Dans le cas contraire, $|s| > |xy|$ ce qui est impossible. Donc, comme $s = xt$ est un suffixe propre de xy, par $L1$, on obtient $t <_{\text{lex}} y$ ce qui implique que y ne peut pas appartenir à $\mathcal{A}''$. □

Lemme 2.14. *Si xy est la factorisation standard d'un élément de $\mathcal{A}''$, alors $x \in \mathcal{A}''$.*

Démonstration. Sans perdre de généralité, on peut supposer que $|x| > 1$ et donc x possède un suffixe propre non vide car on a $\mathcal{A} \subset \mathcal{A}''$.
Considérons donc z un suffixe propre non vide de x. Comme y est le plus long suffixe propre de xy qui appartient à $\mathcal{A}''$, le suffixe propre zy de xy ne peut pas appartenir à $\mathcal{A}''$. Par conséquent, il existe un suffixe propre s de zy tel que $s \leq_{\text{lex}} zy$. Par le lemme 2.13, $s \leq_{\text{lex}} z$. Comme s est aussi un suffixe propre de xy et que $xy \in \mathcal{A}''$, on

doit avoir $xy <_{\text{lex}} s$.
Au final, on a $x <_{\text{lex}} xy <_{\text{lex}} s \leq_{\text{lex}} z$ d'où $x \in \mathcal{A}''$. □

Le résultat suivant permet de montrer l'égalité de ces trois ensembles de mots.

Proposition 2.15. *Les ensembles $\mathcal{A}', \mathcal{A}'', \mathcal{A}'''$ sont égaux.*

Démonstration. Procédons par récurrence sur la taille n des mots. On notera $\mathcal{A}'_n$ (resp. $\mathcal{A}''_n$ et $\mathcal{A}'''_n$) les mots de taille n de $\mathcal{A}'$ (resp. $\mathcal{A}''$ et $\mathcal{A}'''$).
Le cas $n = 1$ est trivial car les éléments de $\mathcal{A}$ appartiennent bien aux trois ensembles. Montrons donc l'égalité des ensembles pour les mots de taille au moins 2. Procédons par double inclusion.

$\underline{\mathcal{A}'_n \subset \mathcal{A}''_n}$: Si $w \in \mathcal{A}'_n$, alors $w = xy$ où $x \in \mathcal{A}'_r$, $y \in \mathcal{A}'_s$, $r + s = n$ et $x <_{\text{lex}} y$. Par hypothèse de récurrence, $x \in \mathcal{A}''_r$, $y \in \mathcal{A}''_s$. Montrons maintenant que, pour tout suffixe propre s de w, $w <_{\text{lex}} s$ par définition de $\mathcal{A}''_n$. Par le lemme 2.13, il suffit de montrer que $x <_{\text{lex}} s$. Le suffixe propre s peut être de trois types :

- s est un suffixe propre de y. Comme $y \in \mathcal{A}''_s$, on a $y <_{\text{lex}} s$ et donc $x <_{\text{lex}} y <_{\text{lex}} s$ d'où $x <_{\text{lex}} y$ par transitivité.
- $s = y$ et cela entraine $x <_{\text{lex}} y = s$.
- Si $s = zy$, on a déjà $x <_{\text{lex}} z$ (par définition $\mathcal{A}''_r$) et $x <_{\text{lex}} z <_{\text{lex}} zy = s$.

$\underline{\mathcal{A}''_n \subset \mathcal{A}'_n}$: Si $w \in \mathcal{A}''_n$, nous pouvons considérer sa factorisation standard $w = xy$ où $x \in \mathcal{A}^*$ et $y \in \mathcal{A}''$. Par le lemme 2.14, on a $x \in \mathcal{A}''$ et, par définition, $y \in \mathcal{A}''$. Donc, par hypothèse de récurrence, $x \in \mathcal{A}'$ et $y \in \mathcal{A}'$. De plus, $x <_{\text{lex}} xy <_{\text{lex}} y$ car $xy \in \mathcal{A}''_n$. Donc, $w \in \mathcal{A}'_n$.

$\underline{\mathcal{A}''_n \subset \mathcal{A}'''_n}$: Si $w \in \mathcal{A}''_n$ et si w' est une permutation cyclique non-triviale de w, alors $w = xy$ et $w' = yx$. Comme $w \in \mathcal{A}''_n$, on a $w <_{\text{lex}} y <_{\text{lex}} yx$ donc $w \in \mathcal{A}'''_n$.

$\underline{\mathcal{A}'''_n \subset \mathcal{A}''_n}$: Si $w \in \mathcal{A}'''_n$ et y un suffixe propre quelconque de w tel que $w = xy$, alors $w < yx$.
Supposons $y \leq_{\text{lex}} w$. Comme $y \neq w$, on va supposer $y <_{\text{lex}} w$. Par L2, on obtient $w = yz$ où $z < x$. Puisque $w \in \mathcal{A}'''_n$, on a $w <_{\text{lex}} zy$ donc $z <_{\text{lex}} x <_{\text{lex}} xy <_{\text{lex}} zb$. En utilisant L2 une nouvelle fois, on obtient $x = zs$ où $s <_{\text{lex}} y$. Donc $yx = xzs = ws = xys$ ce qui est impossible car yx est plus court que xys.

□

Remarque 2.16. Grâce à cette proposition 2.15, il est possible de faire le lien entre les différentes définitions données au début de ce chapitre (cf. définition 2.1). En effet, chacune d'entre elle se rapporte à une définition ou un résultat énoncé ci-dessus. Cependant, dans la suite de ce mémoire, les mots de Lyndon seront caractérisés par la définition 2.9 et les autres définitions deviendront des propriétés triviales.

Remarque 2.17. Il est également possible de définir les mots de Lyndon en utilisant la notion de shuffle. Pour toute personne curieuse, l'article [3] effectue le travail proprement.

2.3 Propriétés des mots de Lyndon

Après avoir défini cette famille de mots, un travail naturel est de savoir quel genre de propriétés satisfait-elle.

2.3.1 Nombre de mots de Lyndon de longueur n

Définition 2.18. (Fonction de Möbius) La fonction $\mu : \mathbb{N} \to \{-1, 0, 1\}$ est telle que

$$\forall n \in \mathbb{N}^*,\ \mu(n) = \begin{cases} 0 & \text{si } n \text{ est divisible par un carré différent de 1} \\ (-1)^k & \text{si } n \text{ est le produit de k nombre(s) premier(s) distinct(s)} \end{cases}$$

Le théorème central permettant de démontrer la formule close donnant le nombre de mots de Lyndon de longueur n est le théorème d'inversion de Möbius et s'énonce de la manière suivante.

Théorème 2.19. *(Inversion de Möbius) Soient f et g deux fonctions définies sur $\mathbb{N}_0$ et à valeurs dans $\mathbb{C}$. On a $g(n) = \sum_{d|n} f(d)$ si et seulement si $f(n) = \sum_{d|n} \mu(d) \cdot g(\frac{n}{d})$.*

Remarque 2.20. Afin de bien comprendre le théorème suivant (surtout sa démonstration), nous pouvons observer une propriété intéressante concernant les mots du monoïde $\mathcal{A}^*$. En effet, un mot est primitif ou non. Donc, pour tout mot w de $\mathcal{A}^*$, il existe un mot primitif z de $\mathcal{A}^*$ et un naturel $k \geq 1$ tel que $w = z^k$. Donc, si w est primitif, alors $w = z$ et $k = 1$.

Théorème 2.21. *Le nombre de mots de Lyndon de longueur n sur un alphabet de taille $q < \infty$ est donné par la formule suivante :*

$$\phi_q(n) = \frac{1}{n} \sum_{d|n} \mu(d) \cdot q^{\frac{n}{d}}$$

Démonstration. Posons $\phi_q(n)$ le nombre de classes de conjugaison de mots primitifs de longueur n. Dès lors, par le raisonnement de la remarque 2.20, si w est un mot de longueur n, alors il s'écrit $w = z^k$ et $n = k \cdot d$ et le nombre de conjugués de w est d. Dès lors, si $|\mathcal{A}| = q$, alors

$$q^n = \sum_{d|n} d \cdot \phi_q(d).$$

Enfin, par le théorème 2.19, on a

$$\phi_q(n) = \frac{1}{n} \sum_{d|n} \mu(d) \cdot q^{\frac{n}{d}}.$$

□

Exemple 2.22. Une application de cette formule permet d'obtenir le nombre de mots de Lyndon en fonction d'une longueur n donnée. Dans le cas d'un alphabet binaire $\mathcal{A} = \{0, 1\}$, la formule devient

$$\phi_2(n) = \frac{1}{n} \sum_{d|n} \mu(d) \cdot 2^{\frac{n}{d}}$$

En appliquant cette formule pour les trois premiers naturels, on obtient

- $\phi_2(1) = \mu(1) \cdot 2^1 = 2$ (cela correspond aux mots 0 et 1),

- $\phi_2(2) = \frac{1}{2} \cdot (\mu(1) \cdot 2^2 + \mu(2) \cdot 2^1) = \frac{1}{2} \cdot (4 - 2) = \frac{2}{2} = 1$ (cela correspond au mot 01),
- $\phi_2(3) = \frac{1}{3} \cdot (\mu(1) \cdot 2^3 + \mu(3) \cdot 2^1) = \frac{1}{3} \cdot (8 - 2) = \frac{6}{3} = 2$ (cela correspond aux mots 001 et 011).

2.3.2 Lien avec les suites de l'OEIS

L'OEIS (On-Line Encyclopedia of Integer Sequences) est un site web comprenant un grand nombre de suites mathématiques ayant un grand intérêt pour la recherche. Ce site a été créé en 1995 mais l'élaboration des différentes suites remonte à plus loin dans le passé. Alors qu'il était étudiant, le mathématicien Neil Sloane commença à s'intéresser aux suites entières et décida de les classer dans le cadre d'un travail en combinatoire mathématique. Au départ, il publia sa sélection dans deux ouvrages, *A Handbook of Integer Sequences* en 1973 contenant 2400 suites et *The Encyclopedia of Integer Sequences* en 1995 contenant, quant à lui, 5487 suites différentes. Après ces deux publications, Neil Sloane reçut de nombreuses suites venant de plusieurs mathématiciens et il lui était devenu impossible de pouvoir les classer dans un livre comme les deux premières fois (le nombre de suite dépasse les 16 000 à ce moment-là). Donc, en août 1994, Sloane rend l'OEIS accessible par courrier électronique et, en 1995, l'OEIS fut disponible sur une interface web. Au moment où ces lignes sont écrites (mai 2024), le site contient 372 720 suites différentes.

Les différentes suites sont classées dans l'OEIS grâce à un numéro de série. Par exemple, la suite $A000001$ est la suite telle que le n^e terme est le nombre de groupe d'ordre n. Parmi les autres suites, nous pouvons également retrouver la suite de Fibonacci ($A000045$), le mot infini de Fibonacci ($A003849$) et la suite de Conway, appelé également la suite "look-and-say", grâce au numéro $A005150$.

Enfin, en ce qui concerne les mots de Lyndon, la suite $A001037$ est la suite dont le n^e terme représente le nombre de mots de Lyndon de longueur n sur un alphabet de taille 2. Cependant, la suite $A001037$ n'est pas seulement lié aux mots de Lyndon. En effet, le n^e terme de cette même suite $A001037$ donne le nombre de polynôme irréductible de degré sur $GF(2)$ (corps fini de dimension 2).

Remarque 2.23. Le premier terme de la suite $A001037$ est 1. Certains mathématiciens considèrent que le mot vide ϵ est un mot de Lyndon. Dans le cadre de notre mémoire, ϵ ne sera pas un mot de Lyndon.

2.3.3 Théorème de factorisation de Chen-Fox-Lyndon

Dans la littérature mathématique, plusieurs algorithmes permettant de factoriser un mot de $\mathcal{A}^*$ peuvent être trouvés : du plus naïf au plus complexe. Parmi la myriade de factorisations possibles, un théorème permet de factoriser un mot w en mots de Lyndon : le théorème de Chen-Fox-Nyldon. Ce théorème assure, non seulement, l'existence mais également l'unicité de cette factorisation grâce à un ordre sur les facteurs intervenant dans la factorisation.

D'un point de vue historique, comme expliqué au début de ce chapitre, il faut savoir que ce n'est pas Chen, Fox et Lyndon qui ont énoncé ou démontré ce théorème.

En effet, comme on peut le lire dans [1] (traduit de l'anglais) : "*En fait, ce théorème a une origine imprécise. On l'accord habituellement à Chen, Fox et Lyndon suite à l'article de Schützenberger dans lequel il apparait comme un exemple de factorisation de monoïde libre. En fait, comme l'ai fait remarquer D.Knuth à l'un d'entre nous en 2004, [l'article de Chen, Fox et Lyndon] ne contient pas cette déclaration. Cependant, elle peut être récupérée à partir de leurs recherches.*" C'est pourquoi nous allons démontrer ce théorème en utilisant les résultats de [3] (en particulier, on utilisera la définition 2.11) et avec le formalisme de [9].

Théorème 2.24. *(Chen-Fox-Lyndon) Soit $\mathcal{A}$ un alphabet totalement ordonné. Tout mot $w \in \mathcal{A}^*$ se factorise en mots de Lyndon lexicographiquement décroissants, et ce de manière unique.*

Démonstration. Soit $w \in \mathcal{A}^*$ et notons L l'ensemble des mots de Lyndon défini par la définition 2.9. Il est trivial de montrer que, si $w = a \in \mathcal{A}$, alors $w \in L$. Dès lors, nous pouvons supposer avoir $w \in \mathcal{A}^{\geq 2}$.

Existence : Donc, dans ce cas, on a $w = l_1 \cdots l_n$ avec $n \geq 2$ et $l_i \in L$, $\forall i \in \{1, \ldots, n\}$. Nous savons qu'une telle décomposition existe car tous les éléments de $\mathcal{A}$ appartiennent à L.
Prenons maintenant une factorisation de w telle que le nombre n de terme de cette factorisation soit minimal. Montrons que, pour tout $i \in \{1, \ldots, n-1\}$, $l_{i+1} \leq_{\text{lex}} l_i$. En effet, s'il existe un i dans $\{1, \ldots, n-1\}$ tel que $l_i \leq_{\text{lex}} l_{i+1}$, par définition de $\mathcal{A}'$ et donc de L, on obtient que $l_i l_{i+1} \in L$ ce qui contredit la minimalité de n.

Unicité : Procédons par l'absurde. Supposons avoir $w = l_1 \cdots l_n = l'_1 \cdots l'_n$ où $l_i, l'_i \in L$, $\forall i \in \{1, \ldots, n\}$ et $l_n \leq_{\text{lex}} \ldots \leq_{\text{lex}} l_2 \leq_{\text{lex}} l_1$ et $l'_n \leq_{\text{lex}} \ldots \leq_{\text{lex}} l'_2 \leq_{\text{lex}} l'_1$.
Supposons avoir $|l_1| > |l'_1|$ (l'autre cas se traite de la même manière). Dans ce cas, $l_1 = l'_1 \cdots l'_i u$ où u est un préfixe propre de l'_{i+1}. Dès lors, comme $l_1 \in L$, on a

- $l_1 <_{\text{lex}} u$ par définition de $\mathcal{A}''$.
- $u \leq_{\text{lex}} l'_{i+1}$ par définition de $\leq_{\text{lex}}$.
- $l'_{i+1} \leq_{\text{lex}} l'_1$ par définition de la factorisation.
- $l'_1 <_{\text{lex}} l_1$ par définition de $<_{\text{lex}}$.

Donc, on obtient $l_1 <_{\text{lex}} l_1$, ce qui est absurde.
Au final, en montrant que le cas $|l'_1| > |l_1|$ est également impossible, on a $l_1 = l'_1$ et donc $l_i = l'_i$ pour tout $i \in \{1, \ldots, n\}$ d'où l'unicité de la factorisation.

□

Chapitre 3

Introduction aux mots de Nyldon

Le dernier théorème du chapitre précédent permet de factoriser n'importe quel mot w en produit décroissant de mots de Lyndon. Donc, si w est mot de longueur n sur $\mathcal{A}^*$ où $\mathcal{A}$ est un alphabet totalement ordonné, la factorisation décroissante en mots de Lydon est la suivante :

$$w = w_1 \cdots w_n \text{ avec } w_n \leq_{\text{lex}} w_{n-1} \leq_{\text{lex}} \ldots \leq_{\text{lex}} w_1.$$

Remarque 3.1. Si u est mot de longueur n de $\mathcal{A}^*$ se factorisant en $u_1 \cdots u_n$, on notera la factorisation $(u_1, \ldots, u_n)$. Dès lors, si w est un mot de Lyndon, alors sa factorisation décroissante en mots de Lyndon est (w).

Récemment, en novembre 2014, Darij Grinberg a écrit un article sur le site mathoverflow (cf. [7]) dans lequel il part du théorème de Chen-Fox-Lyndon pour définir une nouvelle famille de mots : les mots de Nyldon.
Pour ce faire, il définit les mots de Nyldon récursivement de la manière suivante :

Définition 3.2. (Famille des mots de Nyldon)
Soit $\mathcal{A}$ un alphabet totalement ordonné par l'ordre $\leq$. Un mot w est un mot de Nyldon si $w \in \mathcal{A}$ ou w ne peut pas être factorisé de manière croissante en mots de Nyldon plus courts. On notera $\mathcal{N}$ la famille des mots de Nyldon.

À partir de cette définition, cela veut dire que, si $w \in \mathcal{A}^+$, alors w est un mot de Nyldon si sa seule factorisation croissante en mots de Nyldon est (w). De plus, en utilisant un raisonnement similaire aux mots de Lyndon (qui sera détaillé plus tard dans ce chapitre), il est possible de trouver une factorisation croissante unique en mots de Nyldon.

Proposition 3.3. *Pour tout mot fini w de $\mathcal{A}^*$, il existe une factorisation $(w_1, \ldots, w_k)$ de w en mots de Nyldon de $\mathcal{A}^*$ telle que $w_1 \leq_{lex} \ldots \leq_{lex} w_k$.*

Démonstration. Soit w un mot de $\mathcal{A}^*$ et $s = (a_1, \ldots, a_n)$ la factorisation de w sous forme de lettres. Posons $f = a_1$ le premier facteur de cette factorisation. Si $a_2 <_{\text{lex}} a_1$, alors on considère $s = (a_1 a_2, a_3 \ldots, a_n)$. Dans ce cas, f devient $a_1 a_2$ et on le compare à a_3. Si $a_1 \leq_{\text{lex}} a_2$, alors s ne change pas et f devient a_2 et on le compare à a_3. La procédure ainsi décrite va donner lieu à une factorisation $s = (w_1, \ldots, w_k)$ avec $w_1 \leq_{\text{lex}} w_2 \leq_{\text{lex}} \ldots\ldots \leq_{\text{lex}} w_k$ d'où le résultat. □

En partant de la définition 3.2, les mots suivants sont des mots de Nyldon sur l'alphabet $\mathcal{A} = \{0, 1\}$:

$$0, 1, 10, 100, 101, 1000, 1001, 1011, 10000, 10001, 10010, 10011, 10110, 10111, ...$$

3.1 Préfixes et suffixes

3.1.1 Préfixes de mots de Nyldon

Chaque mot de Nyldon donnés ci-avant a un préfixe de longueur 2 de la forme 10. Un résultat plus général vient fixer une condition sur les préfixes de longueur 2 des mots de Nyldon sur un alphabet de taille quelconque.

Proposition 3.4. *Soit un alphabet $\mathcal{A} = \{0, 1, \ldots, k\}$ tel que $0 < 1 < \ldots < k$. Tout mot de Nyldon sur $\mathcal{A}^*$ de longueur au moins 2 commence par ij avec $0 \leq j < i \leq k$.*

Démonstration. Procédons par contradiction. Soit $w = iju$ où $u \in \mathcal{A}^*$ et $0 \leq i \leq j \leq k$. Montrons que w ne peut être un mot de Nyldon. Soit $(n_1, \ldots, n_l)$ une factorisation croissante de ju en mots de Nyldon. Donc, nous avons que n_1 commence nécessairement par j. Comme i est, par définition, un mot de Nyldon, $i \leq_{\text{lex}} j \leq_{\text{lex}} n_1$ et $k \geq 1$, on obtient donc que $(i, n_1, \ldots, n_k)$ est une factorisation de w en mots de Nyldon croissants. Cette factorisation est de longueur au moins 2 d'où w n'est pas un mot de Nyldon. □

Ce résultat permet d'accélérer le test d'appartenance d'un mot w à la famille des mots de Nyldon. En effet, au lieu de tester à chaque fois si w possède une factorisation croissante en mots de Nyldon, le fait de regarder si le préfixe de longueur 2 est un préfixe possible pourrait accélérer la décision. Si ce préfixe est interdit, alors on peut conclure que w n'est pas de Nyldon. Dans le cas contraire, on peut appliquer la procédure pour tester si w est bien un mot de Nyldon (un algorithme basé sur les résultats de ce chapitre sera présenté dans le dernier chapitre de ce mémoire).
Par exemple, si on prend les mots de Nyldon sur un alphabet de taille 2, le seul préfixe autorisé est 10. De manière similaire, si l'alphabet choisi est de taille 3, les préfixes autorisés pour les mots de Nyldon sont 10, 20 et 21. Donc, les autres préfixes sont interdit.

Définition 3.5. Un mot p de $\mathcal{A}^*$ est un préfixe interdit si aucun mot de Nyldon sur $\mathcal{A}$ ne commence par p.

Une question intéressante est de savoir si un mot w est de Nyldon en regardant les préfixes de w. Pour les préfixes de longueur 2, grâce à la proposition 3.4, nous pouvons déjà effectuer un premier tri. Cependant, il existe des mots ayant un préfixe de longueur 2 autorisé mais qui ne sont pas de Nyldon. Par exemple, en prenant un alphabet de taille 3 ($\mathcal{A} = \{0, 1, 2\}$), le mot 102 est un mot qui possède un préfixe autorisé mais ce n'est pas un mot de Nyldon.
En suivant l'idée de la proposition 3.4, on obtient le résultat suivant.

Proposition 3.6. *Soit $\mathcal{A}$ un alphabet. Tous les éléments de l'ensemble*

$$F = \{p_1p_2p_3 \in \mathcal{A}^* : p_1 \in \mathcal{N}, p_1 \leq_{lex} p_2 \text{ et, pour tout } u \in \mathcal{A}^* \text{ et pour toute factorisation croissante en mots de Nyldon } (n_1, \ldots, n_k) \text{ de } p_2p_3u, \text{ on a } |n_1| \geq |p_2|\}$$

sont des préfixes interdits.

Démonstration. Soit $p \in F$. Vu la définition de F, nous pouvons décomposer p en $p = p_1p_2p_3$ où $p_1, p_2, p_3 \in \mathcal{A}^*$ et respectent les conditions d'appartenance à F. Montrons que tout mot w commençant par p n'est pas un mot de Nyldon.
Soit $w = pu$ où $u \in \mathcal{A}^*$. Soit $(n_1, \ldots, n_k)$ une factorisation croissante en mots de Nyldon de p_2p_3u. Par définition de F, nous avons $p_1 \leq_{\text{lex}} p_2$ et p_2 est un préfixe de n_1. Par définition de l'ordre lexicographique et la transitivité de la relation $\leq_{\text{lex}}$, nous avons également que $p_1 \leq_{\text{lex}} n_1$. Comme $p_1 \in \mathcal{N}$, nous avons donc $(p_1, n_1, \ldots, n_k)$ une factorisation croissante en mots de Nyldon de w de longueur au moins 2 d'où w n'est pas un mot de Nyldon. □

Exemple 3.7. Par exemple, à l'aide de cette propriété 3.6, le mot 1010 sera un préfixe interdit sur l'alphabet $\mathcal{A} = \{0, 1\}$. En effet, si $p_1 = 10$, $p_2 = 10$ et $p_3 = \epsilon$, alors $p_1p_2p_3 = 1010 \in F$. De manière générale, les préfixes de la forme 10^k10^k sont également des préfixes interdits pour tout $k \in \mathbb{N}$. Dans [2], il est même montré que les mots de la forme 10^k1011, $101^{k+1}01^{k+1}$ et $10^{k+2}110^{k+1}11$ sont des préfixes interdits pour tout $k \in \mathbb{N}$. En effet, en prenant les facteurs p_1, p_2 et p_3 ci-dessous, l'appartenance de tous ces mots à l'ensemble F est triviale.

Préfixe	p_1	p_2	p_3
10^k10^k	10^k	10^k	ϵ
10^k1011	10^k	101	1
$101^{k+1}01^{k+1}$	101^k	101^k	1
$10^{k+2}110^{k+1}11$	$10^{k+2}1$	$10^{k+1}1$	1

3.1.2 Suffixes de mots de Nyldon

Après s'être intéressé aux préfixes des mots de Nyldon, une question naturelle pourrait être de voir si les suffixes de ces mots possèdent aussi des propriétés remarquables. En particulier, les deux propriétés suivantes concernant les suffixes des mots de Nyldon seront utiles dans la suite de ce mémoire.

Proposition 3.8. *Soit $w \in \mathcal{A}^*$ un mot de Nyldon. Tout suffixe propre s de w tel que s est un mot de Nyldon est lexicographiquement plus petit que w.*

$$\forall s \in \text{Suff}(w) \text{ tel que } s \neq w \text{ et } s \in \mathcal{N}, \text{ on a } s <_{lex} w.$$

Démonstration. Procédons par récurrence sur $|w|$. Si $|w| = 1$, la conclusion est immédiate. Si $|w| = 2$, alors $w = ij$ et on a $j <_{\text{lex}} i <_{\text{lex}} w$ par la proposition 3.4. Supposons maintenant $|w| \geq 3$ et supposons que le résultat est vrai pour tous les mots de Nyldon de longueur inférieure à $|w|$. Procédons par l'absurde et supposons qu'il existe un suffixe propre s de w tel que

$$s \in \mathcal{N} \text{ et } w \leq_{\text{lex}} s. \tag{1}$$

Parmi tous les suffixes propres non vides respectant cette condition 1, on choisit le suffixe s de longueur maximale et on écrit $w = ps$ avec $p \in \mathcal{A}^*$. Montrons que ce suffixe s n'est pas un mot de Nyldon en donnant une factorisation en mots de Nyldon croissants de longueur au moins 2. Premièrement, nous pouvons montrer que $p \notin \mathcal{N}$. De fait, si $p \in \mathcal{N}$ et $p \leq_{\text{lex}} s$, on aurait une factorisation en mots de Nyldon croissants de w qui serait (p, s) et cela contredit l'hypothèse que w est un

mot de Nyldon. Donc, si $p \in \mathcal{N}$, on aurait $s <_{\text{lex}} p <_{\text{lex}} w$ ce qui va à l'encontre de (1) d'où $p \notin \mathcal{N}$.
Soit $(p_1, \ldots, p_k)$ une factorisation en mots de Nyldon croissants de p de longueur $k \geq 2$. Comme $s, w \in \mathcal{N}$, on doit avoir

$$s <_{\text{lex}} p_k. \tag{2}$$

En effet, si $p_k \leq_{\text{lex}} s$, alors $(p_1, \ldots, p_k, s)$ serait une factorisation en mots de Nyldon croissants de longueur $k+1 \geq 2$ de w. De plus, $p_k s \notin \mathcal{N}$ car, dans le cas contraire, $(p_1, p_2, \ldots, p_k s)$ serait une factorisation en mots de Nyldon croissants de longueur $k \geq 2$.
Notons $(n_1, \ldots, n_l)$ une factorisation en mots de Nyldon croissants de longueur $l \geq 2$ de $p_k s$. Montrons qu'il existe un $i \in \{1, \ldots, l\}$ et $x, y \in \mathcal{A}^+$ tel que

$$n_i = xy, \quad p_k = n_1 \cdots n_{i-1} x \quad \text{et } s = y n_{i+1} \cdots n_l$$

En effet, en procédant par l'absurde, supposons avoir $p_k = n_1 \cdots n_j$ et $s = n_{j+1} \cdots n_l$ avec $1 \leq j \leq l-1$. Comme ces deux mots sont des mots de Nyldon, cela implique que $p_k = n_1$ et $s = n_2$ et donc $l = 2$. Mais, comme $n_1 \leq_{\text{lex}} n_2$ par définition d'une factorisation en mots de Nyldon croissants, cela contredit la condition 2.
De plus, il est même possible de montrer que $i \leq l-1$. De fait, si on suppose que $i = l$, on a $y = s$ et $n_l = xs$. Dès lors, en utilisant l'hypothèse de récurrence, nous obtenons que $s <_{\text{lex}} n_l$. De là, en utilisant la condition 1, on en déduit que n_l est un suffixe de Nyldon propre de w plus long que s tel que $w <_{\text{lex}} n_l$ ce qui est impossible car s est le plus long tel suffixe.
Prenons maintenant $(y_1, \ldots, y_t)$ une factorisation en mots de Nyldon croissants de longueur $t \geq 1$ de y. Dès lors, y_t est un mot de Nyldon et un suffixe propre de n_i qui est lui-même un mot de Nyldon. Comme $|n_i| < |w|$, l'hypothèse de récurrence permet d'assurer $y_t <_{\text{lex}} n_i$. De plus, nous avons $n_i \leq_{\text{lex}} n_{i+1}$. Au final, nous avons $(y_1, \ldots, y_t, n_{i+1}, \ldots, n_l)$ une factorisation en mots de Nyldon croissants de longueur au moins 2 de s, ce qui contredit le fait que s soit un mot de Nyldon. □

Proposition 3.9. *Soit $w \in \mathcal{A}^+$ et $(n_1, \ldots, n_k)$ une factorisation de longueur en mots de Nyldon croissants de w. Alors, n_k est le plus long suffixe de Nyldon de w.*

Démonstration. Notons s le plus long suffixe de w tel que $s \in \mathcal{N}$. Si w est un mot de Nyldon, alors la longueur de la factorisation est 1 et $s = w = n_k$. Supposons maintenant que w n'est pas un mot de Nyldon . Dès lors, s est un suffixe propre de w et $k \geq 2$. Nous pouvons donc écrire $w = ps$. Comme n_k est un mot de Nyldon et vu le choix de s, on a $|n_k| \leq |s|$. Montrons que $|s| = |n_k|$ pour conclure. Supposons que $|n_k| < |s|$. Comme s est un mot de Nyldon, on ne peut avoir $s = n_i \cdots n_k$ avec $i < k$. Donc, il existe un $i \in \{1, \ldots, k-1\}$ et $x, y \in \mathcal{A}^+$ tel que $n_i = xy$, $p = n_1 \cdots n_{i-1} x$ et $s = y n_{i+1} \cdots n_k$. Prenons $(y_1, \ldots, y_t)$ une factorisation de longueur $t \geq 1$ en mots de Nyldon croissants de y. Par la proposition 3.8, on obtient que $y_t <_{\text{lex}} n_i$ car y_t est un suffixe propre de n_i. De plus, comme $n_i \leq_{\text{lex}} n_{i+1}$, on trouve que $(y_1, \ldots, y_t, n_{i+1}, \ldots, n_k)$ est une factorisation de longueur au moins 2 en mots de Nyldon croissants de s ce qui contredit le fait que s soit de Nyldon. Au final, $|n_k| = |s|$ et donc $s = n_k$. □

Remarque 3.10. Pour un $w \in \mathcal{A}^*$ donné et une factorisation en mots de Nyldon croissants donnée, il n'est pas forcément vrai que le premier terme de cette factorisation soit le préfixe le plus long. Prenons $w = 10110111111$ par exemple. Une factorisation de Nyldon en mots croissants serait $(101, 10111111)$ mais le préfixe le plus long qui est un mot de Nyldon est 10110.

3.2 Unicité de la factorisation en mots de Nyldon

Ainsi, grâce aux deux résultats démontrés ci-dessus, il est possible de trouver un résultat similaire au théorème de Chen-Fox-Lyndon : une factorisation croissante unique en mots de Nyldon.

Théorème 3.11. *Pour tout mot fini w sur un alphabet $\mathcal{A}$, il existe une unique factorisation en mots de Nyldon $(n_1, \ldots, n_k)$ telle que $w = n_1 \cdots n_k$ et $n_1 \leq_{lex} n_2 \leq_{lex} \ldots \leq_{lex} n_k$.*

Démonstration. L'existence d'une telle factorisation est assurée par la proposition 3.3. Montrons donc l'unicité d'une telle factorisation. Procédons par récurrence sur $|w|$. Pour $|w| = 1$, le résultat est vrai. Supposons maintenant que $|w| \geq 2$ et que le résultat est vrai pour les mots de longueurs plus petites que w. Si w est un mot de Nyldon, alors (w) est la seule factorisation possible en mots de Nyldon croissants. Si w n'est pas un mot de Nyldon, alors soit $(n_1, \ldots, n_k)$ une factorisation croissante en mots de Nyldon. Dès lors, $k \geq 2$ et, par la proposition 3.9, nous savons que n_k est le plus long suffixe de Nyldon de w. Comme $(n_1, \ldots, n_{k-1})$ est une factorisation en mots de Nyldon croissants d'un mot en longueur plus petit que la longueur de w, les facteurs $n_1, \ldots, n_{k-1}$ sont complètement déterminés par w par hypothèse de récurrence. Donc, $(n_1, \ldots, n_k)$ est la seule factorisation en mots de Nyldon croissants. □

Grâce à l'unicité de la factorisation croissante en mots de Nyldon, il est possible de montrer un autre résultat qui permet de déterminer si un mot w est un mot de Nyldon en comparant s, le plus long suffixe propre de Nyldon de w, et le dernier terme de la factorisation en mots de Nyldon du préfixe p tel que $w = ps$.

Proposition 3.12. *Soit $w \in \mathcal{A}^+$, $w = ps$ avec s le plus long suffixe propre de Nyldon de w et soit $(p_1, \ldots, p_k)$ la factorisation en mots de Nyldon de p. Alors w est un mot de Nyldon si et seulement si $p_k >_{lex} s$.*

Démonstration. Montrons, de manière équivalente, que w n'est pas un mot de Nyldon si et seulement si $p_k \leq_{\text{lex}} s$. Si $p_k \leq_{\text{lex}} s$, alors w n'est pas un mot de Nyldon car sa factorisation croissante en mots de Nyldon est $(p_1, \ldots, p_k, s)$ qui est de longueur au moins 2. Supposons maintenant que w n'est pas un mot de Nyldon. Posons $(w_1, \ldots, w_l)$ a factorisation croissante en mots de Nyldon de w. Alors $l \geq 2$. Par la proposition 3.9, on a $w_l = s$. Cependant, $(p_1 \ldots, p_k)$ et $(w_1, \ldots, w_{l-1})$ sont des factorisations croissantes en mots de Nyldon de p. Par le théorème 3.11, on a $p_k = w_{l-1}$. Dès lors, comme $w_{l-1} \leq_{\text{lex}} w_l$, on obtient $p_k = w_{l-} \leq_{\text{lex}} w_l = s$ d'où le résultat. □

3.3 Factorisation de monoïdes libres

Grâce à cette décomposition unique en mots de Nyldon croissants, la classe des mots de Nyldon est un exemple de factorisation complète du monoïde libre $\mathcal{A}^*$.

Définition 3.13. Soit F un sous-ensemble de $\mathcal{A}^*$ doté d'un ordre total noté $\prec$. Une F-factorisation d'un mot w sur $\mathcal{A}^*$ est une factorisation de w de la forme $(f_1, \ldots, f_k)$ où $f_i \in F$ pour tout $i \in \{1, \ldots, k\}$ et telle que $f_1 \succeq f_2 \succeq \ldots \succeq f_k$. De plus, un ensemble F est une factorisation complète de $\mathcal{A}^*$ si tout mot w de $\mathcal{A}^*$ admet une unique F-factorisation.

Dès lors, en prenant le théorème 3.11, les mots de Nyldon ordonnés par la relation d'ordre total $>_{\text{lex}}$ forme une factorisation complète de $\mathcal{A}^*$.

Remarque 3.14. Une conclusion similaire peut être faite avec les mots de Lyndon. En effet, les mots de Lyndon ordonnés avec la relation d'ordre total $<_{\text{lex}}$ forme une factorisation complète de $\mathcal{A}^*$.

3.4 Mots de Nyldon et mots de Lyndon

Maintenant que les mots de Nyldon sont mieux connus, un travail intéressant à effectuer est de comparer les deux familles $\mathcal{L}$ et $\mathcal{N}$ afin de voir si les propriétés que respectaient les mots de Lyndon restent vraies pour les mots de Nyldon.

3.4.1 Classe de conjugaison et suffixes

Une première différence entre les deux familles de mots concerne les classes de conjugaison. Pour rappel, deux mots u et v appartiennent à la même classe de conjugaison s'il existe deux mots p et s tels que $u = ps$ et $v = sp$. Une des nombreuses définitions des mots de Lyndon est qu'ils sont minimaux dans leur classe de conjugaison. Cependant, les mots de Nyldon ne sont pas des éléments extrêmes parmi leur classe de conjugaison, c'est-à-dire pas minimaux ni maximaux. En effet, si on prend le mot de Nyldon 1001 sur l'alphabet binaire $\{0, 1\}$, il n'est pas minimal (car 0011 l'est) et n'est pas maximal (car 1100 l'est).

Ensuite, une deuxième différence entre ces deux familles de mots concerne la relation entre les mots et leurs suffixes. En effet, par la proposition 3.8, pour tout mot de Nyldon w, $s <_{\text{lex}} w$ pour tout suffixe propre de w. Dans le cas des mots de Lyndon, le résultat devient quel que peu différent.

Proposition 3.15. *Soit w un mot de Lyndon sur un alphabet $\mathcal{A}$. Pour tout suffixe propre s de w tel que s est un mot de Lyndon, on a $w <_{lex} s$.*

Démonstration. Pour prouver cette proposition, nous nous ramènerons à la définition récursive des mots de Lyndon. En particulier, nous allons faire abstraction de toutes les autres propriétés de ces derniers (primitivité, minimalité et unicité de la factorisation en mots de Lyndon décroissants). Procédons maintenant par récurrence sur $|w|$. Si $|w| = 1$, alors le résultat est trivialement vérifié. Si $|w| = 2$, alors $w = ij$ avec $i, j \in \mathcal{A}$. Comme j est le seul suffixe propre de w qui est un mot de Lyndon et comme i, j, w sont de Lyndon, on en déduit que $i <_{\text{lex}} j$ car sinon (i, j)

serait une factorisation de longueur 2 en mots de Lyndon décroissants de w. D'où, $ij = w <_{\text{lex}} j$.
Supposons maintenant que $|w| \geq 3$ et que le résultat est vrai pour les mots de Lyndon dont la longueur est inférieure à $|w|$. Procédons par l'absurde et supposons qu'il existe un suffixe propre s de w tels que s est de Lyndon et

$$s <_{\text{lex}} w \tag{1}$$

Il est bien de rappelé que $s \neq w$ comme s est un suffixe propre de w. Parmi tous les suffixes qui respectent cette condition 1, nous notons s le plus long suffixe et $w = ps$ avec $p \in \mathcal{A}^+$.
Montrons d'abord que p n'est pas un mot de Lyndon. Procédons par l'absurde et supposons que p est un mot de Lyndon. Dès lors, nous devons obligatoirement avoir que $p <_{\text{lex}} s$, sinon (p, s) serait une factorisation de longueur 2 de w en mots de Lyndon décroissants. En utilisant la condition 1, on obtient $p <_{\text{lex}} s <_{\text{lex}} w = ps$. Donc, s doit avoir p comme préfixe. Notons donc $s = ps_1$ avec $s_1 \neq \epsilon$. Ensuite, $s = ps_1 <_{\text{lex}} w = ps$ implique que $s_1 <_{\text{lex}} s$. Comme $|s| < |w|$ et en utilisant l'hypothèse de récurrence, on a s_1 n'est pas un mot de Lyndon et notons $(l_1, \ldots, l_k)$ une factorisation de s_1 en mots de Lyndon décroissants de longueur au moins $k \geq 2$. Comme $s = ps_1$, on doit avoir $p <_{\text{lex}} l_1$, sinon $(p, l_1, \ldots, l_k)$ serait une factorisation de longueur au moins 2 de s en mots de Lyndon décroissants. Donc, on obtient $p <_{\text{lex}} s_1 <_{\text{lex}} s = ps_1$ et p doit être un préfixe de s_1 ce qui implique que p^2 doit être un préfixe de s. De manière similaire, on peut écrire $s_1 = ps_2$ avec $s_2 \neq \epsilon$. Nous pouvons également obtenir que $s_2 <_{\text{lex}} s_1$ et que s_2 n'est pas un mot de Lyndon. De plus, p est également un préfixe de s_2 et donc p^3 est un préfixe de s. En itérant le raisonnement, on obtient que p^n est un préfixe de s pour tout n ce qui est impossible car s est de longueur fini. Comme nous savons maintenant que p n'est pas un mot de Lyndon, prenons $(p_1, \ldots, p_k)$ une factorisation de p en mots de Lyndon décroissants de longueur $k \geq 2$. On a que $p_k <_{\text{lex}} s$ car, dans le cas contraire, on aurait $(p_1, \ldots, p_k, s)$ serait une factorisation de longueur au moins 2 de w en mots de Lyndon décroissants ce qui contredirait le fait que w est de Lyndon. Montrons aussi que $p_k s$ ne peut pas être un mot de Lyndon. En effet, si $p_k s$ est un mot de Lyndon, par maximalité de s, on a $w <_{\text{lex}} p_k s$. De plus, comme $|p_k s| < |w|$, on a, par hypothèse de récurrence, $p_k s <_{\text{lex}} s$. En utilisant les deux inégalités, on obtient $w <_{\text{lex}} s$ ce qui contredit la condition 1.
Posons maintenant $(l_1, \ldots, l_r)$ une factorisation en mots de Lyndon décroissants de $p_k s$ de longueur $r \geq 2$. Similairement à la preuve de la proposition 3.8, on peut montrer qu'il existe $i \in \{1, \ldots, r-1\}$ et $x, y \in \mathcal{A}^+$ tel que

$$l_i = xy, \quad p_k = l_1 \cdots l_{i-1} x \text{ et } s = y l_{i+1} \cdots l_r$$

Soit $(y_1, \ldots, y_t)$ une factorisation de y en mots de Lyndon décroissants de y de longueur au moins 2. Dès lors, y_t est un préfixe propre de l_i qui est un mot de Lyndon. Comme l_i est un mot de Lyndon et $|l_i| < |w|$, l'hypothèse de récurrence implique $l_i <_{\text{lex}} y_t$. Cependant, comme $l_{i+1} <_{\text{lex}} l_i <_{\text{lex}} y_t$, on a que $(y_1, \ldots, y_t, l_{i+1}, \ldots, l_r)$ est une factorisation de s en mots de Lyndon décroissants dont la longueur est au moins 2, ce qui contredit que s est un mot de Lyndon. □

Il est même possible d'aller un cran plus loin en caractérisant les mots de Lyndon en fonction de leurs suffixes propres.

Proposition 3.16. *Soit $w \in \mathcal{A}^+$. Les assertions suivantes sont équivalentes :*

(1) *w est un mot de Lyndon.*

(2) *w est plus petit lexicographiquement que tous ses suffixes propres.*

(3) *w est plus petit lexicographiquement que tous ses suffixes propres qui sont des mots de Lyndon.*

Démonstration. Le passage de (1) à (2) se fait aisément via la proposition 2.15. De plus, il est évident que (2) implique (3). Montrons donc que (3) implique (1) ce qui impliquera que la réciproque de la proposition 3.15 est vraie. Procédons par récurrence sur la taille de w. Si $|w| = 1$, alors le résultat est trivialement vérifié. Supposons que $|w| \geq 2$, que w est lexicographiquement plus petit que tous ses suffixes propres qui sont des mots de Lyndon et que, pour tout mot z plus court que w, si z est plus petit lexicographiquement que tous ses suffixes suffixes propres qui sont des mots de Lyndon, alors z est un mot de Lyndon. Écrivons $w = uv$ avec $u, v \in \mathcal{A}^+$. Montrons que $w <_{\text{lex}} vu$ pour conclure que w est un mot de Lyndon. Si v est un mot de Lyndon, alors $w <_{\text{lex}} v$ par hypothèse et donc $w <_{\text{lex}} vu$. Si v n'est pas un mot de Lyndon, en appliquant l'hypothèse de récurrence à v, on a qu'il existe un suffixe propre s de v tel que s est un mot de Lyndon et $s <_{\text{lex}} v$. Comme s est aussi un suffixe propre de w, on a $w <_{\text{lex}} s$. Dès lors, on obtient que $w <_{\text{lex}} s <_{\text{lex}} v <_{\text{lex}} vu$. □

3.4.2 Cardinalité

L'unicité de la factorisation croissante en mots de Nyldon permet d'exhiber un autre lien qu'ont les mots de Nyldon avec les mots de Lyndon, à savoir la cardinalité des mots d'une certaine longueur de ces deux familles.

Théorème 3.17. *Soit $\mathcal{A}$ un alphabet.*
Si $\mathcal{L}_n$ est l'ensemble des mots de Lyndon de longueur n et $\mathcal{N}_n$ est celui des mots de Nyldon de longueur n, alors, pour tout $n \in \mathbb{N}^*$,

$$\#\mathcal{L}_n = \#\mathcal{N}_n$$

Démonstration. Procédons par récurrence $|w|$. Si $|w| = 1$, on a trivialement $\mathcal{L}_1 = \mathcal{N}_1$ car toutes les lettres de l'alphabet sont à la fois des mots de Nyldon et de Lyndon. Supposons maintenant que le résultat est vrai pour les mots de taille $m < n$ et $n \geq 2$ et montrons qu'il reste vrai pour les mots de longueur n. Comme la factorisation est unique, on déduit que le nombre de mots de longueur n qui ne sont pas des mots de Nyldon est égal au nombre de factorisation en mots de Nyldon croissants $(n_1, \ldots, n_k)$ telle que $|n_1|, \ldots, |n_k| < n$. De manière similaire, par le théorème 2.24, la factorisation en mots de Lyndon décroissants étant unique, on obtient que le nombre de mots de longueur n qui ne sont pas des mots de Lyndon est donné par le nombre de factorisation en mots de Lyndon décroissants $(l_1, \ldots, l_k)$ telle que $|l_1|, \ldots, |l_k| < n$. Dès lors, en utilisant l'hypothèse de récurrence, on obtient $\#(\mathcal{N}^c \cap \mathcal{A}^n) = \#(\mathcal{L}^c \cap \mathcal{A}^n)$ et, donc, $\#\mathcal{N}_n = \#\mathcal{L}_n$. □

3.4.3 Factorisation et factorisation standard

Dans cette partie, nous allons nous concentrer sur les différentes factorisations en mots de Nyldon croissants et en mots de Lyndon décroissants. En particulier, nous parlerons des factorisations standards en mots de Nyldon et mots de Lyndon.

Nous allons également remarquer que certaines propriétés sont similaires et d'autres qui concernent les mots de Lyndon ne sont pas vérifiées dans les mots de Nyldon.

Avant de nous attarder sur ces propriétés, définissons proprement les factorisations standards en mots de Nyldon et de Lyndon via quelques résultats.

Proposition 3.18. *Soit $w \in \mathcal{A}^+$, $w = ps$ avec s le plus long suffixe propre de w tel que s est un mot de Nyldon et soit $(p_1, \dots, p_k)$ une factorisation de p en mots de Nyldon croissants. Le mot w est un mot de Nyldon si et seulement $s <_{lex} p_k$.*

Démonstration. Montrons plutôt la contraposée de ce résultat : w n'est pas un mot de Nyldon si et seulement si $p_k \leq_{\text{lex}} s$. Si $p_k \leq_{\text{lex}} s$, alors $(p_1, \dots, p_k, s)$ est une factorisation de longueur au moins 2 de w en mot de Nyldon croissants donc w n'est pas un mot de Nyldon. Supposons maintenant que w n'est pas un mot de Nyldon. Dès lors, il existe une factorisation de longueur au moins 2 $(w_1, \dots, w_l)$ en mot de Nyldon croissants. Comme $l \geq 2$, on a directement que $w_l = s$ par la proposition 3.9. De plus, on a également $(p_1, \dots, p_k) = (w_1, \dots, w_{l-1})$. Par unicité de la factorisation en mot de Nyldon croissants, on obtient $p_k = w_{l-1}$. Enfin, comme $w_{l-1} \leq_{\text{lex}} w_l$, on a $p_k = w_{l-1} \leq_{\text{lex}} w_l = s$. □

Théorème 3.19. *Soit $w \in \mathcal{A}^+$ et soit $w = ps$ avec s le plus long suffixe propre de w tel que s est un mot de Nyldon. Alors le mot w est un mot de Nyldon si et seulement si p est un mot de Nyldon et $s <_{lex} p$.*

Démonstration. $\Leftarrow$ Il s'agit d'une simple application de la proposition précédente dans le cas particulier où p est un mot de Nyldon.

$\Rightarrow$ Supposons maintenant que w est un mot de Nyldon. Si p est un mot de Nyldon, alors on doit avoir $s <_{\text{lex}} p$ car, dans le cas contraire, (p, s) serait une factorisation de w de longueur au moins 2 en mots de Nyldon croissants.
Dès lors, il suffit de montrer que p est un mot de Nyldon. Supposons par l'absurde que p n'est pas un mot de Nyldon. Alors, il existe une factorisation $(p_1, \dots, p_k)$ de p de longueur au moins 2 en mots de Nyldon croissants. Comme w est un mot de Nyldon, alors on doit avoir $s <_{\text{lex}} p_k$ car, dans le cas contraire, $(p_1, \dots, p_k, s)$ serait une factorisation de longueur au moins 2 de w en mots de Nyldon croissants. Remarquons également que s est le plus long suffixe propre de $p_k s$ tel que s est un mot de Nyldon. Dès lors, par la proposition précédente, on obtient que $p_k s$ est un mot de Nyldon. Dès lors, la factorisation en mots de Nyldon croissants de w serait $(p_1, \dots, p_k s)$ qui est de longueur $k \geq 2$ ce qui contredirait le fait que w est un mot de Nyldon. Donc p est bien un mot de Nyldon.

□

Grâce à ces deux résultats, nous pouvons définir la factorisation standard en mots de Nyldon.

Définition 3.20. Soit w un mot de Nyldon sur un alphabet $\mathcal{A}$ et soit s le plus long suffixe propre de w tel que s est un mot de Nyldon. Alors, la factorisation standard de w est donnée par la paire (p, s) avec p est tel que $w = ps$.

Remarque 3.21. Il peut exister plusieurs factorisations (u, v) d'un mot de Nyldon telles que $u, v \in \mathcal{N}$ et $v <_{\text{lex}} u$. Prenons par exemple le mot $w = 1011101$. Nous avons les deux factorisations $(1011, 101)$ et $(1011101, 1)$ qui respectent les deux conditions énoncées. Cependant, la première est la factorisation standard de w. Dès lors, la factorisation standard est une factorisation particulière qui respectent les deux conditions précédentes.

En ce qui concerne les factorisations, une différence entre la factorisation en mots de Nyldon croissants et celle en mots de Lyndon décroissants est notable. Pour rappel, dans le premier chapitre, les mots de Lyndon ont été discutés et, grâce à la proposition 2.15, les mots de Lyndon sont soit des lettres soit de la forme uv avec $u <_{\text{lex}} v$. Il s'agit même d'une caractérisation des mots de Lyndon. De manière similaire, grâce au théorème 3.19, il est possible de trouver une factorisation (u, v) telle que $v <_{\text{lex}} u$. Cependant, ce n'est pas parce qu'il existe une factorisation (u, v) d'un mot w telle que $v <_{\text{lex}} u$ que cela fait de w un mot de Nyldon. En effet, en prenant le mot $w = 1001010010$, on remarque que la factorisation $(u, v) = (10010100, 10)$ respecte la propriété $v <_{\text{lex}} u$ mais w n'est pas un mot de Nyldon car une factorisation en mots de Nyldon croissants $(10010, 10010)$. Le théorème 3.19 précise les conditions à imposer sur la factorisation d'un mot w pour que w soit un mot de Nyldon.

3.4.4 Primitivité

Un autre point commun entre les deux familles de mots est que chaque élément est primitif. En effet, il est possible de démontrer cet aspect primitif grâce à un théorème que l'on doit à Schutzenberger.

Théorème 3.22. *Soit F un sous-emsemble de $\mathcal{A}^*$ muni d'une relation d'ordre total $\prec$. Alors, deux des trois conditions ci-dessous implique la troisième.*

- *(i) Tout mot de $\mathcal{A}^*$ admet au moins une $F-$factorisation.*
- *(ii) Tout mot de $\mathcal{A}^*$ admet au plus une $F-$factorisation.*
- *(iii) Tous les éléments de F sont primitifs et chaque classe de conjugaison de $\mathcal{A}^+$ contient exactement un élément de G.*

En effet, ce théorème permet de conclure que les mots de Nyldon sont primitifs.

Théorème 3.23. *Tout mot de Nyldon est primitif et tout mot primitif admet exactement un mot de Nyldon dans sa classe de conjugaison.*

Démonstration. En effet, les deux premiers points du théorème 3.22 sont vérifiés par les propositions 3.3 et 3.11 si on pose $G = \mathcal{N}$ et $\prec$ qui est $>_{lex}$. Dès lors, tous les mots de Nyldon sont primitifs et tout mot primitif admet exactement un mot de Nyldon dans sa classe de conjugaison par le théorème 3.22. □

En ce ce qui concerne les mots de Lyndon, on pourrait simplement dire que c'est la définition des mots de Lyndon. Cependant, si les mots de Lyndon sont définis via une des définitions alternatives (cf. définition 2.1), il est tout de même possible de récupérer l'aspect primitif via le théorème 3.22.

3.4.5 Codes et comma-free codes

Pour conclure cette brève comparaison, les familles des mots de Lyndon et de Nyldon peuvent être associées aux notions de circular code et comma-free code. Ces deux notions sont définies de la manière suivante :

Définition 3.24. (Code, circular code et comma-free code)
Soit $\mathcal{A}$ un alphabet.

- Un sous-ensemble F de $\mathcal{A}^*$ est un **code** si, pour tout $x_1, \ldots, x_m$, $y_1, \ldots, y_m$ dans F, on a $x_1 \cdots x_m = y_1 \cdots y_n$ si et seulement si $m = n$ et $x_i = y_i$ pour tout $i \in \{1, \ldots, m\}$.
- Un code F est un **circular code** si, pour tout u, $v \in \mathcal{A}^*$, on a $uv, vu \in F^* \Rightarrow u, v \in F^*$.
- Un code F est un **comma-free code** si, pour tout $w \in F^+$ et $u, v \in \mathcal{A}^*$, on a $uwv \in F^* \Rightarrow u, v \in F^*$.

Dans [15], Berstel, Perrin et Reutenauer ont montré que la famille des mots de Lyndon de longueur n sur un alphabet de taille k forme comma-free code pour certain n et k.

Proposition 3.25. *Soient $\mathcal{A}$ un alphabet de taille k et $n \geq 1$.*
L'ensemble des mots de Lyndon de longueur n est un comma-free code si et seulement si $n = 1$, ou $n = 2$ et $k \in \{2, 3\}$, ou $n \in \{3, 4\}$ et $k = 2$.

De même, il est possible de montrer que l'ensemble des mots de Nyldon forme un comma-free code pour certaines longueurs n de mots et certaines cardinalités d'alphabet k.

Lemme 3.26. *L'ensemble des mots de Nyldon de longueur 1, noté $\mathcal{N} \cap \mathcal{A}$, est un comma-free code pour tout alphabet $\mathcal{A}$.*

Démonstration. Il est trivial de montrer que $\mathcal{N} \cap \mathcal{A}$ est un code. De plus, on a $\mathcal{N} \cap \mathcal{A} = \mathcal{A}$. Dès lors, on a bien que $\mathcal{N} \cap \mathcal{A}$ est un comma-free code par définition. □

Lemme 3.27. *Soit $\mathcal{A}$ un alphabet de taille k. Alors $\mathcal{N} \cap \mathcal{A}^2$ est un comma-free code si et seulement si $k \in \{2, 3\}$.*

Démonstration. Si $k = 2$, alors $\mathcal{A} = \{0, 1\}$ et $\mathcal{N} \cap \mathcal{A}^2 = \{10\}$. Prouvons que $\mathcal{N} \cap \mathcal{A}^2$ est un *comma-free code*. Soit $u, v \in \mathcal{A}^*$ et $w \in (\mathcal{N} \cap \mathcal{A}^2)^+$ tels que $uwv \in (\mathcal{N} \cap \mathcal{A}^2)^*$. Montrons, par induction sur $|u|$, que $u, v \in (\mathcal{N} \cap \mathcal{A}^2)^*$. Si $|u| = 0$, alors, par hypothèse, $wv \in (\mathcal{N} \cap \mathcal{A}^2)^*$. Donc, v doit appartenir à $(\mathcal{N} \cap \mathcal{A}^2)^*$ car, sinon, $wv \notin (\mathcal{N} \cap \mathcal{A}^2)^*$. Maintenant, supposons que le résultat est vrai pour tout x tel que $|x| < |u|$ et montrons-le pour u. D'une part, on a trivialement u doit être de longueur paire pour que le résultat soit vrai. D'autre part, notons $u = abx$ avec $a, b \in \mathcal{A}$ et $x \in \mathcal{A}^*$. Donc, si $uwv \in (\mathcal{N} \cap \mathcal{A}^2)^*$, alors $a = 1$ et $b = 0$. Dès lors, par hypothèse de récurrence, on a $xwv \in (\mathcal{N} \cap \mathcal{A}^2)^* \Rightarrow x, v \in (\mathcal{N} \cap \mathcal{A}^2)^*$. Donc $u = 10x \in (\mathcal{N} \cap \mathcal{A}^2)^*$ et $v \in (\mathcal{N} \cap \mathcal{A}^2)^*$. D'où $\mathcal{N} \cap \mathcal{A}^2$ est un *comma-free code* pour $k = 2$. Si $k = 3$, alors $\mathcal{A} = \{0, 1, 2\}$ et $\mathcal{N} \cap \mathcal{A}^2 = \{10, 20, 21\}$. Soit $x \in (\mathcal{N} \cap \mathcal{A}^2)^+$ et $u, v \in \mathcal{A}^*$ tels que $uxv \in (\mathcal{N} \cap \mathcal{A}^2)^*$. Dès lors, il existe $l \geq 1$ et $y_1, \ldots, y_l \in \mathcal{N} \cap \mathcal{A}^2$ tels que $uxv = y_1 \cdots y_l$. Comme tous les mots de $\mathcal{N} \cap \mathcal{A}^2$ sont de longueur 2, pour montrer que $u, v \in (\mathcal{N} \cap \mathcal{A}^2)^*$, il suffit de montrer que x ne commence pas dans un

facteur y_i. Procédons par l'absurde et supposons qu'il existe $i \in \{1, \ldots, l\}$ tel que x commence dans y_i. Si $y_i \in \{10, 20\}$, alors x commence par un 0 et $x \notin (\mathcal{N} \cap \mathcal{A}^2)^+$. Si $y_i = 21$, alors x commence par un 1 et y_{i+1} devrait commencer par un 0 ce qui est impossible. Donc, x ne commence pas dans un facteur y_i pour $i \in \{1, \ldots, l\}$ et $\mathcal{N} \cap \mathcal{A}^2$ est un *comma-free code* pour $k = 3$ via une preuve similaire au cas $k = 2$. Enfin, prenons $k \geq 4$. On a donc $\{0, 1, 2, 3\} \subseteq \mathcal{A}$ et les mots $10, 21, 32$ appartiennent à $\mathcal{N} \cap \mathcal{A}^2$. Prenons $x = 21, u = 3$ et $v = 0$. Donc, $uxv = 3(21)0 = (32)(20) \in \mathcal{N}_2^*$ où $\mathcal{N}_2^*$ désigne l'ensemble des mots construits à partir des mots de Nyldon de longueur 2. Cependant, $u, v \notin (\mathcal{N} \cap \mathcal{A}^2)^*$. Donc, $\mathcal{N} \cap \mathcal{A}^2$ n'est pas un *comma-free code* si $k \geq 4$. □

Lemme 3.28. *Soit $\mathcal{A}$ un alphabet de taille k et $n \in \{3, 4, 5, 6\}$.*
Alors $\mathcal{N} \cap \mathcal{A}^n$ est un comma-free code si et seulement si $k = 2$.

Démonstration. Procédons en deux temps. Tout d'abord, prenons $k = 2$. Donc, $\mathcal{A} = \{0, 1\}$ et vérifions que $\mathcal{N} \cap \mathcal{A}^n$ est un *comma-free code* pour $n \in \{3, 4, 5, 6\}$. Démontrons le cas $n = 3$. On a $\mathcal{N} \cap \mathcal{A}^3 = \{100, 101\}$. Prenons $x \in (\mathcal{N} \cap \mathcal{A}^3)^+$, $u, v \in \mathcal{A}^*$ tels que $uxv \in (\mathcal{N} \cap \mathcal{A}^3)^*$. Dès lors, il existe $l \geq 1$ et $y_1, \ldots, y_l \in \mathcal{N} \cap \mathcal{A}^3$ tels que $uxv = y_1 \cdots y_l$. Comme tous les mots de $\mathcal{N} \cap \mathcal{A}^3$ sont de longueur 3, il suffit de montrer que x ne commence pas dans un facteur y_i pour conclure le cas $n = 3$. Procédons par l'absurde et supposons que x commence dans un facteur y_i pour un $i \in \{1, \ldots, l\}$. Si $y_i = 100$, cela veut dire que x commence par un 0 et $x \notin (\mathcal{N} \cap \mathcal{A}^3)^+$. Si $y_i = 101$, alors commence soit par 1 soit par 0. D'une part, si x commence par 0, on se ramène au cas où $y_i = 100$. D'autre part, si x commence par un 1, alors y_{i+1} commence par un 0 ce qui est impossible. D'où $\mathcal{N} \cap \mathcal{A}^3$ est un *comma-free code*. Les cas où $n \in \{4, 5, 6\}$ se traite de la même manière. Ensuite, montrons que $\mathcal{N} \cap \mathcal{A}^n$ n'est pas un *comma-free code* pour $k \geq 3$ et $n \in \{3, 4, 5, 6\}$. Soit $k \geq 3$. On a $\{0, 1, 2\} \subseteq \mathcal{A}$. Soit $n = 3$. Comme les mots 101 et 210 appartiennent à $\mathcal{N} \cap \mathcal{A}^3$ et comme $2(101)01 = (210)(101)$, on obtient que $\mathcal{N} \cap \mathcal{A}^3$ n'est pas un *comma-free code*. De même, comme

$$1000, 1001, 2100 \in \mathcal{N} \cap \mathcal{A}^4 \text{ et } 2(1001)000 = (2100)(1000),$$
$$10000, 10001, 21000 \in \mathcal{N} \cap \mathcal{A}^5 \text{ et } 2(10001)0000 = (21000)(10000),$$
$$100000, 100001, 210000 \in \mathcal{N} \cap \mathcal{A}^6 \text{ et } 2(100001)00000 = (210000)(100000),$$

on a que les codes $\mathcal{N} \cap \mathcal{A}^n$ ne sont pas des *comma-free codes* pour $n \in \{4, 5, 6\}$. □

Lemme 3.29. *Si $n \geq 7$, alors $\mathcal{N} \cap \mathcal{A}^n$ n'est un* comma-free code *pour aucun alphabet $\mathcal{A}$ de taille plus grande ou égale à 2.*

Démonstration. Supposons que $n \geq 7$ et que $\mathcal{A}$ contienne $\{0, 1\}$. On peut montrer trivialement que 10110^{n-4}, 1001010^{n-6} et $10^{n-4}100$ appartiennent à $\mathcal{N} \cap \mathcal{A}^n$.
Comme $101(10^{n-4}100)1010^{n-6} = (10110^{n-4})(1001010^{n-6})$, on a directement que $\mathcal{N} \cap \mathcal{A}^n$ n'est pas un *comma-free code* car 101 n'appartient pas à $\mathcal{N} \cap \mathcal{A}^n$. □

Le théorème suivant donne une caractérisation de l'ensemble des mots de Nyldon de longueur n qui sont des comma-free codes.

Théorème 3.30. *Soit $\mathcal{A}$ un alphabet de taille k et $n \geq 1$.*
L'ensemble $\mathcal{N} \cap \mathcal{A}^n$ des mots de Nyldon de longueur n sur $\mathcal{A}$ est un comma-free code si et seulement si $n = 1$, ou $n = 2$ et $k \in \{2, 3\}$, ou $n \in \{3, 4, 5, 6\}$ et $k = 2$.

Démonstration. Cela résulte des lemmes 3.26, 3.27, 3.28 et 3.29. □

Chapitre 4

Ensembles de Hall et de Lazard

Il est possible de lier l'ensemble des mots de Nyldon avec les ensembles de Lazard et de Hall. Le but de ce chapitre est triple. Tout d'abord, il s'agira de présenter les ensembles de Lazard et de Hall ainsi que d'exhiber plusieurs de leurs propriétés. Ensuite, une deuxième partie de ce chapitre sera consacrée au lien unissant les ensembles de Lazard et de Hall. Enfin, le dernier travail de celui-ci consistera à démontrer que la famille des mots de Nyldon est un ensemble de Lazard à droite.

4.1 Ensembles de Hall

Dans cette partie, toutes les définitions sont tirées de l'article [13] de Mélançon. Il est possible de prendre une autre définition des ensembles de Hall (cf. [18]) ce qui entrainera divers changements dans les résultats qui seront présentés ici. Ce changement sera discuté plus tard dans ce chapitre.

4.1.1 Introduction aux ensembles de Hall

Avant de donner directement la définition des ensembles de Hall, il semble opportun de définir plusieurs notions utilisées notamment en théorie des graphes.

Définition 4.1. Soit $\mathcal{A}$ un alphabet :

1. Notons $M(\mathcal{A})$ l'ensemble des arbres binaires dont les feuilles sont étiquetées par des éléments de $\mathcal{A}$. Il s'agit du magma libre sur $\mathcal{A}$.
2. Le degré d'un arbre binaire est le nombre de feuilles qu'il possède. On notera ce degré $|t|$.
3. Un arbre de degré 1 sera noté via une lettre de $\mathcal{A}$.
4. Si un arbre t est tel que $|t| \geq 2$, alors on peut écrire $t = [L(t), R(t)]$ où $L(t)$ (resp. $R(t)$) est le sous-arbre direct à gauche (resp. sous-arbre direct à droite).
5. Un sous-arbre à droite d'un arbre t est soit $R(t)$ ou un sous-arbre à droite de $L(t)$ ou de $R(t)$.
6. Un sous-arbre à gauche d'un arbre t est soit $L(t)$ ou un sous-arbre à gauche de $L(t)$ ou de $R(t)$.

Exemple 4.2. Prenons, par exemple, l'arbre binaire $t = [[a, [b, c]], d]$ avec $a, b, c, d \in \mathcal{A}^*$. Cet arbre se représente de la manière suivante :

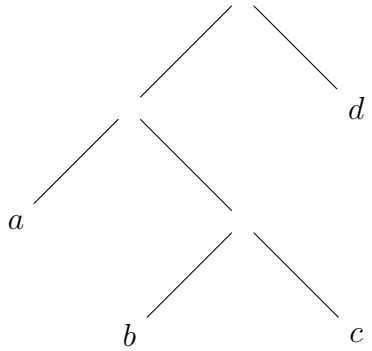


FIGURE 4.1 – Représentation de $t = [[a, [b, c]], d]$.

Dans ce cas-ci, on a un sous-arbre direct à droite de t est $R(t) = d$. On peut remarquer également qu'il s'agit d'un sous-arbre extrême à droite. De plus, $[b, c]$ est un autre sous-arbre à droite de t mais il n'est pas sous-arbre extrême à droite de t.

En considérant le monoïde $\mathcal{A}^*$ muni de l'opération de concaténation $\cdot$, il est possible définir une application canonique f entre $M(\mathcal{A})$ et $\mathcal{A}^*$ comme suit : $f(a) = a$ si $a \in \mathcal{A}$ et $f(t) = f(L(t)) \cdot f(R(t))$ si $t = [L(t), R(t)]$ et $|t| \geq 2$. Dès lors, $f(t)$ est appelé le feuillage de t. La longueur de $f(t)$ est spécifié par la notation usuelle, à savoir $|f(t)|$. On peut remarquer que $|f(t)|$ est aussi le degré $|t|$ de l'arbre t. Grâce à ces différents notions, il est maintenant possible de donner la définition des ensembles de Hall.

Définition 4.3. (Ensemble de Hall)
Soit H un sous-ensemble de $M(\mathcal{A})$ muni d'un ordre total $\leq_H$ satisfaisant

$$t <_{\mathrm{H}} R(t), \text{ pour tout arbre } t \text{ de degré au moins 2.} \qquad (*)$$

Ce sous-ensemble H est appelé **ensemble de Hall** si $\mathcal{A} \subset H$ et si, pour tout arbre $h = [L(h), R(h)]$ de degré au moins 2, $h \in H$ si et seulement si les deux conditions suivantes sont satisfaites :

(1) $L(h), R(h)$ sont des éléments de H et $L(h) <_{\mathrm{H}} R(h)$,

(2) soit $L(h) \in \mathcal{A}$ soit $R(L(h)) \geq_H R(h)$.

Les éléments d'un ensemble de Hall H sont appelés arbres de Hall.

Remarque 4.4. Dans cette définition, l'ordre total $\leq_H$ n'est pas unique. En effet, la définition impose simplement une condition sur l'ordre choisi. Lors de la construction d'un ensemble de Hall, certains choix d'ordre devront être fait et cela se reflétera sur les éléments de cet ensemble. Cependant, une fois que l'ordre choisi respecte la condition $(*)$ et est fixé, la définition 4.3 devient une définition récursive. L'exemple suivant permet de montrer que le choix d'un ordre influence l'ensemble de Hall construit.

Exemple 4.5. Tout d'abord, considérons un alphabet binaire $\mathcal{A} = \{a, b\}$. Par la définition 4.3, tout élément de $\mathcal{A}$ est un arbre de Hall. Un premier choix va porter sur l'ordre des éléments de $\mathcal{A}$. En effet, si $a < b$, alors l'arbre 4.2 est un arbre de Hall.

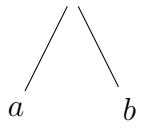


FIGURE 4.2 – Arbre de Hall avec $a < b$

Cependant, si $a > b$, alors l'arbre de la figure 4.2 n'est pas un arbre de Hall mais celui de la figure 4.3 l'est.

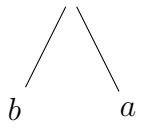
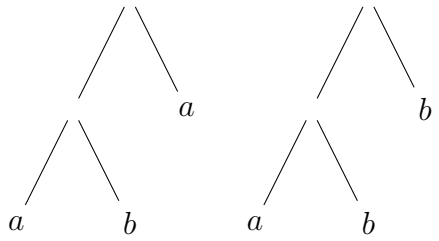


FIGURE 4.3 – Arbre de Hall avec $b < a$

Considérons maintenant que l'ordre choisi sur les lettres est $a < b$ et que l'ordre sur H, noté $\leq_H$, restreint à $\mathcal{A}$ est $\leq$. Une remarque utile pour la suite est le fait que l'arbre de la figure 4.2 est le seul arbre de Hall de degré 2. En effet, les arbres $[a, a]$, $[b, b]$ et $[b, a]$ ne respectent pas la définition 4.3 à cause de l'ordre choisi sur les lettres et ne sont donc pas des arbres de Hall. Maintenant, un deuxième choix concernant l'ordre se pose afin de construire les arbres de degré 3. En effet, les arbres a, b et $[a, b]$ doivent être ordonnés afin de construire les arbres de degré 3. Par exemple, si $[a, b] <_H a <_H b$, alors les arbres de degré 3 sont représentés dans la figure 4.4



(a) Arbre 1 (b) Arbre 2

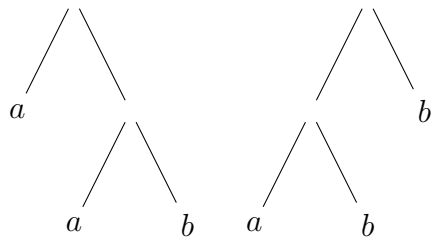
FIGURE 4.4 – Arbre de Hall de degré 3 avec $[a, b] <_H a <_H b$

Cependant, si l'ordre était $a <_H [a, b] <_H b$, alors la figure 4.5 représente les arbres de Hall de degré 3.

Enfin, si l'ordre choisi est tel que $a <_H b <_H [a, b]$, alors les arbres de degré 3 sont représentés par la figure 4.6.

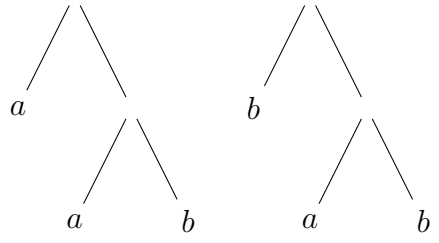
Ces différents cas montrent évidemment que l'ordre choisi sur le sous-ensemble H va influencer ses éléments. Dès lors, l'ensemble de Hall H n'est pas unique mais bien déterminé par un ordre total respectant la condition $(*)$ de la définition 4.3.

Remarque 4.6. Dans la suite, on considéra un ordre total $\leq_H$ qui respecte la condition $(*)$ de la définition. Aucune autre restriction ne sera faite pour que les résultats ci-après soient corrects.



(a) Arbre 1 (b) Arbre 2

FIGURE 4.5 – Arbre de Hall de degré 3 avec $a <_H [a, b] <_H b$



(a) Arbre 1 (b) Arbre 2

FIGURE 4.6 – Arbre de Hall de degré 3 avec $a <_H b <_H [a, b]$

Dans la définition 4.3, l'ordre total $\leq_H$ permet de comparer un arbre h avec son sous-arbre direct à droite $R(h)$. Cependant, il est possible d'aller encore plus loin dans ce sens en comparant un arbre h avec un de ses sous-arbres à droite.

Lemme 4.7. *Soit $h = [L(h), R(h)]$ un arbre de Hall. Si h' est un sous-arbre à droite de h, alors $h' \geq_H R(h)$.*

Démonstration. Par définition, si h' est un sous-arbre à droite de h, alors $h' = R(h)$ ou h' est un sous-arbre à droite de $L(h)$ ou h' est un sous-arbre à droite de $R(h)$. Procédons maintenant par récurrence sur le degré de h. Si $|h| = 2$, alors le résultat est trivial car $h' = R(h)$ donc $h' \geq_H R(h)$. Supposons que le résultat est vérifié pour tous les arbres de Hall de degré $< |h|$ et montrons-le pour les arbres de Hall de degré $|h|$.

- Si $h' = R(h)$, alors $h' \geq_H R(h)$.
- Si h' est un sous-arbre à droite de $L(h)$, alors, par induction, $h' \geq_H R(L(h))$ et, par la définition 4.3, on a $R(L(h)) \geq_H R(h)$. Donc, $h' \geq_H R(h)$.
- Si h' est un sous-arbre à droite de $R(h)$, alors, par induction, $h' \geq_H R(R(h))$. Par la définition 4.3, on a également $R(R(h)) \geq_H R(h)$. Au final, $h' \geq_H R(h)$.

□

La définition 4.1 permet d'introduire la notion de feuillage. Il parait évident qu'à un arbre t de $M(\mathcal{A})$ donné corresponde un unique feuillage $f(t)$. Cependant, une question légitime est de savoir si la réciproque est vraie. En théorie des graphes, en considérant un feuillage $f(t)$, il est possible de construire plusieurs arbres ayant ce feuillage. Cependant, en se plaçant dans le cas où les arbres construits doivent

appartenir à H, le résultat devient vrai. Le lemme suivant permet d'avancer vers le résultat voulu.

Lemme 4.8. *Soit h un arbre de Hall et $w = f(h)$ son feuillage. Supposons que w puisse se factoriser en $w = uv$ avec $u, v \neq \epsilon$. Alors, il existe des arbres de Hall $k_1, \ldots, k_m, h_1, \ldots, h_n$ tels que*

$$u = f(k_1) \cdots f(k_m), \quad v = f(h_1) \cdots f(h_n),$$

et $k_1, \ldots, k_m <_{\mathrm{H}} h_1$, $h_1 \geq_H \ldots \geq_H h_n \geq_H R(h)$.

Démonstration. Prouvons ce lemme en imposant, en plus, que les $h_1, \ldots, h_n$ sont des sous-arbres à droite de h. Soit $h = [L(h), R(h)]$. Par définition du feuillage de h, on a que $f(h) = f(L(h))f(R(h))$. De plus, par hypothèse, on a $f(h) = w = uv$. Trois cas sont, dès lors, possibles. Dans le premier cas, on suppose que $u = f(L(h))$ et $v = f(R(h))$. Si tel est le cas, alors on a $m = n = 1$ et $k_1 = L(h)$ et $h_1 = R(h)$. De plus, par (1) de la définition 4.3, on a $k_1 <_{\mathrm{H}} h_1$. Les deux autres cas se prouve par récurrence sur le degré de h. Les cas de base sont triviaux. Dans le deuxième cas, on suppose que $f(L(h)) = ux$ et $v = xf(R(h))$ avec $x \neq \epsilon$. Ainsi, par hypothèse de récurrence sur $L(h)$, il existe des arbres de Hall $k_1, \ldots, k_m,\ h_1, \ldots, h_{n-1}$ tels que

$$u = f(k_1) \cdots f(k_m), \quad x = f(h_1) \cdots f(h_{n-1}),$$

et $k_1, \ldots, k_m <_{\mathrm{H}} h_1,\ h_1 \geq_H \ldots \geq_H h_{n-1} \geq_H R(L(h))$ où $h_1, \ldots, h_{n-1}$ sont des sous-arbres à droite de $L(h)$. Comme $L(h)$ est un sous-arbre direct à gauche de h, $h_1, \ldots, h_{n-1}$ sont des sous-arbres à droite de h. De plus, on a $R(L(h)) \geq_H R(h)$ par (2) de la définition 4.3. Si on prend $h_n = R(h)$, on obtient

$$v = xf(R(h)) = f(h_1) \cdots f(h_{n-1})f(h_n)$$

ce qui conclut ce cas. Enfin, dans le troisième cas, on suppose maintenant $u = f(L(h))x$ et $f(R(h)) = xv$ avec $x \neq \epsilon$. Dès lors, par hypothèse de récurrence sur $R(h)$, il existe des arbres de Hall $k_2, \ldots, k_m,\ h_1, \ldots, h_n$ tels que

$$x = f(k_2) \cdots f(k_m), \quad v = f(h_1) \cdots f(h_n),$$

et $k_2, \ldots, k_m <_{\mathrm{H}} h_1$, $h_1 \geq_H \ldots \geq_H h_n \geq_H R(R(h))$ où $h_1, \ldots, h_n$ sont des sous-arbres à droite de $R(h)$. Comme $R(h)$ est un sous-arbre à droite de h, $h_1, \ldots, h_n$ sont des sous-arbres à droite de h. De plus, par la définition 4.3, on a $R(R(h)) >_H R(h)$ et, par (2) de cette même définition, on a $L(h) <_{\mathrm{H}} R(h)$. En utilisant l'aspect transitif de l'ordre $\leq_H$, on a $L(h) <_{\mathrm{H}} R(h) <_{\mathrm{H}} R(R(h)) \leq_H h_1$. Donc, en prenant $k_1 = L(h)$, on obtient

$$u = f(L(h))x = f(k_1) \cdots f(k_m) \text{ et } v = f(h_1) \cdots f(h_n)$$

□

Remarque 4.9. Ce lemme permet de trouver la factorisation du suffixe v de w en suivant chaque étape de l'induction de la démonstration. En effet, si le cas 3 est rencontré, cela veut dire que $v = f(h_1) \cdots f(h_n)$ avec $h_1 \geq \ldots \geq h_n \geq R(R(h))$ car v est un suffixe de $R(h)$. Si le cas 2 est rencontré, v n'est pas le feuillage du sous-arbre direct à droite de h, à savoir $R(h)$. En d'autres mots, si v n'est pas le sous-arbre direct à droite de h, alors il y a au moins deux facteurs dans la factorisation de v.

De plus, comme discuté ci-dessus, la démonstration de ce lemme 4.8 donne une méthode pour trouver la factorisation en arbres de Hall de tout mot u et v qui sont les facteurs du feuillage d'un arbre de Hall h ($uv = w = f(h)$). Soient h un arbre de Hall tel que $w = f(h)$ et uv une factorisation de w telle que $u \neq \epsilon \neq v$. La méthode est la suivante :

1. Suivre les branches de l'arbre jusqu'à arriver à une feuille étiquetée avec la première lettre de v,
2. Remonter en sens inverse jusqu'à un noeud,
3. Si, lors de l'arrivée à un noeud, il reste un sous-arbre à droite non visité par le chemin de départ, poser ce sous-arbre comme un h_i,
4. Si, lors de l'arrivée à un noeud, il reste un sous-arbre à gauche non visité par le chemin de départ, poser ce sous-arbre comme un k_j.
5. Itérer le procédé décrit décrit par les points 2, 3 et 4 jusqu'à arriver à la racine de l'arbre de Hall h.
6. Lorsque le chemin inverse reviendra à la racine de h, ordonner les k_j et les h_i pour avoir les factorisations de u et v.

Il est évident que la factorisation dépendra des facteurs u et v choisis.

Exemple 4.10. Tout d'abord, soit un alphabet $\mathcal{A} = \{a, b\}$ avec $a < b$. Ensuite, l'ordre $\leq_H$ qui va être considéré sera tel que

$$a <_H [a, b] <_H [[a, b], b] <_H [[[a, b], b], b] <_H [[[[a, b], b], b], b] <_H b.$$

Dès lors, l'arbre h de la figure 4.7 est un arbre de Hall.

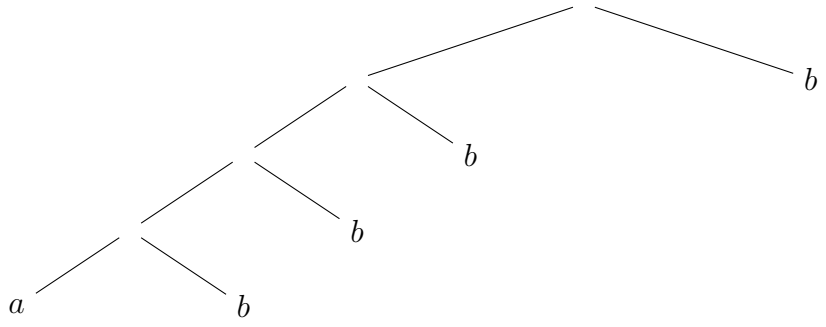


FIGURE 4.7 – Arbre de Hall h

Son feuillage est le mot $f(h) = w = abbbb$. Considérons une première factorisation uv de w telle que $u = ab, v = bbb$. En suivant la méthode décrite ci-dessus, on obtient les sous-arbres k_j et h_i suivants : $k_1 = [a, b]$ et $h_1 = h_2 = h_3 = b$. De plus, l'arbre obtenu à partir de cette factorisation est celui de la figure 4.8 (si f_1 et f_2 sont deux feuilles de même couleur, alors ils appartiennent au même arbre).

On peut aussi remarquer que k_1, h_1, h_2 et h_3 respectent les conditions du lemme, à savoir : $k_1 <_{\mathrm{H}} h_1$ et $h_1 \geq_H h_2 \geq_H h_3 \geq_H R(h) = b$.

Considérons, cette fois-ci, une seconde factorisation uv de w telle que $u = abb, v = bb$. En suivant la même méthode ci-dessus, on obtient les sous-arbres k_j et h_i suivants : $k_1 = [[a, b], b]$ et $h_1 = h_2 = b$. De plus, L'arbre obtenu à partir de cette

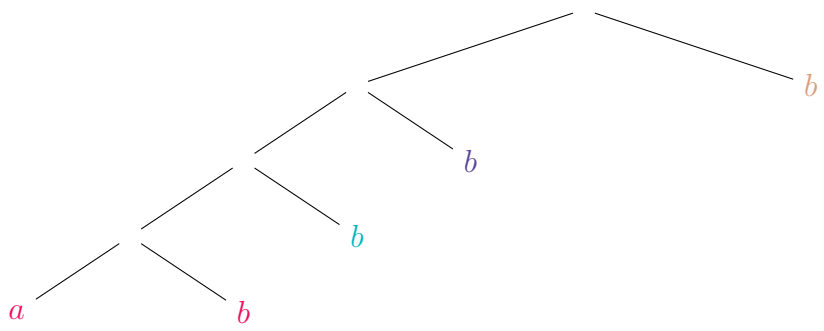


FIGURE 4.8 – Première factorisation de h

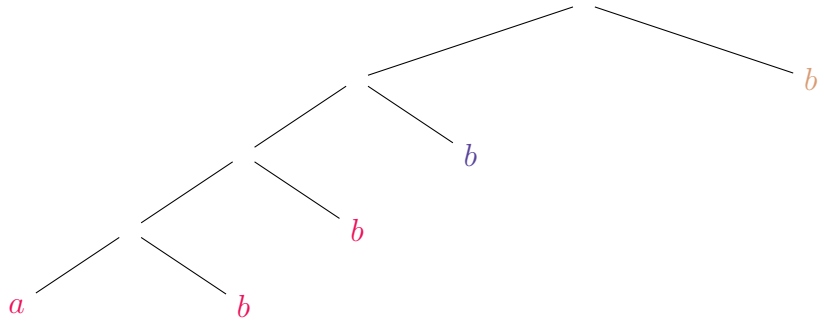


FIGURE 4.9 – Seconde factorisation de h

factorisation est donc celui de la figure 4.9(en utilisant le même système de coloration que la factorisation précédente)

D'autre part, les sous-arbres k_1, h_1 et h_2 respectent également les conditions du lemme car $k_1 <_{\mathrm{H}} h_1$ et $h_1 \geq_H h_2 \geq R(h) = b$.

4.1.2 Mots de Hall

Enfin, pour terminer cette section, il semble intéressant de se pencher un peu plus sur les feuillages de ces arbres de $M(\mathcal{A})$. En effet, grâce à ce lien entre feuillage d'un élément de $M(\mathcal{A})$ et arbre de Hall, il est possible de définir les mots de Hall.

Définition 4.11. (Mots de Hall)
Soient $\mathcal{A}$ un alphabet et H l'ensemble de Hall construit à partir de $\mathcal{A}$. On dit que $w \in \mathcal{A}^*$ est un mot de Hall s'il existe un arbre de Hall h tel que $w = f(h)$.

Il est désormais possible de montrer la véracité de la question de départ à savoir si un mot de Hall est le feuillage d'un unique arbre de Hall.

Théorème 4.12. *Tout mot de Hall w est le feuillage d'un unique arbre de Hall h par l'application f. Si un mot de Hall $w = f(h)$ peut être écrit comme un produit de mots de Hall :*

$$w = f(h_1) \cdots f(h_n) \qquad \text{avec } h_1 \geq_H \ldots \geq_H h_n,$$

alors $n = 1$.

Démonstration. Procédons par récurrence sur le degré d des arbres. On a trivialement le cas de base $d = 1$. Supposons maintenant que le résultat est vrai pour les arbres de degré d et soit $t = [L(t), R(t)]$ un arbre de Hall de degré $d+1$ tel que

$$f(t) = f(L(t))f(R(t)) = f(h_1)\cdots f(h_n), \quad \text{avec } h_1 \geq_H \ldots \geq_H h_n. \qquad (*)$$

Montrons que $n = 1$ et séparons la démonstration en trois cas qui dépendent de la taille de $R(t)$ et h_n.

Cas 1 : $|R(t)| = |h_n|$. Comme l'application f préserve les longueurs, on a, par $(*)$, que $f(L(t)) = f(h_1)\cdots f(h_{n-1})$ et $f(R(t)) = f(h_n)$. En utilisant l'hypothèse de récurrence sur la première égalité, on obtient que $h_1 = L(t)$ et $n - 1 = 1$ donc $n = 2$et, en utilisant l'hypothèse de récurrence sur la deuxième égalité, on obtient $h_n = R(t)$. Cependant, par (1) de la définition 4.3, on a $L(t) <_{\mathrm{H}} R(t)$ et $h_1 \geq_H h_2$ par hypothèse d'où une contradiction et l'impossibilité du cas 1.

Cas 2 : $|R(t)| > |h_n|$. Par $(*)$, comme f préserve la longueur, $f(R(t)) = vf(h_{i+1})\cdots f(h_n)$ où v est un suffixe de $f(h_i)$ pour un $i < n$. Tout d'abord, remarquons que v ne peut pas être le mot vide ϵ. Supposons, par l'absurde, que $v = \epsilon$ et donc que $f(R(t)) = f(h_{i+1})\cdots f(h_n)$. Par hypothèse de récurrence sur $R(t)$, on a que $f(R(t))$ se factorise en un facteur et donc $i + 1 = n$ ce qui contredit $|R(t)| > |h_n|$. Comme v est un suffixe propre de $f(h_i)$, on déduit, par le lemme 4.8, que $v = f(k_1)\cdots f(k_m)$ pour des arbres de Hall $k_1, \ldots, k_m$ avec $k_1 \geq_H \ldots \geq_H k_m \geq_H R(h_i)$. Grâce à $(*)$ de la définition 4.3 et au fait que $h_i \geq_H h_{i+1}$, on obtient que $k_1 \geq_H \ldots \geq_H k_m >_H h_{i+1} \geq_H \ldots \geq_H h_n$. Dès lors, on a

$$f(R(t)) = f(k_1)\cdots f(k_m)f(h_{i+1})\cdots f(h_n).$$

Cette factorisation contient au moins deux éléments car $i < n$ et $m \geq 1$. Cependant, cette factorisation de $R(t)$ est en contradiction avec l'hypothèse de récurrence sur $R(t)$. Donc, ce deuxième cas est également impossible.

Cas 3 : $|R(t)| < |h_n|$. Par $(*)$, on a $f(h_n) = vf(R(t))$ où v est un suffixe de $f(L(t))$. $|R(t)| < |h_n|$ force v à être différent de ϵ.
Si $n > 1$, alors v doit être un suffixe propre de $f(L(t))$ et, par le lemme 4.8, on obtient des arbres de Hall $k_1, \ldots, k_m$ tels que

$$v = f(k_1)\cdots f(k_m) \quad \text{avec } k_1 \geq_H \ldots \geq_H k_m \geq_H R(L(t)).$$

Comme $R(L(t)) \geq_H R(t)$ par (2) de la définition 4.3, on a que $f(R(h_n))$ se factorise en

$$f(h_n) = f(k_1)\cdots f(k_m)f(R(t)) \quad \text{avec } k_1 \geq_H \ldots \geq_H k_m \geq_H R(t) \qquad (**)$$

De plus, si $n > 1$, alors $h_n \in H_d$ où H_d est l'ensemble des éléments de H de degré d donc on peut appliquer l'hypothèse de récurrence à h_n. Cependant, on a une contradiction car il y a la présence d'au moins deux facteurs dans $(**)$.

Donc, l'unique possibilité est d'être dans le cas 3 avec $n = 1$.

Montrons maintenant que, si h et k sont deux arbres de Hall de degré $d+1$ avec $f(h) = f(k)$, alors $h = k$. Notons $h = [L(h), R(h)]$ et $k = [L(k), R(k)]$. Dès lors, on obtient $f(h) = f(L(h))f(R(h)) = f(L(k))f(R(k)) = f(k)$. On peut supposer $|R(h)| \geq |R(k)|$ (la démonstration est symétrique si l'on suppose l'inégalité inverse).

Si $|R(h)| = |R(k)|$, alors $f(R(h)) = f(R(k))$ et $f(L(h)) = f(L(k))$. Donc, par induction, on a $L(h) = L(k)$ et $R(h) = R(k)$ donc $h = k$.
Si $|R(h)| > |R(k)|$, alors $f(R(h)) = vf(R(k))$ où v est un suffixe propre de $L(k)$. Par le lemme 4.8, v se factorise en

$$v = f(r_1) \cdots f(r_m), \tag{4.1}$$

où $r_1, \ldots, r_m$ sont des arbres de Hall satisfaisants $r_1 \geq_H \ldots \geq_H r_m \geq_H R(L(k))$. Par (2) de la définition 4.3, $R(L(k)) \geq_H R(k)$ et on obtient

$$f(R(h)) = f(r_1) \cdots f(r_m) f(R(k)) \quad \text{avec } r_1 \geq_H \ldots \geq_H r_m \geq_H L(k) \tag{4.2}$$

Dans ce cas, on a au moins deux facteurs dans cette factorisation de $f(R(h))$.
Au final, on a une contradiction car, par hypothèse de récurrence, $f(R(h))$ ne peut pas être factorisé en un produit d'au moins deux feuillages d'arbres de Hall. □

Remarque 4.13. Dans la suite, H désignera l'ensemble des mots de Hall sur $\mathcal{A}^*$. Il s'agit d'un abus de notation qui est légitimé par le théorème 4.12. De plus, il est également possible de définir un ordre total $\leq$ sur H via l'ordre total $\leq_H$ de l'ensemble H défini par la définition 4.3.

Le théorème 4.12 permet d'identifier un mot de Hall à un unique arbre de Hall. Par la remarque 4.13, les propriétés de la définition 4.3 se transposent directement dans l'ensemble des mots de Hall. En effet, si h est un mot de Hall, alors

1. la condition "$t <_{\mathrm{H}} t'$, pour tout arbre $t = [L(t), t']$ de degré au moins 2" devient "$h < h'$ si h' est un suffixe propre de h".
2. si $h = h_1 h_2$, la condition "$h \in H$ (H = ensemble d'arbres de Hall) si et seulement si $L(h), R(h)$ sont des éléments de H et $L(h) <_H R(h)$" devient "$h \in H$ (H = ensemble des mots de Hall) si et seulement si h_1, h_2 sont des éléments de H et $h_1 < h_2$.
3. si $h = h_1 h_2$, la condition "$h \in H$ si et seulement si $L(h) \in \mathcal{A}$ soit $L(h) = [k_1, k_2]$ et $k_2 \geq_H R(h)$" devient "$h \in H$ si et seulement si $h_1 \in \mathcal{A}$ ou $h_1 = h_1' h_1''$ et $h_1'' \geq h_2$. "

Pour démontrer l'unicité de la factorisation en mots de Hall, il est nécessaire de parler de factorisation standard d'un mot de Hall. Si h est un mot de Hall, alors, par le théorème 4.12, il existe un unique arbre de Hall t tel que $f(t) = h$. Si $|h| \geq 2$, alors $t = [L(t), R(t)]$. On note $h_p = f(L(t))$ et $h_s = f(R(t))$. Dès lors, h peut s'écrire sous la forme $h_p h_s$ et cette factorisation est appelée factorisation standard de h. De plus, par définition de la factorisation standard de h, les mots h_p et h_s sont non vides. Dans la suite de ce chapitre, le premier facteur de la factorisation standard d'un mot de Hall sera noté à l'aide de l'indice p tandis que le second facteur sera noté à l'aide de l'indice s.

Proposition 4.14. *Si h est un mot de Hall et si $h = h_p h_s$ est sa factorisation standard, alors*

$$h \leq h_s \tag{C1}$$

$$h_p < h_s \tag{C2}$$

Démonstration. Direct via les définitions d'arbres de Hall et du théorème 4.12. □

Maintenant, en prenant un autre mot de Hall k tel que $h < k$, alors hk est un mot de Hall dont la factorisation standard est $(hk)_p = h$ et $(hk)_s = k$ si et seulement si

$$\text{soit } h \text{ est une lettre}, \quad \text{ou } h_s \geq k. \tag{C3}$$

Il est possible d'itérer ce raisonnement en augmentant la longueur du mot considéré. Dès lors, il est possible de considérer une suite standard.

Définition 4.15. Soit $s = (h_1, \ldots, h_n)$ avec $h_1, \ldots, h_n \in H$.
On dit que s est une **suite standard** si, pour tout $i \in \{1, \ldots, n\}$,

$$\text{soit } h_i \in \mathcal{A} \text{ ou } h_i = (h_i)_p(h_i)_s \text{ et } (h_i)_s \geq h_{i+1}, \ldots, h_n \tag{C4}$$

Exemple 4.16. Tout suite de lettre est une suite standard. De même, si $\mathcal{A} = \{0, \ldots, m\}$ avec $0 < 1 < \ldots < m$ et $s = (s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n)$ avec $s_1, \ldots, s_{i-1} \in \mathcal{A}$, $s_i = am$ où $a \in \mathcal{A}$ et $s_{i+1}, \ldots, s_n \in \mathcal{A}$, alors s est une suite standard. Par exemple, pour l'alphabet $\mathcal{A} = \{0, 1\}$, $s = (0, 1, 0, 0, 1, 01, 0, 1, 0, 1)$ est une suite standard car 01 appartient à H et $1 \geq 0$ et $1 \geq 1$.

Définition 4.17. (Accroissement et accroissement légal) Soient $h_1, \ldots, h_n$ des mots de Hall sur un alphabet $\mathcal{A}$ et $s = (h_1, \ldots, h_m)$ une suite de mots de Hall.

- Un accroissement de s est un couple de deux mots consécutifs dans s (h_i, h_{i+1}) tel que $h_i < h_{i+1}$
- Si s est une suite standard qui n'est pas décroissante, un accroissement légal de s est un accroissement (h_i, h_{i+1}) telle que

$$h_{i+1} \geq h_{i+2}, \ldots, h_n$$

Soit une suite s non décroissante et possédant un accroissement (h_i, h_{i+1}). Grâce à cet accroissement, il est possible de définir deux nouvelles suites à partir de s :

$$s' = (h_1, \ldots, h_{i-1}, h_i h_{i+1}, h_{i+2}, \ldots, h_n), \tag{C5}$$

$$s'' = (h_1, \ldots, h_{i-1}, h_{i+1}, h_i, h_{i+2}, \ldots, h_n) \tag{C6}$$

Donc, s' est obtenu à partir de s en concaténant les éléments h_i et h_{i+1} tandis que la suite s'' est obtenue, quant à elle, à partir de s en inversant les éléments h_i et h_{i+1}.

Grâce aux notions de suites standards et d' accroissement, le résultat suivant devient licite.

Proposition 4.18. *Soit $s = (h_1, \ldots, h_n)$ une suite standard telle que $h_1, \ldots, h_n \in H$. Si (h_i, h_{i+1}) est un accroissement de s, alors s' et s'' sont des suites standards. De plus, la factorisation standard de $h_i h_{i+1}$ est telle que $(h_i h_{i+1})_p = h_i$ et $(h_i h_{i+1})_s = h_{i+1}$.*

Démonstration. Découpons cette preuve en trois partie.

1. Montrons que, sous les hypothèses du résultat, $h_i h_{i+1}$ est un mot de Hall ce qui impliquera directement que $(h_i h_{i+1})_p = h_i$ et $(h_i h_{i+1})_s = h_{i+1}$ car $|h_i h_{i+1}| \geq 2$. Comme (h_i, h_{i+1}) est un accroissement, on a que $h_i < h_{i+1}$. De plus, h_i est soit une lettre de $\mathcal{A}$ soit $h_i = h_i' h_i''$ et donc $h_i'' \geq h_{i+1}$ par $C4$. Dès lors, (h_i, h_{i+1}) est un mot de Hall qui satisfait $(h_i h_{i+1})_p = h_i$ et $(h_i h_{i+1})_s = h_{i+1}$ par $C3$.

2. Montrons que s' est une suite standard. Pour ce faire, il faut que s' vérifie les conditions suivantes :

   (i) $h''_j \geq h_i h_{i+1}$ pour tout $j \in \{1, \ldots, i-1\}$,

   (ii) $h_{i+1} \geq h_{i+2}, \ldots, h_n$.

   Comme (h_i, h_{i+1}) est un accroissement de s, (ii) est immédiat. Considérons maintenant $h_j = h'_j h''_j$ la factorisation standard de h_j avec $j \in \{1, \ldots, i-1\}$. Par $C4$, on a $h''_j \geq h_{i+1}$ pour $j < i$ et, par $C1$, on a $h_{i+1} > h_i h_{i+1}$. Dès lors, $h''_j > h_i h_{i+1}$ et s' est une suite standard.

3. Montrons maintenant que s'' est une suite standard. Pour ce faire, il suffit de montrer que $h''_{i+1} \geq h_i$. On a $h_i < h_{i+1}$ car (h_i, h_{i+1}) est un accroissement de s et $h_{i+1} < h''_{i+1}$ par $C1$. Donc, $h''_{i+1} > h_i$ et s'' est une suite standard.

□

4.2 Factorisation en mots de Hall

Une des propriétés de la famille des mots de Hall est qu'elle forme une factorisation complète du monoïde libre $\mathcal{A}^*$.

Théorème 4.19. *(Factorisation en mots de Hall)*
Tout mot w de $\mathcal{A}^*$ peut être factorisé de manière unique en mots de Hall décroissants :

$$w = h_1 \cdots h_n \quad \text{avec } h_1 \geq \ldots \geq h_n.$$

L'existence d'une telle factorisation peut être montrée de manière assez classique en utilisant les suites standards. En ce qui concerne l'unicité, il existe deux manières pour la démontrer. La première se base sur un lemme et est assez classique tandis que la seconde repose sur les suites standards et, plus particulièrement, sur une relation binaire $\rightarrow$ définie sur les suites standards. Dans la suite de ce chapitre, les deux démonstrations vont être envisagées.

4.2.1 Méthode classique

Afin de démontrer le théorème 4.19, il est essentiel de remarquer le lemme suivant.

Lemme 4.20. *Soient $w = h_1 \cdots h_n$ avec $h_1 \geq \ldots \geq h_n$ une factorisation décroissante de w en mots de Hall et v un suffixe propre de h_i. Si $v = r_1 \cdots r_m$ est une factorisation de v définie par le lemme 4.8, alors $r_m > h_{i+1}$ et le mot $vh_{i+1} \cdots h_n$ peut s'écrire comme le produit décroissant de mots de Hall suivant :*

$$vh_{i+1} \cdots h_n = r_1 \cdots r_m h_{i+1} \cdots h_n.$$

Démonstration. Si v est le mot vide, alors le résultat est vrai. Si v est non vide, posons $v = r_1 \cdots r_m$ la factorisation défini par le lemme 4.8. Donc, $r_1, \ldots, r_m \in H$ et $r_1 \geq \ldots \geq r_m \geq (h_i)_s$. De plus, $(h_i)_s \geq h_i$ par $C1$ et $h_i \geq h_{i+1}$ par hypothèse donc $r_m > h_{i+1}$. On peut, dès lors, écrire $vh_{i+1} \cdots h_n$ par la factorisation décroissante en mots de Hall suivante :

$$v_h i + 1 \cdots h_n = r_1 \cdots r_m h_{i+1} \cdots h_n.$$

□

Grâce à ce résultat, on dispose de la preuve du théorème 4.19.

Démonstration. **(Démonstration du théorème 4.19)**

Existence : Le travail mené en amont sur les suites standards fournit un algorithme permettant d'avoir une telle factorisation. En effet, nous allons construire une suite de suites standards $s_0, s_1 = s'_0, s_2 = s'_1, \ldots, s_p = s'_{p-1}$ avec $s_p = (h_1, \ldots, h_n)$, $h_1 \geq \ldots \geq h_n$ et $w = h_1 \ldots h_n$.
Si $w = a_1 \cdots a_m$ avec $a_i \in \mathcal{A}$ pour tout $i \in \{1, \ldots, m\}$, alors $s = (a_1, \ldots, a_m)$ est une suite standard. Dès lors, on peut poser $s_0 = (a_1, \ldots, a_m)$ et calculer $s_1, s_2, \ldots$ Lorsque l'algorithme sera fini, on arrivera à $s_p = (h_1, \ldots, h_n)$. Par $C5$, w est le mot obtenu en concaténant les mots obtenu dans s_i pour tout $i \in \{1, \ldots, p\}$. Donc, en particulier, $w = h_1 \cdots h_n$, et l'existence de la factorisation est démontré.

Unicité : Supposons que le mot w possède plus d'une factorisation :

$$w = k_1 \cdots k_m = h_1 \cdots h_n,$$

où les k_i et les h_j sont des mots de Hall tels que $k_1 \geq \ldots \geq k_m$ et $h_1 \geq \ldots \geq h_n$. Procédons maintenant par l'absurde. Par le théorème 4.12, on suppose que $n > 1$ et $m > 1$. De plus, on suppose $|k_m| > |h_n|$ (car, si $|k_m| = |h_n|$, alors $k_m = h_n$ et, par induction sur la longueur de w, on aurait $m = n$ et $k_i = h_i$). Une fois ces hypothèses faites, on a $k_m = vh_{i+1} \cdots h_n$ où $i < n$ et v est un suffixe non vide de h_i. Dès lors, il est impossible d'avoir $v = h_i$ par le théorème 4.12. Donc, par v est un suffixe propre et non vide de h_i et, par le lemme 4.20, k_m se factorise en

$$k_m = vh_{i+1} \cdots h_n = r_1 \cdots r_p h_{i+1} \cdots h_n$$

avec $r_1, \ldots, r_p \in H$ et $r_1 \geq r_2 \geq \ldots \geq r_p \geq h_{i+1} \geq \ldots \geq h_n$.
Cependant, cette factorisation contredit le théorème 4.12 d'où l'unicité de la factorisation de w.

□

4.2.2 Méthode alternative

L'objectif de cette partie est de donner une preuve alternative de l'unicité de la factorisation définie dans le théorème 4.19. Pour ce faire, comme précisé plus haut, cette démonstration va se baser sur une relation binaire $\to$ définie sur l'ensemble des suites standards de mots de Hall. Si s et t sont deux suites standards sur des mots de Hall,

$$s \to t \Leftrightarrow t = s' \text{ où } s' \text{ est donné par la C5.}$$

En d'autres termes, on a $s \to t$ si t est obtenu en concaténant deux mots consécutifs d'un accroissement. Si $s \to t$, alors on dira que t est dérivé de s.

De cette définition, il est possible de déduire les trois résultats suivants qui seront également utiles dans la démonstration de l'unicité de la factorisation du théorème 4.19.

Proposition 4.21. *La relation $\to$ est confluente. En d'autres termes, si s_1 et s_2 sont deux suites standards dérivées de s, alors il existe une suite standard t telle que $s_1 \to t$ et $s_2 \to t$.*

Démonstration. Soit $s = (h_1, \ldots, h_n)$ une suite standard. Si (h_i, h_{i+1}) et (h_j, h_{j+1}) sont deux accroissements légaux, alors on définit

$$s_1 = (h_1, \ldots, h_{i-1}, h_i h_{i+1}, h_{i+2}, \ldots, h_n)$$

$$s_2 = (h_1, \ldots, h_{j-1}, h_j h_{j+1}, h_{j+2}, \ldots, h_n)$$

On peut supposer que $i < j$. En fait, on peut même supposer que $i + 1 < j$. En effet, si on suppose que $j = i + 1$, on a que $h_{i+1} < h_{i+2}$, car (h_j, h_{j+1}) est un accroissement, et $h_{i+1} \geq h_{i+2}$ car (h_i, h_{i+1}) est aussi un accroissement légal. On obtient donc une contradiction et on peut donc supposer que $i + 1 < j$. Donc, on a (h_i, h_{i+1}) est un accroissement dans s_2 comme (h_j, h_{j+1}) l'est dans s_1. Le rise (h_j, h_{j+1}) dans s_1 est un accroissement car l'accroissement est à droite de h_{i+1}. De même, (h_i, h_{i+1}) est aussi un accroissement dans s_2 via la même justification. En effet, par la proposition 4.18, le mot $h_j h_{j+1}$ possède une factorisation standard telle que $(h_j h_{j+1})_p = h_j$ et $(h_j h_{j+1})_s = h_{j+1}$. On a $h_{i+1} \geq h_{j+1}$ et, par $C1$, $h_{j+1} > h_j h_{j+1}$ ce qui implique $h_{i+1} > h_j h_{j+1}$. Donc, l'accroissement (h_i, h_{i+1}) est un accroissement légal de s_2.
Comme $i + 1 < j$, l'accroissement légal (h_j, h_{j+1}) ne recouvre pas l'accroissement légal (h_i, h_{i+1}) dans s. Dès lors, le travail réalisé sur les accroissements (h_i, h_{i+1}) et (h_j, h_{j+1}) dans, respectivement, s_1 et s_2 produit la même suite standard

$$t = (h_1, \ldots, h_{i-1}, h_i h_{i+1}, \ldots, h_j h_{j+1}, h_{j+2}, \ldots, h_n).$$

□

Il est également possible de définir la relation $\xrightarrow{*}$ comme la clôture réflexive et transitive de $\to$. Si $s \xrightarrow{*} t$, on utilisera toujours le même vocabulaire, à savoir que t est dérivé de s. De plus, si $s \xrightarrow{*} t$ est donné par la chaine $s = s_0 \to s_1 \to \ldots \to s_n = t$, alors on dira que la dérivation $s \xrightarrow{*} t$ est de longueur n. Grâce à cette définition, on obtient le corollaire suivant :

Corollaire 4.22. *La relation $\xrightarrow{*}$ est confluente.*

Démonstration. Cela se démontre via une simple récurrence sur la longueur de la dérivation. □

Revenons quelques instants sur la définition de suites standards. Grâce à cette définition, toutes les suites de lettres sont des exemples de suites standards (cf. exemple 4.16). De plus, par la proposition 4.18, on déduit que n'importe quelle suite standard est dérivée d'une suite de lettres. Le résultat suivant va montrer, en plus, qu'aucune suite standard n'échappe à cette règle.

Proposition 4.23. *Soit $t = (h_1, \ldots, h_n)$ une suite standard. Si tous les h_i ne sont pas des lettres, alors il existe une suite standard s telle que $s \to t$.*

Démonstration. Soit h_i un mot dans t tel que, si h_j n'est pas une lettre, alors $(h_j)_s \geq (h_i)_s$ pour tout $j \in \{1, \ldots, i-1\}$. Un tel mot h_i existe si on le prend tel que h_i est le mot le plus à gauche de t qui n'est pas réduit à une lettre. Montrons maintenant que $s = (h_1, \ldots, h_{i-1}, (h_i)_p, (h_i)_s, h_{i+1}, \ldots, h_n)$ est une suite standard et que $((h_i)_p, (h_i)_s)$ est un accroissement de s. On a $((h_i)_s)_s > (h_i)_s$ par $C1$, $((h_i)_p)_s > (h_i)_s$ par $C3$ et $(h_i)_s \geq h_j$ pour tout $j \in \{i + 1, \ldots, n\}$ car t est une suite standard. On en déduit donc que $((h_i)_s)_s, ((h_i)_p)_s \geq h_j$ pour tout $j \in \{i + 1, \ldots, n\}$. Comme, par

hypothèse, $(h_j)_s \geq (h_i)_s$ pour tout $j \in \{1, \ldots, i-1\}$ et, par $C2$, $(h_i)_s > (h_i)_p$, on a $(h_j)_s \geq (h_i)_s, (h_i)_s$ pour tout $j \in \{1, \ldots, i-1\}$. De plus, comme t est une suite standard, l'accroissement $((h_i)_p, (h_i)_s)$ est *legal* car $(h_i)_s \geq h_{i+1}, \ldots, h_n$. Donc, t doit être dérivé de s. □

À l'aide de ces trois résultats concernant la relation $\to$, il est possible de démontrer l'unicité de la factorisation du théorème 4.19 d'une seconde manière.

Démonstration. **(Démonstration de l'unicité de la factorisation du théorème 4.19)**
Tout d'abord, considérons une suite standard $s = (h_1, \ldots, h_m)$ et $w = h_1 \cdots h_m = a_1 \cdots a_p$ avec $a_i \in \mathcal{A}$ le mot associé à la suite standard s. Dès lors, en appliquant un certain nombre de fois la proposition 4.23, on peut trouver une dérivation $(a_1, \cdots, a_p) \xrightarrow{*} s$.
Supposons maintenant que $w = a_1 \cdots a_p$ possède deux factorisations décroissantes en mots de Hall :

$$w = h_1 \cdots h_m = k_1 \cdots k_n \quad \text{avec } h_1 \geq \ldots \geq h_m,\ k_1 \geq \ldots \geq k_n.$$

Comme $s_1 = (h_1, \ldots, h_m)$ et $s_2 = (k_1, \ldots, k_n)$ sont des suites de mots de Hall décroissantes, elles sont toutes les deux des suites standards. Donc, il existe deux dérivation $(a_1, \ldots, a_p) \xrightarrow{*} s_1$ et $(a_1, \ldots, a_p) \xrightarrow{*} s_2$. Par le corollaire 4.22, il existe une suite standard t telle que $s_1 \xrightarrow{*} t$ et $s_2 \xrightarrow{*} t$. Comme les deux suites s_1 et s_2 sont des suites décroissantes, la seule suite que l'on peut dériver d'elles sont les suites s_1 et s_2 elles-mêmes et donc $s_1 = t = s_2$. De cette égalité, on en tire que $m = n$ et $h_i = k_i$ pour tout $i \in \{1, \ldots, n\}$ □

4.3 Propriétés supplémentaires des mots de Hall

Dans la fin de cette partie consacrée aux mots de Hall, une caractérisation de ces derniers sera exhibée : leur minimalité par rapport aux autres mots de leur classe de conjugaison. Pour ce faire, il est impératif de définir un ordre total $<_H$ sur $\mathcal{A}^*$, et ce, en utilisant la factorisation en mot de H.

Définition 4.24. Soit $u, v \in \mathcal{A}^*$ et considérons leur factorisation décroissante en mots de Hall :

$$u = u_1 \cdots u_n, \quad \text{et} \quad v = v_1 \cdots v_m$$

On dira que u est plus petit que v, noté $u <_H v$, si une des deux conditions est rencontrée :

- si $n < m$ et $u_i = v_i,\ \forall i \in \{1, \ldots, n\}$,
- s'il existe i tel que $u_1 = v_1,\ u_2 = v_2, \ldots,\ u_{i-1} = v_{i-1}$ et $u_i < v_i$ où $<$ est l'ordre total défini sur H par la remarque 4.13.

Pour établir la démonstration de la minimalité par rapport aux autres mots dans leur classe de conjugaison, il est utile de remarquer les deux lemmes ci-dessous.

Lemme 4.25. *Soit $s = (h_1, \ldots, h_n)$ une suite standard de mots de Hall de longueur au moins 2 et telle que h_1 est maximal, c'est-à-dire, $h_1 \geq h_2, \ldots, h_n$. Alors, toute suite dérivée de s est de taille au moins 2 et a pour premier élément h_1.*

Démonstration. On a, par hypothèse, que (h_1, h_2) n'est pas un accroissement de s. Donc, s' est de longueur au moins 2 et est égal à

$$s' = (h_1, \ldots, h_{i-1}, h_i h_{i+1}, h_{i+2}, \ldots, h_n),$$

si (h_i, h_{i+1}) est un accroissement avec $2 \geq i \geq n$. On a également $h_1 \geq h_{i+1}$ et, par $C1$, $h_{i+1} > h_i h_{i+1}$. Dès lors, s' satisfait également les hypothèses du lemme. Donc, cela permet de conclure par récurrence sur la taille des dérivations. □

Lemme 4.26. *Soit $s = (h_1, \ldots, h_n)$ une suite de mots de Hall de longueur au moins 2 telle que h_1 est maximal ($h_1 \geq h_2, \ldots, h_n$). Alors, il existe une suite standard $t = (k_1, \ldots, k_m)$ telle que*

$$h_1 \cdots h_n = k_1 \cdots k_m,\ h_1 = k_1,\ k_1 \geq k_2, \cdots, k_m$$

Démonstration. Pour démontrer ce lemme, définissons la notion de disparité $\delta(s)$ d'une suite s comme suit : pour toute suite $s = (h_1, h_2, \ldots, h_n)$, la disparité $\delta(s) = |h_1| + |h_2| + \ldots + |h_n| - n$. Montrons maintenant le lemme par récurrence sur la disparité $\delta(s)$ de la suite s.

Cas de base : Les suites s dont la disparité $\delta(s)$ vaut 0 sont les suites de suites de lettres et sont donc des suites standards.

Induction : Considérons une suite $s = (h_1, \ldots, h_n)$ vérifiant les hypothèses du lemme. Si la suite s est standard, le résultat est trivial ($t = s$). Si s n'est pas une suite standard, alors il existe $i < j$ tels que

$$i < j, \quad h_i \text{ n'est pas une lettre et } (h_i)_s < h_j.$$

On sait que $i > 1$ car $h_1 < (h_1)_s$ par $C1$ et $h_1 \geq h_2, \ldots, h_n$ par hypothèse. Dès lors, la suite $t = (h_1, \ldots, h_{i-1}, (h_i)_p, (h_i)_s, h_{i+1}, \ldots, h_n)$ a une disparité $\delta(t) = \delta(s) - 1$. De plus, t vérifie les hypothèses. En effet, on a $(h_i)_p < (h_i)_s$ par $C2$ d'où $(h_i)_p < (h_i)_s < h_j \leq h_1$. Comme $h_1 h_2 \cdots h_i \cdots h_n = h_1 h_2 \cdots (h_i)_p (h_i)_s \cdots h_n$, on conclut en utilisant l'hypothèse de récurrence.

□

Grâce à ces deux lemmes, il est maintenant possible de montrer la minimalité d'un mot de Hall dans sa classe de conjugaison.

Proposition 4.27. *Soit w un mot sur $\mathcal{A}^*$. Si w est un mot de Hall, alors, pour toute factorisation de w en mots non-vides, $w = uv$, on a $w <_H vu$.*

Démonstration. Soit w un mot de Hall. Transposons le lemme 4.8 à l'ensemble des mots de Hall. Donc, on a $w = uv$ avec $u, v \in \mathcal{A}^+$ et on a

$$u = q_1 \cdots q_m, \quad v = r_1 \cdots r_n,$$

avec

$$q_i, r_j \in H\ \forall i, j,\ q_1, \ldots, q_m < r_1,\ r_1 \geq \ldots \geq r_n \geq w_s. \qquad (*)$$

Dès lors, $w = uv = q_1 \cdots q_m r_1 \cdots r_n$. En conséquences, le conjugué vu de w s'écrit $vu = r_1 \cdots r_n q_1 \cdots q_m$.
Calculons maintenant la factorisation de vu afin de comparer vu et w. Grâce aux

inégalités de $(*)$, on remarque que la suite $s = (r_1, \ldots, r_n, q_1, \ldots, q_m)$ satisfait les hypothèses du lemme 4.26. Par ce lemme, il est possible de trouver une suite standard

$$t = (k_1, \ldots, k_p)$$

telle que $vu = r_1 \cdots r_n q_1 \cdots q_m = k_1 \cdots k_p$,

$$r_1 = k_1, \quad \text{et } k_1 \geq k_2, \ldots, k_p \tag{$**$}$$

Pour obtenir la factorisation en un produit décroissant de mots de Hall de vu, nous allons travailler sur la suite $t = (k_1, \ldots, k_p)$. Grâce à $(**)$, on remarque aisément que t est une suite respectant les hypothèses du lemme 4.25. Dès lors, toute suite dérivée de t aura une longueur supérieure ou égale à 2 et commencera par k_1. Donc, on sait qu'il y aura au moins deux mots de Hall dans la factorisation de vu et cette dernière commencera par k_1 ce qu'on peut écrire comme suit :

$$vu = h_1 h_2 \cdots h_d \quad \text{avec } d \geq 2 \text{ et } h_1 = k_1 \tag{$***$}$$

Maintenant que l'on dispose de la factorisation de vu, comparons w et vu au travers de leur factorisation en mots de Hall. Pour cela, comparons h_1 et w. En utilisant $(*), (**)$ et $(***)$, on obtient que $h_1 \geq w_s$ et $h_1 > w$ par $C1$. Au final, par définition de $<_H$, on obtient que $h_1 >_H w$ et, en conséquences, $vu >_H w$. □

Une autre propriété des mots de Hall est qu'ils sont primitifs. En effet, il est possible de la démontrer grâce aux théorèmes 3.22 et 4.19. Cependant, une deuxième manière de la démontrer est de se baser uniquement sur la proposition ci-dessus comme le fait le corollaire ci-dessous.

Corollaire 4.28. *Tout mot de Hall w sur $\mathcal{A}^*$ est primitif.*

Démonstration. Procédons par l'absurde et supposons que w ne soit pas primitif. Donc, il existe des mots u, v tels que $w = uv = vu$. Par le théorème 4.27, on obtient que $w <_H vu = w$ ce qui est une contradiction. D'où le fait que w est primitif. □

Une question toute naturelle est de se demander si la réciproque du théorème 4.27 est vraie. Si, pour toute factorisation de w en $w = uv$ avec $u, v \in \mathcal{A}^+$, $w <_H vu$, alors w est un mot de Hall. Pour répondre à cette question, introduisons la notion de suite circulairement standard.

Définition 4.29. (Suite circulairement standard)
Une suite de mots de Hall $\sigma = (h_1, \ldots, h_n)$ est dite circulairement standard si, pour tout $i \in \{1, \ldots, n\}$, σ vérifie une des deux conditions suivantes :

- h_i est une lettre,
- $h_i = (h_i)_p (h_i)_s$ et $(h_i)_s \geq h_1, h_2, \ldots, h_n$.

Remarque 4.30. Les suites circulairement standards peuvent être définies de la manière suivante. La suite σ est dite circulairement standard si, pour tout $i \in \{1, \ldots, n\}$, la suite $(h_i, \ldots, h_n, h_1, \ldots, h_{i-1})$ est une suite standard.

Exemple 4.31. Toute suite de lettre de $\mathcal{A}$ est une suite circulairement standard.

Tous les raisonnements effectués dans le cadre des suites standards peuvent être transposés aux suites circulairement standards. De ce fait, un accroissement légal (h_i, h_{i+1}) d'une suite circulairement standard est défini comme étant un accroissement (on garde la même définition d'accroissement que celle des suites standards) tel que $h_{i+1} \geq h_1, \ldots, h_n$. Autrement dit, (h_i, h_{i+1}) sera un accroissement légal si h_{i+1} est maximal parmi les termes de la suite pour l'ordre $<$ sur les mots de Hall. On considèrera l'accroissement (h_n, h_1) si $h_n < h_1$. Dans l'article [13] de Mélançon, il résume la définition d'accroissement légal et le fait de prendre l'accroissement (h_n, h_1) par dire que les indices sont pris modulo n.

Similairement aux suites standards, on définit également une suite σ' comme suit :

Définition 4.32. Soient σ une suite circulairement standard et (h_i, h_{i+1}) un accroissement légal de σ (les indices étant pris modulo n). On définit la suite σ' par

$$\sigma' = \begin{cases} (h_1, \ldots, h_i h_{i+1}, \ldots, h_n) & \text{si } i < n \\ (h_n h_1, h_2, \ldots, h_{n-1}) & \text{si } i = n \end{cases}$$

Remarque 4.33. On définit le mot associé à une suite circulairement standard de la même manière que pour les suites standards, c'est-à-dire, si $\sigma = (h_1, \ldots, h_n)$, alors le mot associé à σ est $h_1 \cdots h_n$.

Tout comme c'est le cas pour les suites standards, il est également possible de lier l'aspect standard de σ' avec celui de σ.

Proposition 4.34. *Si $\sigma = (h_1, \ldots, h_n)$ une suite circulairement standard, alors σ' est aussi une suite circulairement standard. De plus, les mots associés aux suites σ et σ' appartiennent à la même classe de conjugaison.*

Démonstration. Soit (h_i, h_{i+1}) un accroissement légal, les indices étant pris modulo n. D'une part, par définition d'accroissement légal pour les suites circulairement standard, h_{i+1} est maximal parmi les mots de σ. D'autre part, on a $(h_i h_{i+1})_s = h_{i+1}$ par la proposition 4.18. Dès lors, $(h_i h_{i+1})_s \geq h_1, \ldots, h_{i-1}, h_i h_{i+1}, h_{i+2}, \ldots, h_n$ et σ' est une suite circulairement standard. De plus, les mots associés aux suites σ et σ' appartiennent trivialement à la même classe de conjugaison par définition de σ'. □

À l'aide de ces différents résultats, il convient de montrer que tout mot primitif sur un alphabet $\mathcal{A}$ est le conjugué d'un mot de Hall.

Corollaire 4.35. *Tout mot primitif sur $\mathcal{A}^*$ est le conjugué d'un unique mot de Hall.*

Démonstration. L'idée de la preuve est de construire une suite $(\sigma_n)_n$ de suite circulairement standard à partir de σ_0 une suite dont chaque élément est une lettre du mot de départ, noté w, et définit par $\sigma_{n+1} = \sigma'_n$, $\forall n \geq 0$. Cette procédure s'arrêtera quand la suite ne contient plus qu'un mot ou quand la suite ne contient qu'une répétition du même mot car, dans ce cas, il n'y aura plus d'accroissements. Dans ce cas, le dernier élément de cette suite, noté σ_f, est tel que, si $\sigma_f = (h_1, \ldots, h_n)$, alors w et $h_1 \cdots h_n$ appartiennent à la même classe de conjugaison par la proposition 4.34. Prenons $w \in \mathcal{A}^*$ un mot primitif. Comme expliqué ci-dessus, construisons la suite $(\sigma_n)_n$ de suites circulairement standard. Si $w = a_1 \cdots a_m$ avec $\forall i \in \{1, \ldots, m\}, a_i \in \mathcal{A}$,

alors on pose $\sigma_0 = (a_1, \ldots, a_m)$ et on démarre la procédure décrite ci-dessus. Donc, $\sigma_1 = \sigma'_0$, $\sigma_2 = \sigma'_1$ et ainsi de suite. Lorsque la procédure se termine, on a atteint $\sigma_f = (h_1, \ldots, h_n)$ avec $h_1 = \ldots = h_n$. Par la proposition 4.34, les mots associés à σ_i et σ_{i+1} sont conjugués l'un de l'autre et ce pour tout i. Donc, w et $h_1 \cdots h_n$ sont conjugués. De plus, si un mot est primitif, alors ses conjugués le sont aussi. Par conséquences, $h_1 \cdots h_n$ doit être primitif ce qui implique que $n = 1$. Au final, w est donc bien le conjugué d'un mot de Hall. Comme les mots de Hall sont minimaux dans leur classe de conjugaison par la proposition 4.27, cela implique que h_1 est unique. □

Remarque 4.36. Dans la suite de ce mémoire, l'algorithme décrit dans cette démonstration et permettant de trouver le mot de Hall appartenant dans la même classe de conjugaison qu'un mot donné sera appelé **algorithme de Mélançon**.

Grâce à ces différents résultats, on peut, dès à présent, démontrer la réciproque de la proposition 4.27.

Proposition 4.37. *Soit un mot $w \in \mathcal{A}^*$ tel que, pour toute factorisation de w en mots non vides, $w = uv$, $w <_H vu$. Alors w est un mot de Hall.*

Démonstration. Vu les hypothèses de la proposition, w doit être primitif. Supposons que w n'est pas primitif. Il est donc possible de trouver deux mots non vides u et v tels que $w = uv = vu$. Cependant, cela contredirait l'aspect strictement minimal de w par rapport à ces conjugués ($w <_H vu$). Donc , w est bien un mot primitif. Enfin, cette proposition découle du corollaire 4.35 et de la proposition 4.27. □

Au final, on obtient donc une caractérisation des mots de Hall.

Théorème 4.38. *(Caractérisation des mots de Hall)*
Soit w un mot sur $\mathcal{A}^*$. w est un mot de Hall si et seulement si, pour toute factorisation de w en mots non vides, $w = uv$, $w <_H vu$.

Démonstration. Cela résulte des propositions 4.27 et 4.37. □

Grâce à cette caractérisation, on peut enfin donner un exemple concret de famille de mots de Hall : les mots de Lyndon. En effet, dans le premier chapitre de ce mémoire, la définition 2.9 et le théorème 4.38 permettent de montrer que tout mot de Lyndon est, en fait, un mot de Hall. Dans l'article [13] de Guy Mélançon, on peut même retrouver le fait que les mots de Hall généralise les mots de Lyndon

Après avoir remarqué le lien entre les mots de Lyndon et les mots de Hall, on pourrait se pencher sur les propriétés qu'ont les mots de Hall et qui ressemblent à celle des mots de Lyndon. Dans le premier chapitre de ce mémoire, les mots de Lyndon sont définis par le fait qu'ils soient primitifs et minimaux parmi leur classe de conjugaison. Il a été montré que l'on pouvait aussi définir un mot de Lyndon par le fait qu'un mot de Lyndon était plus petit que tout ses suffixes propres (cf. définition 2.11). Dès lors, il sera également possible de définir les mots de Hall de la même manière. On peut montrer que c'est une caractérisation des mots de Hall à l'aide de la proposition ci-dessous.

Proposition 4.39. *Soit $h = h_p h_s$ un mot de Hall de longueur au moins 2. Alors*

(i) parmi tous les suffixes propres de h, h_s est de longueur maximal,

(ii) parmi tous les suffixes propres de h, h_s est minimal pour l'ordre $<_H$.

Démonstration. (i) Soit v un suffixe propre de h de longueur supérieur à celle de h_s. Par le lemme 4.8, la factorisation de v est donnée par

$$v = h_1 \cdots h_n, \quad \text{avec } h_1, \ldots, h_n \in H,\ h_1 \geq \ldots \geq h_n.$$

Par la remarque 4.9, il y a au moins deux facteurs dans la factorisation de v et donc v n'est pas un mot de Hall.

(ii) Pour prouver ce point, procédons en deux temps. Soit v un suffixe propre de h dont la longueur est supérieur à celle de h_s. Par la remarque 4.9, on a

$$v = h_1 \cdots h_n \text{ avec } h_1, \ldots, h_n \in H,\ h_1 \geq \ldots \geq h_n,\ n \geq 2,\ h_1 \geq (h_p)_s. \quad (*)$$

Comparer v et h_s revient à comparer h_1 et h_s. Par $C1, C3$ et $(*)$, on trouve que $h < h_s \leq (h_p)_s \leq h_1$. Donc, $h_s <_H v$.
Prenons maintenant un suffixe propre v de h dont la longueur est inférieure à celle de h_s. Par la remarque 4.9, on a de nouveau

$$v = h_1 \cdots h_n,\ h_1, \ldots, h_n \in H,\ h_1 \geq \ldots \geq h_n,\ n \geq 2,\ h_1 \geq \ldots \geq h_n \geq (h_s)_s. \quad (**)$$

De même, comparer v et h_s revient à comparer h_1 et h_s. En utilisant $C1$ et $(**)$, on a $h < h_s < (h_s)_s \leq h_1$. Donc, $h_s <_H v$.

□

Théorème 4.40. *Soit w un mot sur $\mathcal{A}^*$. Alors w est un mot de Hall si et seulement si w est plus petit que tout ses suffixes propres.*

Démonstration. $\Rightarrow$ Supposons que w est un mot de Hall et prenons v un suffixe propre de w. Dès lors, par la proposition 4.39 (ii), on a $w_s \leq_H v$. Comme $w <_H w_s$, on a $w <_H v$.

$\Leftarrow$ Supposons maintenant que w est plus petit que tout ses suffixes propres. Procédons par l'absurde en supposons que w n'est pas un mot de Hall. On peut donc le factoriser en

$$w = h_1 \cdots h_n \quad \text{avec } h_1 \geq \ldots \geq h_n,\ n \geq 2$$

On veut obtenir que $h_1 > h_n$. Si, au contraire, $h_1 = \ldots = h_n$, alors, par définition de $<_H$, on aurait

$$h_i \cdots h_n <_H h_1 \cdots h_n = w \quad \text{pour } i \in \{2, \ldots, n\}.$$

Mais, cela contredit l'hypothèse faite sur w. Donc, $h_1 < h_n$. Cependant, cela implique que $h_n <_H h_1 \cdots h_n = w$ ce qui contredit l'hypothèse faite sur w. Donc, w doit être un mot de Hall.

□

Grâce à ce théorème, il est même possible de généraliser la proposition 4.39 à l'ensemble de tous les mots sur $\mathcal{A}^*$.

Proposition 4.41. *Soient w un mot de $\mathcal{A}^*$ et $w = h_1 \cdots h_n$ sa factorisation décroissante en mots de Hall. Alors*

- (i) *parmi tous les suffixes de w qui sont des mots de Hall, h_n est de longueur maximal*
- (ii) *parmi tous les suffixes de w, h_n est minimal pour l'ordre $<_H$.*

Démonstration. Si w est un mot de Hall, alors le point (i) est direct et le point (ii) est trivial par le théorème 4.40.
Supposons maintenant que w n'est pas un mot de Hall et que $n \geq 2$. Soit z un suffixe propre de w. Si $|z| < |h_n|$, alors, par le théorème 4.40, on a $h_n <_H z$. Si $|z| > |h_n|$, alors $z = vh_{i+1} \cdots h_n$ avec v un suffixe de h_i avec $i < n$. Dès lors, soit $v = h_i$, soit v est un suffixe propre de h_i, auquel cas, par le lemme 4.20, z se factorise en

$$z = vh_{i+1} \cdots h_n = k_1 \cdots k_m h_{i+1} \cdots h_n, \qquad \text{avec } k_1 \geq \ldots \geq k_m > h_{i+1} \geq \ldots \geq h_n \tag{$*$}$$

Peu importe le cas considéré, on remarque la présence d'au moins deux facteurs dans la factorisation de z. Donc, z n'est pas un mot de Hall et le point (i) de cette proposition est montré.

En ce qui concerne le point (ii), pour comparer z et h_n, il suffit de comparer k_1 et h_n. En utilisant $(*)$, on a $k_1 > h_n$. Donc, $z >_H h_n$ ce qui montre le deuxième point. □

Enfin, la factorisation standard d'un mot de Hall w peut être trouvé à partir de la factorisation en mots de Hall de son suffixe propre de taille maximal.

Corollaire 4.42. *Soit $w \in \mathcal{A}^*$ un mot de Hall de longueur au moins deux : $w = az$ avec $a \in \mathcal{A}$ et $z \in \mathcal{A}^*$, $|z| \geq 1$. Supposons que la factorisation de z en mots de Hall soit*

$$z = h_1 \cdots h_n, \qquad \text{avec } h_1 \geq \ldots \geq h_n.$$

Alors, $w_s = h_n$.

Démonstration. Cela découle des propositions 4.41 et 4.39. □

4.4 Ensembles de Lazard

D'autres ensembles liés à l'ensemble des mots de Nyldon sont les ensembles de Lazard. Pour commencer le travail consistant à lier ensemble de Lazard et ensemble des mots de Nyldon, voici les définitions d'ensemble de Lazard et de Viennot.

Définition 4.43. (Ensemble de Lazard et ensemble de Viennot)
Soit $\mathcal{A}$ un alphabet.
Un **ensemble de Lazard à gauche** (resp. **ensemble de Lazard à droite**) est un sous-ensemble F de A^+ muni d'un ordre total $\prec$ tel que, pour tout naturel $n \geq 1$, si $F \cap A^{\leq n} = \{u_1, \ldots, u_{k+1}\}$ avec $k \geq 1$ et $u_1 \prec \ldots \prec u_{k+1}$ (resp. $u_1 \succ \ldots \succ u_{k+1}$), et si on considère la suite d'ensembles $(Y_i)_{i \geq 1}$ défini par $Y_0 = \mathcal{A}$ et, pour $i \geq 0$, $Y_{i+1} = u_i^*(Y_i \backslash \{u_i\})$ (resp. $Y_{i+1} = (Y_i \backslash \{u_i\})u_i^*$), alors on a

(*i*) pour tout $i \in \{0, \ldots, k\}$, $u_{i+1} \in Y_i$

(*ii*) $Y_k \cap \mathcal{A}^{\leq n} = \{u_{k+1}\}$.

Si F est un ensemble de Lazard à gauche et à droite, alors F est appelé un **ensemble de Viennot**.

Suite à cette définition, il est trivial de remarquer que, pour tout $i \in \{0, \ldots, k\}$, aucun mot de Y_i n'est préfixe d'un autre mot de Y_i. De plus, un exemple d'ensemble de Viennot bien connu est la famille des mots de Lyndon. Dans l'exemple ci-dessous, on peut voir comment construire les mots de Lyndon jusqu'à la longueur 6 en se basant sur l'algorithme donné par la définition 4.43.

Exemple 4.44. Concentrons-nous d'abord sur l'ensemble de Lazard à gauche. Construisons la suite d'ensemble $(G_i)_{i\geq 0}$ telle que $G_0 = \mathcal{A} = \{0, 1\}$ et, pour tout $i \geq 0, G_{i+1} = m_i^*(G_i \backslash \{m_i\})$ avec $m_i = \min_{<_{\text{lex}}}(G_i \cap \{0, 1\}^{\leq 6})$. Le tableau 4.10 est construit de la manière suivante : la première colonne représente le nombre d'étape réalisé par l'algorithme, la deuxième colonne représente la liste construite par $m_i^*(G_i \backslash \{m_i\})$ et la troisième colonne représente l'élément m_i que l'on va enlever. Dès lors, on obtient le tableau 4.10.

i	$G_i \cap \mathcal{A}^{\leq 6}$	m_i
1	$\{0, 1\}$	0
2	$\{000001, 00001, 0001, 001, 01, 1\}$	000001
3	$\{00001, 0001, 001, 01, 1\}$	00001
4	$\{000011, 0001, 001, 01, 1\}$	000011
5	$\{0001, 001, 01, 1\}$	0001
6	$\{000101, 00011, 001, 01, 1\}$	000101
7	$\{00011, 001, 01, 1\}$	00011
8	$\{000111, 001, 01, 1\}$	000111
9	$\{001, 01, 1\}$	001
10	$\{00101, 0011, 01, 1\}$	00101
11	$\{001011, 0011, 01, 1\}$	00101
12	$\{0011, 01, 1\}$	0011
13	$\{001101, 00111, 01, 1\}$	001101
14	$\{00111, 01, 1\}$	00111
15	$\{001111, 01, 1\}$	001111
16	$\{01, 1\}$	01
17	$\{01011, 011, 1\}$	01011
18	$\{010111, 011, 1\}$	010111
19	$\{011, 1\}$	011
20	$\{0111, 1\}$	0111
21	$\{01111, 1\}$	01111
22	$\{011111, 1\}$	011111
23	$\{1\}$	1

FIGURE 4.10 – Tableau 1 : mots de Lyndon

Dans cette construction, on dit que G_{i+1} est obtenu par élimination de Lazard de m_i dans G_i. On remarque que tous les mots de Lyndon de longueur inférieure ou égale à 6 sont construits via cet algorithme. De plus, les conditions (i) et (ii) sont bien respectées. Comme cette construction respectera toujours (i) et (ii) pour tout n, l'ensemble des mots de Lyndon forme un ensemble de Lazard à gauche.

Ensuite, il est possible de répéter l'algorithme mais en respectant la définition des ensembles de Lazard à droite. Donc, dans ce cas-ci, la suite des ensembles $(D_i)_{i\geq 0}$ est construite tels que $D_0 = \mathcal{A} = \{0, 1\}$ et, pour tout $i \geq 0$, $D_{i+1} = (D_i \backslash \{M_i\})M_i^*$ où $M_i = \max_{<_{\text{lex}}}(D_i \cap \mathcal{A}^{\leq 6})$. Dès lors, en suivant les mêmes règles de construction du tableau ci-dessus, on obtient le tableau 4.11.

i	$D_i \cap \mathcal{A}^{\leq 6}$	M_i
1	$\{1, 0\}$	1
2	$\{011111, 01111, 0111, 011, 01, 0\}$	0111111
3	$\{01111, 0111, 011, 01, 0\}$	01111
4	$\{0111, 011, 01, 001111, 0\}$	0111
5	$\{011, 010111, 01, 001111, 00111, 0\}$	011
6	$\{010111, 01011, 01, 001111, 00111, 0011, 0\}$	010111
7	$\{01011, 01, 001111, 00111, 0011, 0\}$	01011
8	$\{01, 001111, 00111, 0011, 001011, 0\}$	01
9	$\{001111, 001101, 0011, 001011, 00101, 001, 0\}$	001111
10	$\{00111, 001101, 0011, 001011, 00101, 001, 0\}$	00111
11	$\{001101, 0011, 001011, 00101, 001, 000111, 0\}$	001101
12	$\{0011, 001011, 00101, 001, 000111, 0\}$	0011
13	$\{001011, 00101, 001, 000111, 00011, 0\}$	001011
14	$\{00101, 001, 000111, 00011, 0\}$	00101
15	$\{001, 000111, 00011, 000101, 0\}$	001
16	$\{000111, 00011, 000101, 0001, 0\}$	000111
17	$\{00011, 000101, 0001, 0\}$	00011
18	$\{000101, 0001, 000011, 0\}$	000101
19	$\{0001, 000011, 0\}$	0001
20	$\{000011, 00001, 0\}$	000011
21	$\{00001, 0\}$	00001
22	$\{000001, 0\}$	000001
23	$\{0\}$	0

FIGURE 4.11 – Tableau 2 : mots de Lyndon

Comme dans la construction précédente, on dit que les D_{i+1} sont obtenus par élimination de Lazard de m_i dans D_i. De même, tous les mots de Lyndon de longueur inférieure ou égale à 6 sont construits via cet algorithme. Enfin, les conditions (i) et (ii) sont bien respectées. Comme cette construction respectera toujours (i) et (ii) pour tout n, l'ensemble des mots de Lyndon forme un ensemble de Lazard à droite. Donc, l'ensemble des mots de Lyndon forme un ensemble de Viennot.

Dans la suite de ce travail, l'algorithme permettant de construire les ensembles Y_i de la définition 4.43 sera appelé procédure de Lazard. Si on construit la suite d'ensemble $(Y_i)_{i\geq 0}$ en construisant $Y_{i+1} = u_i^*(Y_i \backslash \{u_i\})$ (resp. $Y_{i+1} = (Y_i \backslash \{u_i\})u_i^*$), on précisera qu'il s'agit de la procédure de Lazard à gauche (resp. à droite).

4.5 Équivalence entre ensembles de Lazard et ensembles de Hall

Dans la section précédente, un exemple des ensembles de Lazard à gauche est la famille $\mathcal{L}$ des mots de Lyndon tandis qu'un exemple d'ensembles de Lazard à droite est la famille $\mathcal{N}$ des mots de Nyldon (ce dernier sera montré plus tard dans ce mémoire) . De plus, suite à l'étude des ensembles de Hall, en particulier le théorème 4.38, il est trivial d'affirmer que la famille $\mathcal{L}$ des mots de Lyndon forme un ensemble de Hall. Suite à ce constat, une question naturelle est de savoir si la famille $\mathcal{N}$ des mots de Nyldon forme également un ensemble de Hall. En effet, $\mathcal{N}$ respecte quelques propriétés relatives aux mots de Hall dont, entre autre, le corollaire 4.28, le théorème 4.19, le fait que le dernier terme de la factorisation croissante en mots de Nyldon soit de longueur maximal. Plus généralement, la question est de savoir s'il existe une équivalence entre les ensembles de Lazard et les ensembles de Hall auquel cas il est possible d'affirmer que l'ensemble $\mathcal{N}$ des mots de Nyldon est un ensemble de Hall.

4.5.1 Lien entre les ensembles de Hall de Mélançon et de Viennot

Lors du début de ce chapitre, dans la partie concernant les ensembles de Hall, il a bien été spécifié que les définitions utilisées provenaient de l'article [13] de Mélançon. Cependant, les résultats permettant de prouver l'équivalence entre les ensembles de Hall et les ensembles de Lazard proviennent de l'article [18] de Gérard Viennot qui adopte d'autres définitions que celles de Mélançon dont une autre définition des ensembles de Hall. Pour être cohérent avec les raisonnements futurs, il semble opportun de redéfinir les notions de magma, de longueur.

Définition 4.45. Soit $\mathcal{A}$ un alphabet.

1. Notons $M(\mathcal{A})$ l'ensemble des mots parenthésés muni de la loi de composition interne $\cdot$ défini comme suit :

$$\forall u, v \in M(\mathcal{A}),\ u \cdot v = (u, v) \in M(\mathcal{A})$$

   $(M(\mathcal{A}), \cdot)$ est donc un magma.
2. Si h est un élément de $M(\mathcal{A})$, alors la longueur (ou degré) de h, noté $|h|$, est défini récursivement de la manière suivante :
   — Si $h \in \mathcal{A}$, alors $|h| = 1$.
   — Si $h \in M(\mathcal{A}) \setminus \mathcal{A}$, alors $h = (u, v)$ et $|h| = |u| + |v|$.

Remarque 4.46. Afin de ne pas être confondu avec le magma décrit dans la définition 4.1, le magma défini ci-dessus sera noté $M_{\mathcal{A}}$.

Grâce à ces réajustements, la définition des ensembles de Hall donnée par Viennot devient claire.

Définition 4.47. (Ensembles de Hall par G.Viennot) Soit H un sous-ensemble du magma $M_{\mathcal{A}}$ totalement ordonné par $<$. Alors H est un ensemble de Hall si

Ha_1 : $\mathcal{A} \subset H$

Ha_2 : $\forall h \in M_{\mathcal{A}} \backslash \mathcal{A}$, $h = (u, v) \in H$ si et seulement si les trois conditions suivantes sont respectées

1. $u, v \in H$
2. $u < v$
3. $v \in \mathcal{A}$ ou $v = (v', v'')$ avec $v' \leq u$.

Ha_3 : $\forall u, v \in H$, $(u, v) \in H \Rightarrow u < (u, v)$

Outre le fait que le magma considéré n'est pas le même, deux différences entre la définition 4.3 et cette définition 4.47 semblent rendre ces deux définitions totalement différentes. Tout d'abord, concernant les magmas de ces deux définitions, il est possible de les ramener au monoïde $\mathcal{A}^*$ grâce au feuillage (pour le magma $M(\mathcal{A})$) et de l'application $\phi : M(\mathcal{A}) \to \mathcal{A}^*, (u, v) \mapsto uv$ (pour le magma $M_{\mathcal{A}}$). Ensuite, les deux différences concernent les conditions $Ha_2.3$ et Ha_3. En effet, dans la définition de Mélançon, moyennant une réécriture pour être comparé à celle de Viennot, la condition $Ha_2.3$ est $u \in \mathcal{A}$ ou $u = (u', u'')$ avec $u'' \geq v$ et la condition Ha_3 devient $(u, v) \in H \Rightarrow (u, v) < v$ pour tout $u, v \in H$. Afin d'éviter toute ambiguïté, les ensembles de Hall définis par Mélançon seront appelés ensembles de Hall à droite et les ensembles de Hall définis par Viennot seront appelés ensembles de Hall à gauche. Mais, comme spécifie l'article [14], il est possible de passer d'une définition à l'autre en inversant l'ordre $<$ et en prenant les miroirs de chaque mot.

Remarque 4.48. Si $w = w_1 \cdots w_n \in \mathcal{A}^*$, alors le miroir de w, noté w^R, est le mot $w_n \cdots w_1$.

Remarque 4.49. En appliquant ce procédé à la définition 4.47, on obtient les changements suivants

— H est ordonné par $>$.

— $\forall h \in M(\mathcal{A}) \backslash \mathcal{A}$, $h = (v^R, u^R) \in H$ si et seulement si les trois conditions suivantes sont respectées

1. $u^R, v^R \in H$
2. $u^R > v^R$
3. $v^R \in \mathcal{A}$ ou $v^R = (v', v'')$ avec $v' \geq u^R$.

— $\forall u^R, v^R \in H, (v^R, u^R) \in H \Rightarrow (v^R, u^R) < u^R$.

En posant $X = v^R$ et $Y = u^R$, on réobtient la définition 4.3 des ensembles de Hall de Mélançon.

4.5.2 Équivalence entre ensembles de Lazard et ensembles de Hall

Le but de cette partie est de montrer que les ensembles de Lazard à gauche et les ensembles de Hall à gauche sont équivalents. Tout d'abord, afin de démontrer cette équivalence, il semble opportun d'introduire la définition de parenthésage, de déparenthésage et de projection sur le premier facteur d'un élément d'un ensemble de Lazard F.

Définition 4.50. Soit F un ensemble de Lazard.

1. L'opération de parenthésage $\Pi : F \to M_{\mathcal{A}}$ est défini comme suit

$$\Pi(x) = \begin{cases} x & \text{si } x \in \mathcal{A} \\ (\Pi(y), (\Pi(y), \ldots, (\Pi(y), \Pi(z))) \ldots) & \text{si } x \notin \mathcal{A} \text{ et } x = y^n z \end{cases}$$

2. L'opération de déparenthésage $\delta : M_{\mathcal{A}} \to \mathcal{A}^*$ est défini par

$$\delta(h) = \begin{cases} h & \text{si } h \in \mathcal{A} \\ \delta(u) \cdot \delta(v) & \text{si } h = (u, v) \in M_{\mathcal{A}} \backslash \mathcal{A} \end{cases}$$

où l'opération $\cdot$ est l'opération de concaténation du monoïde $\mathcal{A}^*$.

3. Si $h = (u, v) \in M_{\mathcal{A}} \backslash \mathcal{A}$, alors $p_1(h)$ est le premier facteur de h et est égal à u.

Grâce à ces différentes notions, il est possible de démontrer la première partie de l'équivalence entre les ensembles de Hall à gauche et de Lazard à gauche.

Proposition 4.51. *Soit F un ensemble de Lazard à gauche ordonné par $<$ et Π le parenthésage de F dans $M_{\mathcal{A}}$. Alors $\Pi(F)$ est un ensemble de Hall à gauche.*

Démonstration. Posons $H = \Pi(F)$, ordonné par l'ordre correspondant à F, i.e., pour tout $x, y \in F$, $x < y \Rightarrow \Pi(x) < \Pi(y)$. Soient $n \geq 1$ et $F \cap \mathcal{A}^{\leq n} = \{u_1, \ldots, u_{k+1}\}$ tel que $u_1 < u_2 < \ldots < u_{k+1}$. Soient $Y_0, \ldots, Y_k$ les ensembles apparaissant dans la définition 4.43. Dans ce cas, $Y_0 = \mathcal{A}, Y_1 = u_1^*(Y_0 \backslash \{u_1\}), \ldots, Y_k = u_k^*(Y_{k-1} \backslash \{u_k\})$. Pour tout élément w de $F \cap \mathcal{A}^{\leq n}$, on définit $\theta(w)$ comme étant le plus petit entier p tel que $w \in Y_p$. Cette fonction θ vérifie les propriétés suivantes :

$$\forall p \in [1, k+1]\,,\ \theta(u_p) < p. \tag{1}$$

$$\forall h \in \Pi(F \cap \mathcal{A}^{\leq n}),\ \delta(p_1(h)) = u_{\theta(\delta(h))}. \tag{2}$$

$$\forall p, q \in [1, k+1]\,,\ \theta(u_q) \leq p < q \Rightarrow u_p u_q \in F \text{ et } \Pi(u_p u_q) = (\Pi(u_p), \Pi(u_q)). \tag{3}$$

Tout d'abord, par définition de la méthode de construction des ensembles de Lazard et comme $Y_k \cap \mathcal{A}^{\leq n} = \{u_k\}$, l'alphabet $\mathcal{A}$ est bien inclus dans H et donc H respecte bien la condition Ha_1. Ensuite, soit $h = (u, v) \in H \backslash \mathcal{A}$ tel que $|h| \leq n$ et $p = \theta(h)$. Par définition de H, on a

$$u, v \in H \tag{4}$$

et on peut écrire $\delta(h) = u_p^i w$ avec $i \geq 1$, $\delta(u) = u_p \in Y_{p-1}$, $w \in Y_{p-1} \backslash \{u_p\}$. Pour tout $j \geq 1$ tel que $|u_p^j w| \leq n$, on a $u_p < u_p^j w$ par définition de la procédure de Lazard à gauche de longueur n. En particulier, $u_p < u_p^{i-1} w = \delta(v)$. Dès lors, on obtient

$$u < h \text{ et } u < v. \tag{5}$$

De plus, si $v \notin \mathcal{A}$. Si $i \geq 2$, alors $p_1(v) = \Pi(u_p) = u$. Sinon $i = 1$ et $v = \Pi(w) \in \Pi(Y_{p-1})$. Soit $q = \theta(\delta(v))$ et, par définition de θ, $q \leq p - 1$. Cependant, d'après (2), $\delta(p_1(v)) = u_q$ et donc

$$\delta(p_1(v)) < u. \tag{6}$$

Maintenant, prenons $h = (u, v) \in M_{\mathcal{A}} \backslash \mathcal{A}$ tel que $u, v \in H$, $u < v$ et $v \in \mathcal{A}$ ou $p_1 v \leq u$. Notons $\delta(u) = u_p$ et $\delta(v) = v_q$.

- Si $v_q \in \mathcal{A}$, alors $\theta(v_q) = 0$ et $h \in H$ par (3).
- Si $v_q \notin \mathcal{A}$, notons $\delta(p_1(v)) = u_r$ avec $r \leq p$. Par (2), $r = \theta(v_q)$ et donc $h \in H$ par (3).

Dès lors, si h appartient soit à $M_{\mathcal{A}} \backslash \mathcal{A}$ soit à $H \backslash \mathcal{A}$, alors $h \in H$. Les relations (4), (5) et (6) prouvent que H respecte la condition Ha_2. Enfin, la relation (5) démontre que H respecte Ha_3. Au final, H est bien un ensemble de Hall à gauche. □

De même, il est possible de démontrer la deuxième partie de l'équivalence annoncée, à savoir le fait que les ensembles de Hall à gauche sont équivalents aux ensembles de Lazard à gauche. Le résultat ci-dessous démontre même quelque chose de plus fort.

Proposition 4.52. *Soit $H \subset M_{\mathcal{A}}$ un ensemble de Hall à gauche totalement ordonné par $<$. La restriction à H de l'application de déparenthésage δ est injective et $F = \delta(H)$ ordonné par l'ordre correspondant à H, est un ensemble de Lazard à gauche. De plus, le parenthésage $\Pi : F \to M_{\mathcal{A}}$ est la bijection réciproque de la restriction de δ à H.*

Démonstration. Soient $H \subset M_{\mathcal{A}}$ un ensemble de Hall, $F = \delta(H)$ et $n \in \mathbb{N}$. Soit $\{h_1, \ldots, h_{k+1}\}$ l'ensemble des éléments de H de longueur inférieure ou égale à n ordonné par l'ordre induit de $H : h_1 < h_2 < \ldots < h_{k+1}$. Par définition de H, on a $h_1 \in H$. En effet, si $h_1 \notin$, alors $h_1 = (h_i, h_j)$ avec $i, j \in \{2, \ldots, k+1\}$. Or, par le point Ha_3 de H, $h_i < h_1$ ce qui est une contradiction. Notons donc $u_1 = h_1$ et $Y_1 = u_1^*(\mathcal{A} \backslash \{u_1\})$. Soit $\Pi_1 : \{u_1\} \cup Y_1 \to M_{\mathcal{A}}$ défini par récurrence :

$$\Pi_1(u) = \begin{cases} x & \text{si } x \in \mathcal{A} \\ (u_1, \Pi_1(u_1^{i-1}y)) & \text{si } u = u_1^i y \text{ ou } i \geq 1, y \in \mathcal{A} \backslash \{u_1\} \end{cases}$$

Alors, il est possible de montrer que

(R_1') $\Pi_1(Y_1) \subseteq H$

(R_1'') $\forall h = (h_1, h'') \in M_{\mathcal{A}}$, $h \in H \Rightarrow (\delta(h) \in Y_1$ et $h = \Pi_1(\delta(h)))$.

En effet, procédons par récurrence sur les longueurs des éléments y de $\Pi_1(Y_1)$ pour montrer (R_1'). Si $|y| = 1$, alors $y \in \mathcal{A}$ et $\Pi_1(y) = y \in H$ par définition de Π_1 et de H. Supposons maintenant que tous éléments $y \in \Pi_1(Y_1)$ de longueur inférieure ou égale à $m \geq 2$ soient dans H et montrons que tous les éléments $y \in \Pi_1(Y_1)$ de longueur m sont dans H. Si $y \in \Pi_1(Y_1)$ et $|y| = m$, alors $y = u_1^{m-1}w$ avec $p \geq 1$ et $w \in \mathcal{A} \backslash \{u_1\}$. Par définition de Π_1, on obtient que $\Pi_1(y) = (\Pi_1(u_1), \Pi_1(u_1^{p-1}w)$. Alors, par hypothèse de récurrence, on obtient $\Pi_1(u_1^{p-1}w) \in H$, $\Pi_1(u_1) = u_1 \in H$, $u_1 < \Pi_1(u_1^{p-1}w)$, $\Pi_1(u_1^{p-1}w) = (u_1, \Pi_1(u_1^{p-2}w))$ avec $u_1 \leq u_1$ et $u_1 < (u_1, \Pi_1(u_1^{p-1}w))$. Donc, $y \in H$. En ce qui concerne (R_1''), il est trivial de démontrer cette affirmation. Maintenant, considérons un naturel p tel que $1 \leq p < k$ et supposons que les trois conditions suivantes sont vérifiées.

(R_p) $u_1 = \delta(h_1), \ldots, u_p = \delta(h_p)$ sont des éléments distincts tels que l'on puisse écrire

$$u_2 \in Y_1,\ u_3 \in Y_2 = u_2^*(Y_1 \backslash \{u_2\}), \ldots,\ u_p \in Y_{p-1} = u_{p-1}^*(Y_{p-2} \backslash \{u_{p-1}\}).$$

Notons $Y_p = u_p^*(Y_{p-1} \backslash \{u_p\})$. On peut définir par récurrence une application $\Pi_p : \{u_1\} \cup \ldots \cup \{u_p\} \cup Y_p \to M_{\mathcal{A}}$, qui coïncide avec Π_{p-1} sur $\{u_1\} \cup \ldots \cup \{u_{p-1}\} \cup Y_{p-1}$ et telle que, pour $u = u_p^i y$ avec $y \in Y_{p-1} \backslash \{u_p\}$, $i \geq 1$,$\Pi_p(u) = (\Pi_p(u_p), \Pi_p(u_p^{i-1}y))$.

$(R_p^{'})$ $\Pi_p(u_1) = h_1, \ldots, \Pi_p(u_p) = h_p, \Pi_p(Y_p) \subseteq H$.

$(R_p^{''})$ $\forall i \in \{1, \ldots, p\},\ \forall h = (h_i, h'') \in M_{\mathcal{A}},\ h \in H \Rightarrow \delta(h) \in \{u_{i+1}\} \cup \ldots \cup \{u_p\} \cup Y_p$ et $h = \Pi_p(\delta(h))$.

Maintenant, si $h_{p+1} \in \mathcal{A}$, alors (R_p) prouve que $h_{p+1} \in Y_p$. En effet, cela se déduit de l'ordre des éléments de H. Si $h_{p+1} \notin \mathcal{A}$, alors on peut écrire $h_{p+1} = (h', h'')$ avec $h', h'' \in H$. À l'aide de la condition Ha_3 que suit H, $h' = h_i$ pour un $i \in \{1, \ldots, p\}$. Alors, $(R_p^{'})$ et $(R_p^{''})$ impliquent $\delta(h_{p+1}) = u_{p+1} \in Y_p$ et $h_{p+1} = \Pi_p(u_{p+1})$. Soient $Y_{p+1} = u_{p+1}^*(Y_p \backslash \{u_{p+1}\})$ et $\Pi_{p+1} : \{u_1\} \cup \ldots \cup \{u_p\} \cup \{u_{p+1}\} \cup Y_{p+1} \to M_{\mathcal{A}}$ défini de la même façon que Π_p. On a bien $\Pi_{p+1}(u_{p+1}) = h_{p+1}$. Grâce à une récurrence sur les longueurs des éléments de Y_{p+1} et à l'aide des conditions (R_p) et Ha_2, il est possible de montrer $\Pi_{Y_{p+1}} \subseteq H$. Donc, on a $(R_{p+1}^{'})$ qui est équivalent à $(R_p^{'})$ (en remplaçant p par $p+1$). D'autre part, soient $i \in \{1, \ldots, p+1\}$ et $h = (h_i, h'') \in H$. Si $i \leq p$, alors, d'après $(R_p^{''})$, h vérifie $(R_{p+1}^{'})$. Si $i = p+1$, une récurrence sur la longueur de h'' démontre, avec les conditions $Ha_1, Ha_2, Ha_3, R_p^{'}, R_p^{''}$, que $\delta(h) \in \{u_{p+1}\} \cup Y_{p+1}$ et $h = \Pi_{p+1}(\delta(h))$. Ainsi les trois conditions $(R_{p+1}), (R_{p+1}^{'})$ et $(R_{p+1}^{''})$ sont vérifiées. En effectuant une récurrence sur p, le résultat est démontré. □

Dès lors, grâce à ces deux propositions, il existe bel et bien une équivalence entre les ensembles de Lazard à gauche et les ensembles de Hall à gauche.

Théorème 4.53. *Les ensembles de Lazard à gauche sont en bijection avec les ensembles de Hall à gauche.*

Démonstration. Définissons la bijection $\phi : H \to F$ telle que $\phi(H) = \delta(H)$ et $\phi^{-1}(F) = \Pi(F)$. On conclut que ϕ est bien une bijection satisfaisant l'énoncé grâce aux propositions 4.51 et 4.52. □

Ce théorème lie les ensembles de Lazard à gauche et les ensembles de Hall. Cependant, il est possible de lier les ensembles de Lazard à droite et les ensembles de Hall à droite en prenant l'ordre $<$ et où chaque mot u est remplacé par u^R comme expliqué dans la remarque 4.49.

Théorème 4.54. *Les ensembles de Lazard à droite sont en bijection avec les ensembles de Hall à droite.*

Démonstration. En inversant l'ordre $<$ et en prenant les miroirs de chaque mot, on obtient une bijection ϕ défini par $\phi(H) = f(H)$ où f est l'application de feuillage des arbres binaires de $M(\mathcal{A})$ et $\phi^{-1}(F) = \Pi(F)$. De plus, cette bijection satisfait le théorème grâce aux propositions 4.51 et 4.52. □

Remarque 4.55. En ce qui concerne l'équivalence entre les ensembles de Lazard à droite et les ensembles de Hall à droite, l'application de parenthésage correspond au crochet utilisé dans la définition 4.3. De plus, la définition de ϕ dans le théorème 4.54 n'utilise pas l'application δ. Cependant, il est trivial de remarquer que les applications f et δ font la même chose, à savoir enlever la structures d'arbres ou de mots parenthésés afin d'avoir un élément du monoïde $\mathcal{A}^*$. Dès lors, dans la suite de ce chapitre, pour éviter toutes confusions possibles, ϕ_G désignera la bijection permettant de passer des ensembles de Hall à gauche aux ensembles de Lazard à gauche tandis que ϕ_D désignera la bijection permettant de passer des ensembles de Hall à droite aux ensembles de Lazard à droite.

4.6 Ensembles de Lazard et mots de Nyldon

Dans le chapitre précédent, nous avons introduit la notion de factorisation complète du monoïde $\mathcal{A}^*$ (cf. définition 3.13). Il se trouve que, grâce aux théorèmes 4.53 et 4.54, il existe un lien entre la notion de factorisation complète du monoïde $\mathcal{A}^*$ et les ensembles de Lazard.

Théorème 4.56. *Tout ensemble de Lazard à gauche (resp. à droite) muni d'un ordre total $\prec$ est une factorisation complète du monoïde libre $\mathcal{A}^*$ muni de ce même ordre total $\prec$.*

Démonstration. Considérons d'abord le cas des ensembles de Lazard et de Hall à gauche. Soit $H = \phi_G^{-1}(F)$ un ensemble de Hall ordonné par l'ordre $<$ induit par celui de F. Rappelons que tous les éléments de H sont des mots parenthésés. Grâce à l'application de déparenthésage δ et au théorème 4.19, l'ensemble $\delta(H)$ muni de l'ordre $<$ forme, par la définition 3.13, une factorisation complète du monoïde libre $\mathcal{A}^*$. Or, $\delta(H) = \phi_G(H) = \phi_G(\phi_G^{-1}(F)) = F$ par la proposition 4.52. Dès lors, l'ensemble F muni de l'ordre $<$ est une factorisation complète du monoïde libre $\mathcal{A}^*$. Démontrons maintenant le théorème pour les ensembles de Lazard et de Hall à droite. Soit $H = \phi_D^{-1}(F)$ un ensemble de Hall à droite ordonné par l'ordre $<$ induit par celui de F. Rappelons que, dans ce cas-ci, tous les éléments de H sont des arbres binaires respectant la définition 4.3. Grâce à l'application de feuillage f restreint à H et au théorème 4.19, l'ensemble $f(H)$ muni de l'ordre $<$ forme, par la définition 3.13, une factorisation complète du monoïde libre $\mathcal{A}^*$. Or, $f(H) = \phi_D(\phi_D^{-1}(F)) = F$ par la proposition 4.52. Au final, l'ensemble F muni de l'ordre $<$ est une factorisation complète du monoïde libre $\mathcal{A}^*$. □

Cependant, toutes les factorisations complètes de $\mathcal{A}^*$ ne sont pas des ensembles de Lazard à gauche et/ou à droite. Le théorème ci-dessous donne une caractérisation des ensembles de Lazard parmi les factorisations complètes.

Théorème 4.57. *Soit F une factorisation complète de $\mathcal{A}^*$ muni d'un ordre total $\prec$. Alors F est un ensemble de Lazard à gauche (resp. à droite) muni de l'ordre $\prec$ si et seulement si, pour tout $f, g \in F$, $fg \in F$ implique que $f \prec fg$ (resp. $g \succ fg$).*

Démonstration. Considérons d'abord le cas des ensembles de Lazard à gauche. Si F est un ensemble de Lazard à gauche, alors $\Pi(F)$ est aussi est un ensemble de Hall à gauche. De plus, $\delta(\Pi(F)) = F$ est l'ensemble des mots de Hall à gauche. De ce fait, si $f, g \in F$ et $fg \in F$, alors il est trivial de montrer que $f \prec fg$ (par la définition 4.45 et celle de l'application δ). Supposons maintenant que $f, g \in F$ et, si $fg \in F$, alors $f \prec fg$. Montrons par récurrence sur n que F vérifie la définition 4.43. Pour ce faire, appelons (La'_n) et (La''_n) les conditions de la définition 4.43.

(La'_n) $F \cap \mathcal{A}^{\leq n} = \{u_1, \ldots, u_{k+1}\}$ avec $u_1 \prec u_2 \prec \ldots \prec u_{k+1}$.

(La''_n) $u_1 \in \mathcal{A} = Y_0,\ u_2 \in Y_1 = u_1^*(\mathcal{A} \setminus \{u_1\}),\ \ldots,\ u_{k+1} \in Y_k = u_k^*(Y_{k-1} \setminus \{u_k\})$.

L'ensemble F vérifie trivialement les conditions (La'_1) et (La''_1). Supposons maintenant que F respecte les conditions (La'_m) et (La''_m) pour tout $m \leq n$ et montrons que F respecte les conditions (La'_{n+1}) et (La''_{n+1}). Remarquons que tout mot $w \in \mathcal{A}^+$ peut se factoriser de manière unique sous la forme

$$w = uv_{k+1}v_k \cdots v_1 \tag{I}$$

avec $v_1 \in \{u_1\}^*, \ldots, v_{k+1} \in \{u_{k+1}\}^*, u \in Y_{k+1}^*$ où $Y_{k+1} = u_{k+1}^*(Y_k \backslash \{u_{k+1}\})$. Montrons maintenant que $Y_{k+1} \cap \mathcal{A}^{n+1} = F \cap \mathcal{A}^{n+1}$. Soient $w \in Y_{k+1} \cap \mathcal{A}^{n+1}$ et $Fac(w) = (w_1, \ldots, w_p)$ le $p-$uple tel que $w = w_1 \cdots w_p$ avec $p \geq 1$, $w_i \in F$ pour tout $1 \leq i \leq p$ et $w_1 \succeq \ldots \succeq w_p$. Ce $p-$uple est unique car F est une factorisation complète de $\mathcal{A}^*$. Si $p > 1$, alors, d'après la condition (La_n''), tous les w_i sont des éléments de $\{u_1, \ldots, u_{k+1}\}$ et w admet donc deux factorisations de la forme (I) : $w = w$ et $w = w_1 \cdots w_p$. Dès lors, $p = 1$ et $w \in F \cap \mathcal{A}^{n+1}$. Supposons désormais que $w \in F \cap \mathcal{A}^{n+1}$. Alors, dans la factorisation (I), on a $u \neq \epsilon$. Dans le cas contraire, w aurait deux factorisations différentes dans F ce qui contredit l'hypothèse sur F. De plus, par la condition (La_n''), on obtient $|u| = n+1$, d'où $w = u$ et $w \in Y_{k+1} \cap \mathcal{A}^{n+1}$. Au final, on a bien prouvé que

$$Y_{k+1} \cap \mathcal{A}^{n+1} = F \cap \mathcal{A}^{n+1} \tag{II}$$

Comme $fg \in F$ implique $f \prec fg$ pour tout $f, g \in F$ et par (La_n'), il vient que

$$\forall p \in \{1, \ldots, k\},\ \forall w \in Y_p \cap \mathcal{A}^{n+1},\ u_p \prec w \tag{III}$$

Grâce aux conditions (II) et (III), on peut écrire

$$F \cap \mathcal{A}^{n+1} = \{u_1, v_{a_1}, v_{b_1}, \ldots, v_{j_1}, u_2, v_{a_2}, \ldots, u_{k+1}, v_{a_{k+1}}, \ldots, v_{j_{k+1}}\} \tag{IV}$$

ordonné dans cet ordre avec les conditions supplémentaires

$$v_{a_1}, \ldots, v_{j_1} \in Y_1 \cap \mathcal{A}^{n+1}, \ldots, v_{a_{k+1}}, \ldots, v_{j_{k+1}} \in Y_{k+1} \cap \mathcal{A}^{n+1}\,. \tag{V}$$

Dès lors, grâce à (La_n') et (V), la suite ordonnée des éléments du second membre de (IV) vérifie (La_{n+1}'). De plus, d'après (La_n'') et (II), la condition (La_{n+1}'') est également vérifiée. Dès lors, l'ensemble F est bien un ensemble de Lazard. Le cas des ensembles de Lazard à droite se traite de la même manière. □

De ces deux théorèmes, on en déduit la caractérisation des ensembles de Viennot suivante :

Corollaire 4.58. *Soit F une factorisation complète du monoïde libre $\mathcal{A}^*$ muni d'un ordre total $\prec$. Alors F est un ensemble de Viennot si et seulement si, pour tout $f, g \in F$, $fg \in F$ implique $f \prec fg \prec g$.*

Démonstration. Le résultat est trivial en utilisant les théorèmes 4.56 et 4.57. □

Remarque 4.59. Grâce à cette caractérisation, il est trivial de montrer que l'ensemble des mots de Lyndon muni de la relation d'ordre total $<_{\text{lex}}$ forme un ensemble de Viennot. En effet, si $\mathcal{L}$ est l'ensemble des mots de Lyndon, alors, pour tout $u, v \in \mathcal{L}$, on a $u <_{\text{lex}} uv <_{\text{lex}} v$ car la première inégalité provient de la définition de l'ordre lexicographique et la seconde provient d'une des propriétés des mots de Lyndon.

Après s'être penché sur le cas des mots de Lyndon, un autre cas à considérer est celui des mots de Nyldon. Grâce aux différents raisonnements et résultats de ce chapitre, il est direct de remarquer que $\mathcal{N}$ est un ensemble de Lazard à droite mais n'est pas un ensemble de Lazard à gauche. En conséquence, l'ensemble des mots de Nyldon $\mathcal{N}$ n'est pas un ensemble de Viennot. De plus, comme annoncé dans le chapitre précédent, la famille $\mathcal{N}$ des mots de Nyldon forme une factorisation complète

du monoïde $\mathcal{A}^*$ (via le théorème 3.11 et la définition 3.13). Grâce au lien entre $\mathcal{N}$ et les ensembles de Lazard, il est possible de récupérer ce résultat (cf. théorème 4.56).

Montrons, d'abord, que l'ensemble $\mathcal{N}$ n'est pas un ensemble de Lazard à gauche. Pour ce faire, nous allons montrer ce résultat de deux manières différentes en utilisant la caractérisation des ensembles de Viennot du lemme 4.58 ou en l'omettant momentanément.

Proposition 4.60. *L'ensemble des mots de Nyldon $\mathcal{N}$ n'est pas un ensemble de Lazard à gauche.*

Démonstration. Démontrons le résultat de deux manières différentes.
Méthode 1 : Utilisation du lemme 4.58.
Procédons par l'absurde et supposons que $\mathcal{N}$ est un ensemble de Lazard à gauche muni d'un ordre $\prec$. Grâce aux théorèmes 3.11 et 4.56, nous savons que l'ordre $\prec$ sur $\mathcal{N}$ doit coïncider avec l'ordre lexicographique décroissant $>_{lex}$. Dès lors, grâce au théorème 4.57, pour tout $f, g \in \mathcal{N}$ tels que $fg \in \mathcal{N}$, on devrait avoir $f >_{lex} fg$ ce qui est une contradiction.
Méthode 2 : Omission du lemme 4.58.
Si $\mathcal{N}$ était un ensemble de Lazard à gauche, alors, par les théorèmes 3.11 et 4.56, on a directement que l'ordre $\prec$ sur $\mathcal{N}$ doit être $>_{lex}$. Montrons maintenant que l'ensemble des mots de Nyldon $\mathcal{N}$ ne peut pas être obtenu par la procédure de Lazard à gauche avec la relation d'ordre $>_{lex}$. Sans perte de généralité, on peut supposer que $\mathcal{A} = \{0, \dots, m\}$ avec $0 <_{\text{lex}} 1 <_{\text{lex}} \dots <_{\text{lex}} m$. Procédons par l'absurde et supposons que l'ensemble des mots de Nyldon $\mathcal{N}$ peut être produit par la procédure de Lazard à gauche avec l'ordre lexicographique décroissant. Dès lors, pour toute longueur n, on a $Y_1 = \mathcal{A}$ et $u_1 = \min_{>_{lex}} Y_1 = \max_{<_{\text{lex}}} Y_1 = m$. Donc, $Y_2 = m^*\{0, \dots, m-1\}$. Si $n \geq 3$, alors $mm0$ sera extrait à un moment dans la procédure. Cependant, $mm0$ n'est pas un mot de Nyldon par la proposition 3.4 d'où la contradiction. □

Maintenant, montrons que l'ensemble des mots de Nyldon $\mathcal{N}$ est un ensemble de Lazard à droite.

Théorème 4.61. *L'ensemble des mots de Nyldon muni de l'ordre lexicographique décroissant $\geq_{lex}$ forme un ensemble de Lazard à droite.*

Démonstration. D'une part, grâce à la définition 3.13 et au théorème 3.11, on a trivialement que l'ensemble des mots de Nyldon $\mathcal{N}$ muni de l'ordre $>_{lex}$ est une factorisation complète du monoïde $\mathcal{A}^*$.
D'autre part, grâce à la proposition 3.8, on sait que, si $w \in \mathcal{N}$ et $s \in \mathcal{N}$ est un suffixe propre de w, alors $s <_{\text{lex}} w$. En particulier, si f et g sont des mots de Nyldon tels que fg est un mot de Nyldon, alors $g <_{\text{lex}} fg$. Donc, par la caractérisation du théorème 4.57, on obtient directement que l'ensemble $\mathcal{N}$ des mots de Nyldon muni de la relation d'ordre $>_{\text{lex}}$ est un ensemble de Lazard à droite. □

Exemple 4.62. Tout comme l'exemple 4.44, construisons les mots de Nyldon via la procédure de Lazard à droite. Fixons la longueur $n = 6$ et appliquons la procédure de Lazard. En utilisant les mêmes conventions pour le tableau que l'exemple 4.44, on obtient le tableau 4.12. Ainsi, ce tableau est tel que $D_0 = \mathcal{A} = \{0, 1\}$ et $D_{i+1} = (D_i \backslash \{u_i\})u_i^*$ où, à chaque étape, $u_i = \max_{>_{lex}}(D_i \cap \{0,1\}^{\leq 6}) = \min_{<_{\text{lex}}}(D_i \cap \{0,1\}^{\leq 6})$.

i	$D_i \cap \mathcal{A}^{\leq 6}$	M_i
1	$\{0, 1\}$	0
2	$\{1, 10, 100, 1000, 10000, 100000\}$	1
3	$\{10, 100, 1000, 10000, 100000, 100001, 10001,$ $100011, 1001, 10011, 100111, 101, 1011, 10111, 101111\}$	10
4	$\{100, 1000, 10000, 100000, 100001, 10001, 100010, 100011,$ $1001, 10010, 10011, 100110, 100111, 101, 1011, 10110,$ $10111, 101110, 101111\}$	100
5	$\{1000, 10000, 100000, 100001, 10001, 100010, 100011,$ $1001, 10010, 10011, 100110, 100111, 101, 1011, 10110,$ $10110010111, 101110, 101111\}$	1000
6	$\{10000, 100000, 100001, 10001, 100010, 100011, 1001,$ $10010, 10011, 100110, 100, 111, 101, 1011,$ $10110, 101100, 10111, 101110, 101111\}$	10000
7	$\{100000, 100001, 10001, 100010, 100011, 1001, 10010,$ $10011, 100, 110, 100111, 101, 1011, 10110,$ $101100, 10111, 101110, 101111\}$	100000
8	$\{100001, 10001, 100010, 100011, 1001, 10010, 10011, 100110,$ $100111, 101, 1011, 10110, 101100, 10111, 101110, 101111\}$	100001
9	$\{10001, 100010, 100011, 1001, 10010, 10011, 100110,$ $100111, 101, 1011, 10110, 101100, 10111, 101110, 101111\}$	10001
10	$\{100010, 100011, 1001, 10010, 10011, 100110, 100111,$ $101, 1011, 10110, 101100, 10111, 101110, 101111\}$	100010
11	$\{100011, 1001, 10010, 10011, 100110, 100111, 101, 1011, 10110,$ $101100, 10111, 101110, 101111\}$	100011
12	$\{1001, 10010, 10011, 100110, 100111, 101, 1011, 10110, 101100,$ $10111, 101110, 101111\}$	1001
13	$\{10010, 10011, 100110, 100111, 101, 1011, 10110, 101100, 10111,$ $101110, 101111\}$	10010
14	$\{10011, 100110, 100111, 101, 1011, 10110, 101100, 10111, 101110,$ $101111\}$	10011
15	$\{100110, 100111, 101, 1011, 10110, 101100, 10111, 101110, 101111\}$	100110
16	$\{100111, 101, 1011, 10110, 101100, 10111, 101110, 101111\}$	100111
17	$\{101, 1011, 10110, 101100, 10111, 101110, 101111\}$	101
18	$\{1011, 10110, 101100, 10111, 101110, 101111\}$	1011
19	$\{10110, 101100, 10111, 101110, 101111\}$	10110
20	$\{101100, 10111, 101110, 101111\}$	101100
21	$\{10111, 101110, 101111\}$	10111
22	$\{101110, 101111\}$	101110
23	$\{101111\}$	101111

FIGURE 4.12 – Extraction des mots de Nyldon

Comme l'ensemble des mots de Nyldon est un ensemble de Lazard à droite, une question naturelle est de savoir quand un mot w donné est généré par l'algorithme. Pour répondre à cela, on définit la procédure de Lazard de longueur n comme la procédure de Lazard de la définition 4.43 mais où l'on fixe la longueur n. Il est évident que la procédure de Lazard dont il est question est celle à droite. Dès lors, la question ci-dessus devient : si w est un mot de Nyldon, à partir de quelle longueur n, w est produit par la procédure de Lazard ? Pour répondre à cette question, Swapnil ([17]) montre que tous les mots de Nyldon sont générés par la procédure de Lazard de longueur n pour autant que n soit "suffisamment grand".

Lemme 4.63. *Tout mot de Nyldon w est généré de manière unique par la procédure de Lazard de longueur n avec $n \geq |w|$. De plus, cette génération est la même pour $n \geq |w|$.*

Démonstration. Appliquons la procédure de Lazard sur un alphabet $\mathcal{A}$ quelconque. De manière générale, tout mot v dans un des Y_i de la procédure a été généré en concaténant des mots de Nyldon de manière croissante comme suffixe à une lettre de Y_1 comme le stipule la définition 4.43. De plus, comme l'ensemble des mots de Nyldon forme un ensemble de Lazard à droite, tout mot de Nyldon w apparait comme un mot d'un certain Y_i (pour autant que l'on exécute la procédure pour une longueur $n \geq |w|$). Dès lors, tout mot de Nyldon de longueur supérieure ou égale à 2 peut s'écrire comme $au_{i_1}u_{i_2}\cdots u_{i_k}$ avec $a \in \mathcal{A}$ et $i_1 \leq i_2 \leq \ldots \leq i_k$ comme le décrit la procédure. Donc, $u_{i_1} \preceq u_{i_2} \preceq \ldots \preceq u_{i_k}$ sont des mots de Nyldon. De plus, par le théorème 3.11, $u_{i_1}u_{i_2}\cdots u_{i_k}$ est l'unique factorisation en mots de Nyldon de $a^{-1}w$ et donc il n'y a qu'une unique manière de générer w en exécutant la procédure de Lazard à droite (génération qui est la même pour tout n). Dès lors, le w sera dans $\mathcal{N} \cap \mathcal{A}^{\leq n}$ pour un $n \geq |w|$ et tous les mots u_{i_j} sont de longueur au plus n. Au final, le mot w est bien généré par la procédure de Lazard de longueur $n \geq |w|$. □

Dans le suite de cette section, nous allons travailler sur un alphabet $\mathcal{A}$ quelconque (pas forcément un alphabet binaire). Sans perte de généralité, si $\mathcal{A}$ est un alphabet de taille $m+1$, on pose $\mathcal{A} = \{0, 1, \ldots, m\}$ avec $0 < 1 < \ldots < m$.

Pour une certaine longueur l fixé, il est possible de trouver le mot de Nyldon de longueur au plus l maximal pour l'ordre lexicographique.

Lemme 4.64. *Le mot de Nyldon de longueur au plus l maximal pour l'ordre lexicographique est le mot $m(m-1)m^{l-2}$.*

Démonstration. Par la proposition 3.4, il est impossible d'avoir un mot de Nyldon commençant par mm. Donc, $m(m-1)m^{l-2}$ est le mot maximal pour l'ordre $<_{\text{lex}}$. En appliquant un algorithme de factorisation, on remarque que $m(m-1)m^{l-2}$ est, en fait, un mot de Nyldon. □

Lors de l'application de l'algorithme de Mélançon (défini par 4.36), pour chaque accroissement, les éléments de ce dernier sont concaténés avant de passer à l'étape suivant de l'algorithme. Le résultat de concaténation est appelé bloc. De plus, la longueur d'un bloc est défini par la longueur du mot contenu dans ce bloc. Par exemple, si, lors de l'exécution de l'algorithme de Mélançon, il se forme un bloc au avec $a \in \mathcal{A}$ et $u \in \mathcal{A}^*$, alors la longueur de ce bloc est $1 + |u|$. Grâce à cette notion de bloc et aux deux propositions suivantes, il est possible de déterminer la forme de plusieurs mots de Nyldon.

Proposition 4.65. *Pour tout naturel $l > 1$ et pour tout mot $v \in \mathcal{A}^*$, en exécutant l'algorithme de Mélançon sur $w = m(m-1)m^l v$, il y aura un bloc de longueur $l+1$ au début de w à une certaine étape de l'algorithme.*

Démonstration. Soit $w = m(m-1)m^{l-1}v$. Le but de cette démonstration est de montrer que le préfixe $u = m(m-1)m^{l-1}$ qui est un mot de Nyldon va former un bloc à une certaine étape de l'algorithme. Procédons par l'absurde et supposons que u n'apparaitra pas comme un bloc durant l'exécution de l'algorithme. Comme u est un mot de Nyldon, par le lemme 4.64, alors u formera un bloc à un moment donné de l'algorithme de Mélançon. En effet, la lettre $m-1$ est concaténé avec m puis les $l-1$ copies de m seront concaténés avec $m(m-1)$. La seule manière d'empêcher que u forme un bloc est que cette concaténation soit bloqué par mv. Plus spécifiquement, il faudrait que le bloc le plus à gauche de mv se concatène avec le bloc le plus à droite de u. Cependant, jusqu'à ce que u forme un bloc, son bloc le plus à droite est toujours la lettre m qui est aussi grand lexicographiquement que le block le plus à gauche de mv. Donc, on a une contradiction car u formera bien un bloc d'où le résultat. □

Proposition 4.66. *Soit l un naturel strictement positif. Si v est un mot de longueur au plus l, alors $w = m(m-1)m^{l+1}v$ est un mot de Nyldon.*

Démonstration. En exécutant l'algorithme de Mélançon sur w, à une certaine, on aura un bloc de longueur $l+2$ au début de w par la proposition 4.65. Ce bloc est plus grand lexicographiquement que tous les autres blocs sur la droite car la longueur de ces blocs est inférieure à $l+2$ (la longueur de w est au plus $2l+3$). De plus, ce premier bloc est le plus grand mot de Nyldon de longueur au plus $l+2$ par le lemme 4.64. Donc, w est bien un mot de Nyldon. □

Enfin, lors de l'exécution de la procédure de Lazard de longueur n, on remarque que les dernières étapes de l'algorithme prennent la forme d'éliminations successives d'élément. Dans le tableau de l'exemple 4.62, après l'extraction de l'élément 100 et la création du nouvel ensemble $D_5 \cap \mathcal{A}^{\leq 6}$, aucun autre mot sera créé. Dès lors, la question est de savoir à partir de quelle étape aucun autre mot ne sera créé. En d'autres termes, quel serait l'élément à extraire à partir duquel aucun autre mot de Nyldon de longueur inférieure ou égale à n ne serait créé. La proposition suivante permet de donner la réponse à cette question.

Proposition 4.67. *Soit un naturel $l \geq 9$ et supposons que l'étape à partir de laquelle tous les mots de Nyldon de longueur inférieure ou égale à l sont générés par la procédure de Lazard de longueur l corresponde à celle où le mot u_i est retiré pour un certain i. Si l est impair, alors $u_i = m(m-1)m^{\frac{l-5}{2}}$, si l est pair, alors $u_i = m(m-1)m^{\frac{l-6}{2}}(m-1)$.*

Démonstration. L'idée de cette démonstration est que le mot u_i doit être le mot de Nyldon lexicographiquement maximal tel qu'il soit encore possible de le concaténer à droite d'un autre mot de Nyldon tout en restant dans les limites imposées par la procédure (la longueur du mot ainsi créé doit être inférieure ou égale à l). Supposons maintenant qu'à l'étape où u_i est extrait et où l'ensemble des mots de Nyldon créé par la concaténation avec u_i est créé, il existe encore un mot de Nyldon qui n'est pas apparu. Dès lors, ce mot doit être égal à ab avec $u_i <_{\text{lex}} b <_{\text{lex}} a$. En effet, on a $b <_{\text{lex}} a$ par définition des mots de Nyldon et $u_i <_{\text{lex}} b$ car sinon ab devrait être apparu avant

l'extraction de u_i. Considérons le cas où l est impair. Dès lors, comme a et b doivent être lexicographiquement plus grand que u_i, alors ils doivent être de longueur au moins $\frac{l+1}{2}$ car u_i est le mot de Nyldon le plus grand lexicographiquement parmi ceux de longueur au plus $\frac{l-1}{2}$ par le lemme 4.64. Donc, on a $|ab| \geq l+1$ ce qui est une contradiction avec le fait que ab apparait dans la procédure de Lazard à droite de longueur l. Considérons maintenant le cas où l est pair. Dans ce cas, comme a et b sont lexicographiquement supérieur à u_i, ils doivent avoir une longueur d'au moins $\frac{l}{2}$ par le lemme 4.64. De plus, comme $|ab|$ doit être au plus l, la seule manière d'y arriver est que $a = b = u_i(m-1)^{-1}m = m(m-1)m^{\frac{l-6}{2}}m$ ce qui est impossible car a et b ne peuvent pas être égal. Donc, cela prouve que, lors de l'extraction de u_i, tous les mots de Nyldon de longueur inférieure ou égale à l ont été généré. De plus, lors de l'extraction de u_i, si l est impair, alors $(u_i m)u_i$ est généré. De même, si l est pair, alors $(u_i(m-1)^{-1}m)u_i)$ est généré. Ces deux mots sont de nouveaux mots de Nyldon ce qui prouve le résultat. □

Exemple 4.68. Grâce à ce résultat, il est donc possible de déterminer à partir de quel moment tous les mots de Nyldon seront créés. Dans le tableau 4.13, la première colonne contient la longueur l de la procédure de Lazard à droite utilisée tandis que la seconde colonne contient les mots u_i à partir desquels tous les mots de Nyldon sont générés pour chaque procédure de Lazard à droite de longueur l. Enfin, l'étape d'extraction des u_i figure dans la troisième colonne.

l	u_i	Étape
9	1011	76
10	10110	126
11	10111	302
12	101110	531
13	101111	1140
14	1011110	2067
15	1011111	4228

FIGURE 4.13 – Mot u_i de la proposition 4.67

4.7 Ordre d'un ensemble de Lazard et mots de Nyldon

Dans la définition de la procédure de Lazard (définition 4.43), la notion d'ordre sur F n'est pas défini proprement. De ce fait, il existe deux manières d'ordonner les éléments de F. Une première manière de faire est de disposer d'une relation d'ordre $\prec$ sur $\mathcal{A}^*$ et de la restreindre sur F. De ce fait, soit il existe un ensemble de Lazard à gauche et/ou à droite soit il n'y en a pas. Dans le cas où il existe un ensemble de Lazard à gauche (resp. à droite), pour toute longueur n, la procédure de Lazard à gauche (resp. à droite) va calculer les mots de longueur n de F les uns après les autres en les extrayant les mots de $Y_i \cap \mathcal{A}^{\leq n}$ dans l'ordre croissant (selon l'ordre préexistant de $\mathcal{A}^*$) jusqu'à obtenir le singleton $Y_k \cap \mathcal{A}^{\leq n}$. Un exemple de cette manière de procéder est l'exemple 4.44. En effet, les éléments de $\mathcal{A}$ sont ordonnés à

l'aide de l'ordre $<_{\text{lex}}$ et, lors de l'extraction dans les ensembles D_i et G_i, $<_{\text{lex}}$ est utilisé pour sélectionner l'élément à extraire. Une deuxième manière de faire stipule qu'il n'est pas nécessaire de disposer d'une relation d'ordre total $\prec$ sur $\mathcal{A}^*$ pour munir F d'un ordre. En effet, tout élément u_i extrait d'un ensemble Y_i après chaque étape va déterminer un ordre total sur F. Par exemple, si u_i est extrait avant u_{i+1} dans la procédure de Lazard à gauche, alors $u_i \prec u_{i+1}$. Dans le cas de la procédure de Lazard à droite, si u_i est extrait avant u_{i+1}, on aura $u_i \succ u_{i+1}$. En particulier, comme $\mathcal{A} = Y_1 \subset F$, ce procédé induit un ordre total sur l'alphabet $\mathcal{A}$. Cependant, aucune condition n'impose que l'ordre construit via cette deuxième manière faire ne s'étende en un ordre total sur $\mathcal{A}^*$.

Dans l'exemple 4.44, en sélectionnant l'élément minimal pour $<_{\text{lex}}$ dans les Y_i, la procédure de Lazard à gauche de longueur n permet de construire tous les mots de Lyndon jusque la longueur n (inclus). De manière similaire, sélectionner l'élément maximal pour $<_{\text{lex}}$ dans les Y_i de la procédure de Lazard à droite de longueur n permet également de retrouver tous les mots de Lyndon jusque la longueur n (inclus). En ce qui concerne les mots de Nyldon, comme le montre l'exemple 4.62, en prenant l'élément minimal pour $<_{\text{lex}}$ dans les Y_i, la procédure de Lazard à droite de longueur n permet de retrouver cette famille de mots dont la longueur est inférieure ou égale à n. Dans ces différents cas, le seul changement concerne l'extraction d'un élément u_i. Cependant, le type de procédure, l'ordre de l'alphabet et le choix d'extraction d'un élément u_i vont influencer sur la famille de mots obtenue à la fin de la procédure. Pour résumer le travail qui est fait dans la suite, il faut

- fixer la procédure de Lazard (à gauche ou à droite),
- fixer un ordre sur l'alphabet $\mathcal{A}$ donné,
- décider d'un élément extrême à extraire.

Dans la suite, le travail sera fait sur un alphabet $\mathcal{A} = \{0, 1, \ldots, m\}$.

4.7.1 Procédure de Lazard à gauche

Dans cette partie, le choix du type de procédure est la procédure de Lazard à gauche. Comme rappelé ci-dessus, $0 < 1 < \ldots < m$ et en extrayant l'élément minimal à chaque étape de la procédure, la famille des mots de Lyndon est construite. Que se passera-t-il si l'on extrait l'élément maximal pour l'ordre $<_{\text{lex}}$? En effectuant la procédure de Lazard à gauche, un ensemble de mots semblable à l'ensemble des mots de Lyndon est ainsi défini : l'ensemble $\overline{\mathcal{L}} = \{\overline{w} \in \{0, 1, \ldots, m\}^* : w \in \mathcal{L}\}$. De plus, $\overline{w}$ est défini par récurrence sur la longueur de w :

- Si $w \in \mathcal{A}$, alors $\overline{w} = m - w$.
- Si $|w| \geq 2$ et $w = u \cdot v$, alors $\overline{w} = \overline{u} \cdot \overline{v}$.

Exemple 4.69. En prenant les trois premiers mots de Lyndon par ordre croissant de longueur, à savoir $0, 1, 01$, on obtient que

1. $\overline{0} = 1$,
2. $\overline{1} = 0$,
3. $\overline{01} = 10$.

En ce qui concerne l'ordre sur $\overline{\mathcal{L}}$, la définition suivante permet de répondre à la question.

Définition 4.70. Soient $\mathcal{A} = \{0, 1, \ldots, m\}$ un alphabet et π une permutation agissant sur $\mathcal{A}$. Si $\prec$ est une relation d'ordre total sur $\mathcal{A}^*$ induit de l'ordre total $0 < 1 < \ldots < m$ sur les lettres, alors on note $\prec^\pi$ la relation d'ordre total induit par l'ordre total $\pi(0) < \pi(1) < \ldots < \pi(m)$ sur les lettres. De plus, on peut étendre la définition de π à $\mathcal{A}^+$ en posant $\pi(uv) = \pi(u)\pi(v)$ pour tout $u, v \in \mathcal{A}^+$. Enfin, si F est un sous-ensemble de $\mathcal{A}^+$, alors $\pi(F) = \{\pi(w) : w \in F\}$.

Remarque 4.71. L'ensemble $\overline{\mathcal{L}}$ possède une relation d'ordre total $\prec^\pi$ induit par $0 >_{\text{lex}} 1 >_{\text{lex}} \ldots >_{\text{lex}} m$ en posant $\pi(i) = \overline{i}$ pour tout $i \in \mathcal{A}$.

Remarque 4.72. Si π est l'identité, alors $\prec^\pi$ correspond à l'ordre $\prec$.

Dès lors, l'ensemble $\overline{\mathcal{L}}$ correspond donc à $\pi(\mathcal{L})$ où π est la permutation sur $\{0, 1, \ldots, m\}$ défini par $\pi(i) = m - i$ pour tout i. Il est possible de montrer que $\overline{\mathcal{L}}$ muni de la relation d'ordre $<^\pi_{lex}$ induit par $0 > 1 > \ldots > m$ est un ensemble de Viennot par la proposition suivante.

Proposition 4.73. *Soient $\mathcal{A} = \{0, 1, \ldots, m\}$ un alphabet et π une permutation agissant sur $\mathcal{A}$. Si un sous-ensemble F de $\mathcal{A}^+$ est un ensemble de Lazard à gauche (resp. à droite) muni d'une relation d'ordre total $\prec$ induit par l'ordre total $0 < 1 < \ldots < m$ sur les éléments de $\mathcal{A}$, alors l'ensemble $\pi(F)$ muni de l'ordre total $\prec^\pi$ est aussi un ensemble de Lazard à gauche (resp. à droite).*

Démonstration. Démontrons le résultat pour F un ensemble de Lazard à gauche. Le cas où F est un ensemble de Lazard à droite est symétrique. Par le théorème 4.56, l'ensemble F muni de l'ordre $\prec$ est une factorisation complète de $\mathcal{A}^*$. De plus, remarquons que, pour tout $u, v \in \mathcal{A}^+$, $u \prec v$ si et seulement si $\pi(u) \prec^\pi \pi(v)$. Ainsi, l'ensemble $\pi(F)$ muni de l'ordre $\prec^\pi$ est également une factorisation complète du monoïde $\mathcal{A}^*$. Montrons, à l'aide du théorème 4.57, que $\pi(F)$ est un ensemble de Lazard à gauche muni de l'ordre $\prec^\pi$. Soient $f, g \in \pi(F)$ tels que $fg \in \pi(F)$. On a donc $f = \pi(u)$ et $g = \pi(v)$ avec $u, v \in F$. Comme $\pi(uv) = fg \in \pi(F)$, on obtient que $uv \in F$ et que $u \prec uv$ car F est un ensemble de Lazard à gauche. Dès lors, $f \prec^\pi fg$. On conclut que $\pi(F)$ est un ensemble de Lazard à gauche par le théorème 4.57. □

Corollaire 4.74. *Soient $\mathcal{A} = \{0, 1, \ldots, m\}$ un alphabet et π une permutation agissant sur $\mathcal{A}$. Si un sous-ensemble F de $\mathcal{A}^+$ est un ensemble de Viennot muni de la relation d'ordre $\prec$ induit par l'ordre total $0 < 1 < \ldots < m$ sur les éléments de $\mathcal{A}$, alors l'ensemble $\pi(F)$ muni de l'ordre total $\prec^\pi$ est aussi un ensemble de Viennot*

Dès lors, par le corollaire 4.74, $\overline{\mathcal{L}}$ est un ensemble de Viennot muni de l'ordre total $<^\pi_{\text{lex}}$.

Exemple 4.75. Appliquons les procédures de Lazard à gauche et à droite pour illustrer le fait que $\overline{\mathcal{L}}$ est un ensemble de Viennot. Pour ce faire, nous illustrerons cette propriété avec un alphabet binaire $\mathcal{A} = \{0, 1\}$ et $n = 5$. Ainsi, posons $G_1 = D_1 = \{0, 1\}$. Ensuite, pour tout $i \geq 1$, $G_{i+1} = g_i^*(G_i \backslash \{g_i\})$ et $D_{i+1} = (D_i \backslash \{d_i\})d_i^*$ où $g_i = \min_{<^\pi_{\text{lex}}}(G_i \cap \{0, 1\}^{\leq 5})$ et $d_i = \max_{<^\pi_{\text{lex}}}(D_i \cap \{0, 1\}^{\leq 5})$. Après application des deux procédures, on obtient les deux tableaux 4.14 et 4.15.

i	$G_i \cap \mathcal{A}^{\leq 5}$	g_i
1	$\{0, 1, \}$	1
2	$\{0, 10, 110, 1110, 11110\}$	11110
3	$\{0, 10, 110, 1110\}$	1110
4	$\{0, 11100, 10, 110\}$	11100
5	$\{0, 10, 110\}$	110
6	$\{0, 1100, 10, 11010\}$	11010
7	$\{0, 1100, 10\}$	1100
8	$\{0, 11000, 10\}$	11000
9	$\{0, 10\}$	10
10	$\{0, 100, 10100\}$	10100
11	$\{0, 100\}$	100
12	$\{0, 1000\}$	1000
13	$\{0, 10000\}$	10000
14	$\{0\}$	0

FIGURE 4.14 – Procédure de Lazard à gauche pour trouver $\overline{\mathcal{L}}$

i	$D_i \cap \mathcal{A}^{\leq 5}$	d_i
1	$\{0, 1\}$	0
2	$\{1, 10, 100, 1000, 10000\}$	10000
3	$\{1, 10, 100, 1000\}$	1000
4	$\{1, 11000, 10, 100\}$	100
5	$\{1, 1100, 11000, 10, 10100\}$	10100
6	$\{1, 1100, 11000, 10\}$	10
7	$\{1, 110, 11010, 1100, 11000\}$	11010
8	$\{1, 110, 1100, 11000\}$	11000
9	$\{1, 110, 1100\}$	1100
10	$\{1, 11100, 110\}$	110
11	$\{1, 1110, 11100\}$	11100
12	$\{1, 1110\}$	1110
13	$\{1, 11110\}$	11110
14	$\{1\}$	1

FIGURE 4.15 – Procédure de Lazard à droite pour trouver $\overline{\mathcal{L}}$

L'extraction des g_i se fait dans l'ordre croissant pour $<^{\pi}_{\text{lex}}$, c'est-à-dire, $g_1 <^{\pi}_{\text{lex}} g_2 <^{\pi}_{\text{lex}} \dots <^{\pi}_{\text{lex}} g_{14}$ tandis que l'extraction des d_i se fait, quant à elle, par ordre décroissant pour $<^{\pi}_{\text{lex}}$, c'est-à-dire, $d_{14} <^{\pi}_{\text{lex}} \dots <^{\pi}_{\text{lex}} d_2 <^{\pi}_{\text{lex}} d_1$.
De plus, $g_i = \max_{<_{\text{lex}}}(G_i \cap \{0,1\}^{\leq 5})$. Cela ne semble point surprenant mais il n'est pas toujours vrai que $\max_{<_{\text{lex}}} S = \min_{<^{\pi}_{\text{lex}}} S$ si S est un sous-ensemble de $\mathcal{A}^*$. En effet, en regardant la construction des D_i, on constate que $\min_{<_{\text{lex}}}(D_2 \cap \{0,1\}^{\leq 5}) = 1 \neq \max_{<^{\pi}_{\text{lex}}}(D_2 \cap \{0,1\}^{\leq 5}) = d_2 = 10000$. Si $\min_{<_{\text{lex}}}(D_i \cap \{0,1\}^{\leq 5}) = \max_{<^{\pi}_{\text{lex}}}(D_i \cap \{0,1\}^{\leq 5})$ pour tout i, alors les mots produits par la procédure de Lazard à droite

serait les mots de Nyldon. Or, d_7 n'est pas un mot de Nyldon par la proposition 3.4. L'égalité

$$\max_{<_{\text{lex}}}(G_i \cap \{0,1\}^{\leq 5}) = \min_{<^{\pi}_{\text{lex}}}(G_i \cap \{0,1\}^{\leq 5}) \tag{1}$$

est vrai car les ensembles G_i sont dits sans préfixe pour tout i (tout élément de G_i n'est préfixe d'aucun autre élément de G_i).

Jusqu'à présent, le seul élément pris en compte est le choix de l'élément à extraire. Une autre question naturelle est de savoir si l'ordre sur $\mathcal{A}$ modifie la famille engendrée par la procédure de Lazard à gauche. Cependant, les différents choix d'extraction resteront les mêmes : l'élément maximal ou l'élément minimal pour un ordre donné.

D'une part, si $\mathcal{A} = \{0, 1, \ldots, m\}$ est un alphabet tel que $0 > 1 > \ldots > m$, en effectuant la procédure de Lazard à gauche en décidant d'extraire l'élément maximal, la procédure donne la famille des mots de Lyndon. Dans l'exemple ci-dessous, en effectuant la procédure de Lazard de longueur 4 à un alphabet binaire $\mathcal{A} = \{0, 1\}$ et en extrayant l'élément $g_i = \max_{\prec_{\text{lex}}}(G_i \cap \mathcal{A}^{\leq 4})$ où $\prec_{\text{lex}}$ est un ordre induit par l'ordre $0 > 1$, on obtient le tableau 4.16 :

i	$G_i \cap \mathcal{A}^{\leq 4}$	g_i
1	$\{0, 1\}$	0
2	$\{1, 01, 001, 0001\}$	0001
3	$\{1, 01, 001\}$	001
4	$\{1, 0011, 01\}$	0011
5	$\{1, 01\}$	01
6	$\{1, 011\}$	011
7	$\{1, 0111\}$	0111
8	$\{1\}$	1

FIGURE 4.16 – Mots de Lyndon et ensemble de Lazard

En appliquant la procédure de Lazard à gauche de longueur 4 comme décrite dans l'exemple 4.44, on remarque que les mots ainsi extraits sont les mêmes et dans le même ordre. Ce résultat est dû au fait que l'égalité (1) discutée ci-dessus dans le cadre de l'ensemble $\overline{\mathcal{L}}$ est vérifiée car les ensembles $G_i \cap \mathcal{A}^{\leq 4}$ sont dits sans préfixe. Dans ce cas, renverser l'ordre sur $\mathcal{A}$ et extraire l'élément maximal lors de la procédure de Lazard à gauche renvoit la famille des mots de Lyndon.

D'autre part, si $\mathcal{A} = \{0, \ldots, m\}$ est un alphabet tel que $0 > 1 > \ldots > m$, en effectuant la procédure de Lazard à gauche et en extrayant l'élément minimal, la procédure renvoit la famille $\overline{\mathcal{L}}$. Dans le tableau ci-dessous, en prenant l'alphabet binaire $\mathcal{A} = \{0, 1\}$ et en effectuant la procédure en extrayant $g_i = \min_{\prec_{\text{lex}}}$ où $\prec_{\text{lex}}$ est un ordre induit par $0 > 1$, on a le tableau 4.17

Les différentes situations peuvent être résumées par le tableau 4.18.

i	$G_i \cap \mathcal{A}^{\leq 4}$	g_i
1	$\{0, 1\}$	1
2	$\{0, 10, 110, 1110\}$	1110
3	$\{0, 10, 110\}$	110
4	$\{0, 1100, 10\}$	1100
5	$\{0, 10\}$	10
6	$\{0, 100\}$	100
7	$\{0, 1000\}$	1000
8	$\{0\}$	0

FIGURE 4.17 – Ensemble de Lazard et $\overline{\mathcal{L}}$

G	$0 < 1 < \ldots < m$	$0 > 1 > \ldots > m$
Élément minimal extrait	$\mathcal{L}$	$\overline{\mathcal{L}}$
Élément maximal extrait	$\overline{\mathcal{L}}$	$\mathcal{L}$

FIGURE 4.18 – Résumé des ensembles de Lazard à gauche

4.7.2 Procédure de Lazard à droite

Vu le travail effectué avec la procédure de Lazard à gauche, cette partie sera beaucoup plus rapide. De fait, il est déjà connu que les familles $\mathcal{L}$ et $\overline{\mathcal{L}}$ sont des ensembles de Viennot par le théorème 4.57 et le corollaire 4.74. Une instance de la procédure de Lazard à droite de longueur 6 a été faite dans l'exemple 4.44. En ce qui concerne l'ensemble $\overline{\mathcal{L}}$, en extrayant l'élément maximal d_i avec un alphabet $\mathcal{A} = \{0, 1\}$ tel que $0 > 1$, on obtient l'instance représentée par la figure 4.19 pour la procédure de Lazard de longueur 4 :

i	$D_i \cap \mathcal{A}^{\leq 4}$	d_i
1	$\{0, 1\}$	0
2	$\{1, 10, 100, 1000\}$	1000
3	$\{1, 10, 100\}$	100
4	$\{1, 1100, 10\}$	10
5	$\{1, 110, 1100\}$	1100
6	$\{1, 110\}$	110
7	$\{1, 1110\}$	1110
8	$\{1\}$	1

FIGURE 4.19 – Instance de la procédure de Lazard à droite de longueur 4.

Le choix de l'élément à extraire est imposé par la proposition 4.73.

Un autre résultat connu est le fait que l'ensemble $\mathcal{N}$ des mots de Nyldon est un ensemble de Lazard à droite par le théorème 4.61. Une instance de la procédure de

Lazard à droite pour retrouver les mots de Nyldon a été faite à l'exemple 4.62. Enfin, en utilisant la proposition 4.73, l'ensemble $\pi(\mathcal{N}) = \{\overline{w} : w \in \mathcal{N}\}$, plus communément noté $\overline{\mathcal{N}}$, est également un ensemble de Lazard à droite. Pour retrouver cet ensemble $\overline{\mathcal{N}}$, il suffit d'extraire l'élément minimal de chaque ensemble D_i avec un ordre $\prec$ induit par l'ordre $0 > 1 > \ldots > m$ sur $\mathcal{A}$. En appliquant la procédure de Lazard de longueur 4 sur un alphabet binaire $\mathcal{A} = \{0, 1\}$ tel que $0 > 1$, on obtient le tableau de la figure 4.20.

i	$D_i \cap \mathcal{A}^{\leq 4}$	d_i
1	$\{0, 1\}$	1
2	$\{0, 01, 011, 0111\}$	0
3	$\{01, 010, 0100, 011, 0110, 0111\}$	01
4	$\{010, 0100, 011, 0110, 0111\}$	011
5	$\{010, 0100, 0110, 0111\}$	0111
6	$\{010, 0100, 0110\}$	0110
7	$\{010, 0100\}$	010
8	$\{0100\}$	0100

FIGURE 4.20 – Ensemble de Lazard à droite et $\overline{\mathcal{N}}$

Le tableau de la figure 4.21 est un résumé des différentes possibilités lors de l'application de la procédure de Lazard à droite selon l'ordre de l'alphabet $\mathcal{A}$ et l'élément extrême extrait.

D	$0 < 1 < \ldots < m$	$0 > 1 > \ldots > m$
Élément minimal extrait	$\mathcal{N}$	$\overline{\mathcal{N}}$
Élément maximal extrait	$\mathcal{L}$	$\overline{\mathcal{L}}$

FIGURE 4.21 – Résumé des ensembles de Lazard à droite

Chapitre 5

Algorithmes de factorisation

Au travers des différents chapitres, plusieurs algorithmes ont été mis en place afin de répondre à des problématiques. Parmi ceux-ci, les algorithmes résultant du théorème de Chen-Fox-Lyndon ou l'algorithme de Mélançon. Dans ce chapitre, tout ces algorithmes seront présentés et leur complexité sera mise en avant. Pour être plus précis, la complexité dans le pire cas de chaque algorithme sera mise en avant afin de pouvoir raisonner dans le pire scénario possible. Une autre question à laquelle toute personne pourrait s'attaquer est de savoir s'il est possible d'optimiser ces algorithmes. Dès lors, les présentations ci-dessous utiliseront les conventions suivantes :

- La première lettre d'un mot w est noté w_1,
- Le sous-mot $w_i \cdots w_j$ sera noté $w[i : j]$. De plus, $w[: j]$ et $w[i :]$ désignent respectivement les sous-mots $w_1 \cdots w_j$ et $w_i \cdots w_n$ si $w = w_1 \cdots w_n$.
- Lorsqu'une boucle **for** est effectuée sur un certain élément i de $\{1, \ldots, n\}$, i prend **TOUTES** les valeurs entre 1 et n compris,
- L'élément i de la liste l sera noté $l[i]$. En particulier, une liste l est indicé de 1 à la longueur $|l|$ de la liste.
- Si l est une liste, alors l'élément $l[-i]$ désigne l'élément $l[|l| - i + 1]$.

5.1 Théorème de Chen-Fox-Lyndon

Dans le premier chapitre de ce mémoire, tout le travail accompli a permis de démontrer le fameux théorème de Chen-Fox-Lyndon. Pour ce faire, il a fallu comparer deux mots à l'aide d'un ordre $>_{\text{lex}}$ et regarder si, parmi toutes les factorisations d'un mot w, il en existait une qui respectait les conditions du théorème. Si telle est le cas, alors le mot w n'est pas un mot de Lyndon. Dans le cas contraire, w est bien un mot de Lyndon. Tout d'abord, il semble essentiel d'avoir un algorithme qui permet de comparer deux mots. L'algorithme 1 permet de vérifier si deux mots u et v sont décroissants.

Require: $u, v \in \mathcal{A}^*$
Ensure: Vrai si $u \geq_{\text{lex}} v$, Faux sinon
  if $|u| = |v|$ **then**
    for $i \in \{1, \ldots, |u|\}$ **do**
      if $u_i < v_i$ **then**
        return False
      end if
      if $u_i > v_i$ **then**
        return True
      end if
    end for
    return True
  else if $|u| < |v|$ **then**
    for $i \in \{1, \ldots, |u|\}$ **do**
      if $u_i > v_i$ **then**
        return True
      end if
      if $u_i < v_i$ **then**
        return False
      end if
    end for
    return False
  else
    for $j \in \{1, \ldots, |v|\}$ **do**
      if $u_j < v_j$ **then**
        return False
      end if
      if $u_j > v_j$ **then**
        return True
      end if
    end for
    return True
  end if

Algorithm 1: Lex_Order_Dr(u, v)

Cet algorithme est séparé en trois cas selon la longueur des deux arguments u et v. Cependant, dans chacun des trois cas, le pire scénario possible est de parcourir tout le mot dont la longueur est la plus petite. Dès lors, cet algorithme possède une complexité de l'ordre de $O(\min(|u|, |v|))$ dans le pire cas.

Après avoir implémenté cet algorithme, il est possible d'en créer un autre similaire. Le but de cet dernier est de déterminer si une liste de mot est décroissante ou non. Mais cette fois-ci, l'algorithme prendra en argument une liste. Dans ce qui va suivre, la taille de la liste sera noté $|l|$ et correspond à la somme des longueurs des éléments qui la compose. Pour chaque valeur de i dans $\{0, \ldots, |l| - 1\}$, la fonction Lex_Order_Dr est effectuée. Comme discuté ci-avant, la complexité de cet algorithme est $O(\min(|u|, |v|))$ où u, v sont ses deux arguments. Dans ce cas, la complexité de Lex_Order_Dr_l dans le pire cas est $O(\sum_{i=1}^{|l|-1} O(\min(|l[i]|, |l[i+1]|)))$. Si on pose $k = \max\{\min(|l[i]|, |l[i+1]|) : 1 \leq i \leq |l| - 1\}$, alors la complexité dans le pire cas devient $O(k \cdot (|l| - 1)) = O(k \cdot |l|)$.

Require: l une liste de mot de $\mathcal{A}^*$
Ensure: Vrai si l est une liste décroissante de mots, Faux sinon.
  for $i \in \{1, \ldots, |l| - 1\}$ **do**
    if not Lex_Order_Dr$(l[i], l[i+1])$ **then**
      return False
    end if
  end for
  return True

Algorithm 2: Lex_Order_Dr_l(l)

Enfin, avant de présenter un algorithme de test d'appartenance à la famille des mots de Lyndon basé sur le théorème de Chen-Fox-Lyndon, il faudrait disposer d'un autre algorithme permettant d'avoir toutes les factorisations possibles d'un mot w de $\mathcal{A}^*$.

Require: w un mot de $\mathcal{A}^*$.
Ensure: La liste l de toutes les factorisations de w dans $\mathcal{A}^*$.
  $l = [\,]$
  if $|w| = 1$ **then**
    $l \leftarrow [w]$
    return l
  else
    $l \leftarrow l + [[w]]$
    for $i \in \{1, \ldots, |w| - 1\}$ **do**
      $a \leftarrow$ Factorisation$(w[i+1:])$
      FirstPart $\leftarrow w[:i]$
      for $j \in \{1, \ldots, |a|\}$ **do**
        temp $\leftarrow$ FirstPart $+a[j]$
        $l \leftarrow l+$ temp
      end for
    end for
    return l
  end if

Algorithm 3: Factorisation(w)

Si w est une lettre, alors Factorisation(w) renvoit la liste $l = [w]$. Dans le cas où w n'est pas une lettre, alors la liste des factorisations possibles de w contient évidemment w lui-même. Pour construire le reste de la liste l, on considère un préfixe p de w d'une certaine longueur et on ajoute toutes les factorisations du suffixe s tel que $ps = w$. Maintenant, une question toute naturelle est de savoir la complexité de cet algorithme 3 dans le pire cas. Tout d'abord, le pire cas arrive dès que la longueur du mot donné en argument est supérieure à 2. Supposons maintenant que la longueur de l'argument w vaut $n \geq 2$. L'appel récursif à la fonction *Factorisation* prendra plus de temps pour renvoyer un résultat dans le cas où l'argument est de taille $n - 1$. De plus, si a est effectivement la liste des factorisations du suffixe de longueur $n-1$ de w, alors la longueur de a est 2^{n-2}. Procédons par récurrence pour montrer cette affirmation. Le cas où $n = 1$ est trival car il n'existe qu'une seule manière de factoriser une lettre. Supposons maintenant que le nombre de factoriser un mot de longueur $m \leq n$ est 2^{m-1}. Si w est un mot de $\mathcal{A}^{n+1}$, alors le nombre de manières de factoriser w est $1 + \sum_{i=1}^{n} 2^{i-1}$. En effet, w est une factorisation de

lui-même et, en considérant un préfixe de longueur $1 \leq i \leq n$, il y a 2^{i-1} de factoriser le suffixe de longueur $n+1-i$. Au final, on a

$$1 + \sum_{i=1}^{n} 2^{i-1} = 1 + \sum_{j=0}^{n-1} 2^j = 1 + \frac{1-2^n}{1-2} = 1 + 2^n - 1 = 2^n$$

D'où le fait qu'il y a 2^{n-1} façons de factoriser un mot de longueur n. Pour en revenir à l'algorithme 3, la complexité dans le pire peut être donné par la formule récursive suivante où $T(n)$ représente la complexité de l'algorithme 3 avec un argument de taille n.

$$T(n) = (n-1) \cdot T(n-1) + (n-1) \cdot 2^{n-2} \tag{I}$$

Une technique pour passer de cette formule récursive à une formule close est la technique dite du "plug-and-chug" ou méthode du télescopage. Cette méthode consiste à utiliser la récurrence afin de supprimer les termes $T(n-i)$ du membre de droite. Dans le cas de la formule I, en utilisant une première fois la formule de récurrence, on obtient

$$T(n) = (n-1) \cdot (n-2) \cdot T(n-2) + (n-1) \cdot (n-2)2^{n-3} + (n-1) \cdot 2^{n-2}$$

En itérant le télescopage, on obtient l'égalité suivante

$$T(n) = (n-1)! + \sum_{i=2}^{n-2} \frac{(n-1)!}{(n-i)!} \cdot 2^{n-i} + C$$

où C correspond à une constante dans le cas où l'on se ramène à la factorisation d'une lettre. Comme $(n-1)! \geq 2^{n-2}$ pour tout $n \geq 2$, la complexité $T(n)$ de la fonction *Factorisation(w)* est de l'ordre de $O((n-1)!)$.

Maintenant que ces algorithmes ont été décrits, il est possible de créer un algorithme Lyndon(w) qui va déterminer si oui ou non un mot w est un mot de Lyndon.

Grâce au théorème de Chen-Fox-Lyndon, il est possible de déterminer si w est un mot de Lyndon. En effet, si w admet une factorisation décroissante en mots de Lyndon plus courts, alors w n'est pas un mot de Lyndon. De ce fait, si $\mathcal{A}$ est l'alphabet sur lequel on travaille, tout élément de $\mathcal{A}$ est un mot de Lyndon mais le mot vide ϵ n'est pas, quant à lui, un mot de Lyndon. Si le mot w donné en argument est de longueur supérieure ou égale à 2, alors l'algorithme procède en deux temps. Tout d'abord, parmi toutes les factorisations possibles de w, Lyndon(w) va stocker dans une liste, nommée *temp*, toutes les factorisations décroissantes en mots de $\mathcal{A}^*$. Ensuite, pour chacune de ces factorisations, l'algorithme va vérifier que tous les mots de la factorisation sont des mots de Lyndon. Si tel est le cas, alors il existe une factorisation décroissante en mots de Lyndon et l'algorithme va renvoyer la valeur booléenne *False*. Mais, si dans toute factorisation décroissante, chacun des mots apparaissant est un mot de Lyndon, alors w n'est pas un mot de Lyndon. Si un mot u est un mot apparaissant dans une des factorisations décroissantes et n'est pas un mot de Lyndon, alors la factorisation considérée n'est pas conforme à celle décrite dans le théorème de Chen-Fox-Lyndon. La variable *Lyn* permet de détecter cette incompatibilité. Passons maintenant à la complexité de cet algorithme. Le pire cas est évidemment rencontré quand la longueur du mot donné en argument est

Require: w un mot de $\mathcal{A}^*$
Ensure: Vrai si w est un mot de Lyndon et Faux sinon.
  if $|w| = 0$ **then**
    return False
  else if $|w| = 1$ **then**
    return True
  else
    liste $\leftarrow$ Factorisation(w)[1 :]
    temp $\leftarrow$ [ ]
    for $i \in \{1, |\text{liste}|\}$ **do**
      if Lex_Order_Dr_l(liste[i]) **then**
        temp $\leftarrow$ temp + liste[i]
      end if
    end for
    if $|\text{temp}| \neq 0$ **then**
      for $j \in \{1, \ldots, |\text{temp}|\}$ **do**
        Lyn $\leftarrow$ True
        for word $\in$ temp[j] **do**
          if not Lyndon(word) **then**
            Lyn $\leftarrow$ False
          end if
        end for
        if Lyn **then**
          return False
        end if
      end for
      return True
    else
      return True
    end if
  end if

Algorithm 4: Lyndon(w)

supérieure ou égale à 2. Ensuite, il faut créer la liste des factorisations ce qui est possible en $O((n-1)!)$ avec n la longueur du mot w passé en argument. Ensuite, il faut identifier les factorisations décroissantes parmi toutes les factorisations non triviales (différentes du mot w lui-même). Donc, il faut tester $2^{n-1} - 1$ factorisations à l'aide de la fonction *Lex_ Order_ Dr_ l* dont la complexité est de l'ordre du $O(k \cdot |l|)$ avec $k = \max\{\min(|l[i]|, |l[i+1]|) : 1 \leq i \leq |l| - 1\}$. Dans ce cas de figure, si *liste* contient toutes les factorisations possibles et k est défini comme ci-avant, alors posons $M = \max\{k \cdot |l| : l \in \textit{liste}\}$. Dès lors, l'étape d'identification des factorisations décroissantes s'effectue en $O(2^{n-1} \cdot M)$. Ainsi, déterminer les factorisations décroissantes de w est fait en $O((n-1)! + 2^{n-1} \cdot M)$. Une fois les étapes de sélection des factorisations décroissantes et leurs insertions dans une liste nommée *temp* effectuées , il faut encore vérifier que chaque mot apparaissant dans ces factorisations sont des mots de Lyndon. En posant $N = \max\{|w_i| : \exists j \in \{1, \ldots, |\textit{temp}|\}$ tel que $w_i \in \textit{temp}[j]\}$ et $L = \max\{|l| : l \in \textit{temp}\}$, l'appartenance à $\mathcal{L}$ des mots présents dans les factorisations décroissantes sont de l'ordre de $O(|\textit{temp}| \cdot N \cdot L + C)$ avec C une

constante qui correspond au temps qu'il faut pour vérifier que la variable *Lyn* soit vrai ou non. Au final, la complexité dans le pire cas de $Lyndon(w)$ est de l'ordre de $O((n-1)! + M \cdot 2^{n-1} + |temp| \cdot N \cdot L + C)$ avec n la longueur de w.

Un autre algorithme pour trouver la factorisation décroissante en mots de Lyndon est un algorithme présenté par Duval dans son article [5]. L'algorithme est le suivant.

Require: $w = a_1 \cdots a_n$ un mot de $\mathcal{A}^*$
Ensure: La liste Fact contient les indices des mots de Lyndon de la factorisation décroissante de w.

```
Fact ← [ ]
k ← 0
n ← |w|
while k < n do
  i ← k + 1
  j ← k + 2
  while j ≠ n + 1 and a_i ≤ a_j do
    if a_i < a_j then
      i ← k + 1
      j ← j + 1
    else if a_i = a_j then
      i ← i + 1
      j ← j + 1
    end if
  end while
  while k < i do
    k ← k + (j − i)
    Fact ← Fact +[k]
  end while
end while
return Fact
```

Algorithm 5: Duval(w)

Le but de cet algorithme est d'avoir les indices des sous-mots de w apparaissant dans la factorisation de w respectant le théorème de Chen-Fox-Lyndon. Par exemple, si le mot 0100111 est donné en argument, alors l'algorithme 5 renvoit la liste *Fac*$= [1, 6]$ ce qui implique que la factorisation respectant le théorème de Chen-Fox-Lyndon est $w = (01, 00111)$. De manière générale, si la fonction *Duval* renvoit la liste $(k_1, \ldots, k_m)$, alors la factorisation respectant le théorème de Chen-Fox-Lyndon est $(w_1 \cdots w_{k_1}, \ldots, w_{k_{m-1}} \cdots w_{k_m})$. Pour trouver cette liste d'indices, l'algorithme se base sur le raisonnement. Tout d'abord, le mot w donné en argument sera lu de gauche à droite. Ensuite, on introduit deux variables i et j telles que j soit l'indice de la lettre qui est en train d'être lu et i est tel que

$$a_1 \cdots a_{i-1} = a_{j-i+1} \cdots a_{j-1} \text{ et } a_1 \cdots a_{j-i} \in \mathcal{L} \qquad (*)$$

Ensuite, lors de la comparaison de a_i avec a_j, si $a_i = a_j$ ou $a_i < a_j$, alors j augmente de 1 i est modifié de telle sorte que l'égalité (*) soit respectée. Dans le cas où $a_i > a_j$ ou $j = n+1$, considérons que $i = i'$ et $j = j'$. Alors la variable $k = j' - i'$ est introduite afin que le suffixe restant soit $a_{k+1} \cdots a_n$ et le premier facteur de la factorisation du théorème de Chen-Fox-Lyndon est $a_1 \cdots a_{j'-i'}$. Dès lors, on met k dans la liste *Fact* et le préfixe $a_1 \cdots a_k$ n'est plus considéré. Après cette opération, on recommence le raisonnement avec $i = k + 1$ et $j = k + 2$. Si $k \geq n$, alors on est à la fin du mot w et

on renvoit la liste *Fact*. En ce qui concerne la complexité de cet algorithme 5 dans le pire cas, elle est de l'ordre de $O(n)$. En effet, le nombre maximal de comparaison à effectuer dans cet algorithme arrive quand le mot w donné en argument est de la forme $w = a^*$ avec $a \in \mathcal{A}$. En effet, dans la deuxième boucle *while*, il est possible d'y rentrer n fois en ayant, à chaque fois, $a_i \leq a_j$. De plus, dans la troisième boucle *while*, le nombre maximal d'entrée est de n. Cela correspond au cas où $j = n+1$ et $i = n$. Ce cas n'est possible que si i a été augmenté de 1 à chaque fois. Dès lors, il faut que $a_i = a_j$. Au final, il faut bien que $w \in a^*$ avec $a \in \mathcal{A}$ et la complexité dans la pire cas de l'algorithme 5 est bien $O(n)$.

5.2 Algorithmes concernant les mots de Nyldon

Dans le deuxième chapitre de ce mémoire, plusieurs résultats permettent de déduire des algorithmes ainsi que des optimisations de ces derniers.

5.2.1 Définition récursive

Suite au théorème de Chen-Fox-Lyndon, comme expliqué au début du deuxième chapitre, Darij Grinberg posta un article (cf. [7]) posant une définition récursive des mots de Nyldon, un équivalent du théorème de Chen-Fox-Lyndon. Cette définition permet de construire l'algorithme 6.

Dans cet algorithme 6, la seule différence qui peut être faite avec l'algorithme 4 est le fait que les factorisations ne sont plus décroissantes mais croissantes. Sinon, le raisonnement reste identique à celui de l'algorithme 4. De même, la complexité de Nyldon(w) est identique à celle de Lyndon(w). En effet, la fonction *Lex_ Order_ Cr_ l* est définie à partir de la fonction *Lex_ Order_ Dr_ l*. De plus, la fonction *Lex_ Order_ Cr* utilisée dans la définition de *Lex_ Order_ Cr_ l* utilise la fonction *Lex_ Order_ Dr* dans sa définition.

Require: $u, v \in \mathcal{A}^*$
Ensure: Vrai si $u \leq_{\text{lex}} v$ et Faux sinon
  return Lex_Order_Dr(v, u)

Algorithm 7: Lex_Order_Cr(u, v)

Require: l une liste de mots de $\mathcal{A}^*$
Ensure: Vrai si la liste est croissante de mots de $\mathcal{A}^*$ et Faux sinon
  for $i \in \{1, \ldots, |l| - 1\}$ **do**
    if not Lex_Order_Cr(l[i], l[$i+1$]) **then**
      return False
    end if
  end for
  return True

Algorithm 8: Lex_Order_Cr_l(l)

Donc, la complexité de *Lex_ Order_ Cr_ l* dans le pire cas est de l'ordre du $O(k \cdot |l|)$ avec $k = \max\{\min(l[i], l[i+1]) : 1 \leq i \leq |l| - 1\}$ et $|l|$ la longueur de la liste donnée en argument. Au final, comme toutes les autres opérations sont identiques à celles de *Lyndon(w)*, la complexité de *Nyldon(w)* est, comme *Lyndon(w)*, de l'ordre de $O((n-1)! + M \cdot 2^{n-1} + |temp| \cdot N \cdot L + C)$ en utilisant les mêmes notations que le cas de l'analyse de la complexité de l'algorithme 4.

```
Require: w ∈ A*
Ensure: True si w est un mot de Nyldon et False sinon.
  if |w| = 0 then
    return False
  else if |w| = 1 then
    return True
  else
    liste ← Factorisation[1 : ]
    temp ← [ ]
    for i ∈ {1, ..., |liste|} do
      if Lex_Order_Cr_l(liste[i]) then
        temp.append(liste[i])
      end if
    end for
    if |temp| ≠ 0 then
      for j ∈ {1, ..., |temp|} do
        Nyl ← True
        for word ∈ temp[j] do
          if not Nyldon(word) then
            Nyl ← False
          end if
        end for
        if Nyl then
          return False
        end if
      end for
      return True
    else
      return True
    end if
  end if
```

Algorithm 6: Nyldon(w)

Imaginons maintenant que le type d'argument change. Par exemple, si une liste l est passée en argument, il faudrait une fonction permettant d'afficher les mots de Nyldon appartenant à l. L'algorithme suivant permet d'appliquer ce raisonnement.

La complexité de *Nyldon_ l* dans le pire cas est de l'ordre de $O(|l| \cdot ((n-1)! + M \cdot 2^{n-1} + |temp| \cdot N \cdot L + C))$. En particulier, si $\mathcal{A}$ est un alphabet de taille k, alors la complexité de *Nyldon_ l* devient $O(k^n \cdot ((n-1)! + M \cdot 2^{n-1} + |temp| \cdot N \cdot L + C))$. Cependant, il est possible de réduire cette complexité grâce à un changement dans la fonction *Nyldon* de l'algorithme 6. En effet, dans le deuxième chapitre de ce mémoire, la proposition 3.4 permet d'éliminer certains mots de longueur au moins 2 dont le préfixe de longueur 2 n'est pas adéquat. Dès lors, la fonction *Nyldon2* repris dans l'algorithme 10 prend en compte cette proposition 3.4.

Require: l la liste de mots de longueur n de $\mathcal{A}^*$
Ensure: Les mots w de l qui sont des mots de Nyldon
  for $i \in \{1, \ldots, |l|\}$ **do**
    if Nyldon(l[i]) **then**
      print l[i]
    end if
  end for

Algorithm 9: Nyldon_l(l)

Require: w un mot de $\mathcal{A}^*$
Ensure: Vrai si w est un mot de Nyldon et Faux sinon
  if $|w| = 0$ **then**
    return False
  else if $|w| = 1$ **then**
    return True
  else if $w[1] \leq w[2]$ **then**
    return False
  else
    liste $\leftarrow$ Factorisation[1 :]
    temp $\leftarrow$ [ ]
    for $i \in \{1, \ldots, |\text{liste}|\}$ **do**
      if Lex_Order_Cr_l(liste[i]) **then**
        temp.append(liste[i])
      end if
    end for
    if $|\text{temp}| \neq 0$ **then**
      for $j \in \{1, \ldots, |\text{temp}|\}$ **do**
        Nyl $\leftarrow$ True
        for word $\in$ temp[j] **do**
          if not Nyldon(word) **then**
            Nyl $\leftarrow$ False
          end if
        end for
        if Nyl **then**
          return False
        end if
      end for
      return True
    else
      return True
    end if
  end if

Algorithm 10: Nyldon2(w)

À première vue, cette modification ne semble pas influencer la complexité dans le pire cas de *Nyldon2*. En effet, cette dernière reste de l'ordre de $O((n-1)! + M \cdot 2^{n-1} + |temp| \cdot N \cdot L + C)$. Cependant, en utilisant *Nyldon2* à la place de *Nyldon* dans la fonction *Nyldon_ l*, le changement sur la complexité est visible. Regardons maintenant le nombre de mots de longueur $n \geq 2$ sur un alphabet de taille k qui serait exclut de l'ensemble $\mathcal{N}$ par la proposition 3.4. Par ce résultat, un mot w n'appartient pas à $\mathcal{N}$ s'il commence par ij avec $0 \leq i \leq j \leq k$. Donc, on a, avec un alphabet $\mathcal{A} = \{0, \ldots, k-1\}$,

- si $i = 0$, alors $j \in \mathcal{A}$. Il y a k possibilité(s) pour choisir une valeur de j.
- si $i = 1$, alors $j \in \{1, \ldots, k-1\}$. Il y a $k-1$ possibilité(s) pour choisir une valeur de j.
- si $i = 2$, alors $j \in \{2, \ldots, k-1\}$. Il y a $k-2$ possibilité(s) pour choisir une valeur de j.
  $\vdots$
- si $i = k-1$, alors $j = k-1$. Il y a $k-(k-1) = 1$ possibilité pour choisir une valeur de j.

Dès lors, en partant de ce constat, on peut obtenir le nombre de mots de longueur $n \geq 2$ qui sont éliminés par la proposition 3.4. Dans le tableau ci-dessous, on suppose que w peut s'écrire $ijw_2 \cdots w_{n-1}$ avec $i, j \in \mathcal{A}$ un alphabet de taille k.

Valeur de i	Nombre de mots de longueur n éliminés
0	$k \cdot k^{n-2} = k^{n-1}$
1	$(k-1) \cdot k^{n-2} = k^{n-1} - k^{n-2}$
2	$(k-2) \cdot k^{n-2} = k^{n-1} - 2k^{n-2}$
3	$(k-3) \cdot k^{n-2} = k^{n-1} - 3k^{n-2}$
$\vdots$	$\vdots$
$k-1$	$(k-(k-1)) \cdot k^{n-2} = k^{n-2}$
Total	$\sum_{i=0}^{k-1}[(k-i) \cdot k^{n-2}] = \dfrac{k^n + k^{n-1}}{2}$

Au total, l'algorithme détermine l'exclusion de $\mathcal{N}$ de $\frac{k^n+k^{n-1}}{2}$ mots de longueur n en temps constant ce qui représente plus de la moitié des mots de longueur n sur un alphabet $\mathcal{A}$ de taille k. Au final, l'algorithme *Nyldon_ l* qui utilise *Nyldon2* a une complexité dans le pire cas qui est de l'ordre de $O(\frac{k^n-k^{n-1}}{2} \cdot ((n-1)! + M \cdot 2^{n-1} + |temp| \cdot N \cdot L + C))$.

5.2.2 Optimisation

Les différents algorithmes présentés ci-dessus concernant les mots de Nyldon utilisent la définition récursive et, comme présenté ci-dessus, cela augmente la complexité des différents algorithmes. Une autre idée permettant de déterminer si un mot w est un mot de Nyldon est de regarder sa factorisation croissante en mots de Nyldon. Si cette dernière est de longueur au moins 2, alors w n'est pas un mot de Nyldon. Pour construire une telle factorisation, l'article [2] présente l'algorithme suivant.

Require: $w \in \mathcal{A}^+$
Ensure: NylF est la factorisation croissante en mots de Nyldon de w
  $n \leftarrow |w|$
  NylF $\leftarrow [w[n]]$
  for $i \in \{1, \ldots, n-1\}$ **do**
    NylF $\leftarrow (w[n-i]$, NylF)
    while $|\text{NylF}| \geq 2$ **and** NylF$[1] >_{\text{lex}}$ NylF$[2]$ **do**
      NylF $\leftarrow$ (NylF$[1]\cdot$ NylF$[2]$, NylF$[3], \ldots,$ NylF$[-1])$
    end while
  end for
  return NylF

Algorithm 11: Factorisation croissante en mots de Nyldon

Maintenant, l'objectif est de savoir si cet algorithme est correct, c'est-à-dire, si la sortie NylF contient effectivement la factorisation croissante en mots de Nyldon du mot w donné en argument.

Proposition 5.1. *L'algorithme 11 s'arrête pour tout $w \in \mathcal{A}^+$ et NylF contient bien la factorisation croissante en mots de Nyldon.*

Démonstration. En effet, l'algorithme 11 s'arrête pour tout mot $w \in \mathcal{A}^+$ donné en argument car la bouche *for* est itérée $n-1$ fois et, pour chaque valeur de cette boucle *for*, la boucle *while* est itérée i fois dans le pire des cas. Prouvons maintenant que l'algorithme 11 est correct. Le but est de montrer que, pour tout étape i de la boucle *for*, si NylF contient la factorisation croissante en mots de Nyldon du suffixe $w[n-i+1:n]$ de longueur i de w avant de rentrer une nouvelle fois dans la boucle *for*, alors, après une nouvelle sortie de la boucle *for*, NylF contient la factorisation croissante en mots de Nyldon du suffixe $w[n-i:n]$ de longueur $i+1$ de w. Comme NylF est initialement égal à $[w[n]]$, qui est la factorisation croissante en mots de Nyldon du suffixe de longueur 1 de w, le but énoncé ci-avant implique que, après la dernière itération de la boucle *for*, qui correspond à $i = n-1$, NylF contient la factorisation croissante de w en mots de Nyldon. Pour prouver l'objectif ci-dessus, supposons que $1 \leq i < n$ et que, avant la $i^{\text{ième}}$ itération de la boucle *for*, NylF contient la factorisation croissante en mots de Nyldon du suffixe $w[n-i+1:n]$ de longueur i de w que l'on notera $(u_1, \ldots, u_k)$. Tout d'abord, NylF est modifié en $(w[n-i], u_1, \ldots, u_k)$ avant de rentrer dans la boucle *while*. Par la proposition 3.9, pour tout $j \in \{1, \ldots, k\}$, u_j est le plus long suffixe qui est un mot de Nyldon de $u_1 \cdots u_j$. De plus, comme $w[-i]$ est une lettre, u_j est aussi le plus suffixe propre de Nyldon de $w[n-i]u_1 \cdots u_j$. Par la proposition 3.12, on obtient récursivement que, pour tout $j \in \{1, \ldots, k\}$, le mot $w[n-i]u_1 \cdots u_j$ est un mot de Nyldon si et seulement si $w[n-i]u_1 \cdots u_{j-1} >_{\text{lex}} u_j$. Dès lors, deux cas sont possibles. D'une part, il existe un $j \in \{1, \ldots, k-1\}$ tel que

$$\begin{aligned} w[n-i] &>_{\text{lex}} u_1 \\ w[n-i]u_1 &>_{\text{lex}} u_2 \\ &\vdots \\ w[n-i]u_1u_2\cdots u_{j-1} &>_{\text{lex}} u_j \\ w[n-i]u_1\cdots u_j &\leq_{\text{lex}} u_{j+1}, \end{aligned}$$

auquel cas les mots $w[n-i]u_1 \cdots u_{j'}$ sont des mots de Nyldon pour tout $j' \in \{1, \ldots, j\}$ par la proposition 3.12. Dans ce cas, en itérant la boucle *while*, l'algorithme 11 va modifier NylF de $(w[n-i], u_1, \ldots, u_k)$ en $(w[n-i]u_1, u_2, \ldots, u_k)$, $\ldots$, $(w[n-i]u_1 \cdots u_j, u_{j+1}, \ldots, u_k)$. La dernière valeur de NylF, qui est $(w[n-i]u_1 \cdots u_j, u_{j+1}, \ldots, u_k)$, est donc la factorisation croissante en mots de Nyldon du suffixe $w[n-i:n]$ de longueur $i+1$ de w. D'autre part, on pourrait avoir

$$\begin{aligned} w[n-i] &>_{\text{lex}} u_1 \\ w[n-i]u_1 &>_{\text{lex}} u_2 \\ &\vdots \\ w[n-i]u_1u_2 \cdots u_{k-1} &>_{\text{lex}} u_k \end{aligned}$$

auquel l'algorithme 11 modifiera successivement NylF de $(w[n-i], u_1, \ldots, u_k)$ en $(w[n-i]u_1, u_2, \ldots, u_k)$, $(w[n-i]u_1u_2, u_3, \ldots, u_k), \ldots, (w[n-i]u_1 \cdots u_k)$. De la même que dans l'autre cas de figure, par la proposition 3.12, on obtient $w[n-i:n] = w[n-i]u_1 \cdots u_k$ est un mot de Nyldon. Dès lors, la dernière valeur de NylF correspond à la factorisation croissante en mots de Nyldon du suffixe de longueur $i+1$ de w. □

En ce qui concerne la complexité de cet algorithme 11, elle est de l'ordre de $O(n^2)$. En effet, comme énoncé au début de la démonstration de la proposition 5.1, pour chaque i de la boucle *for*, la boucle *while* est effectué au plus i fois. Dès lors, comme les manipulations sur les listes et la comparaison entre deux mots peuvent être effectuées en temps constant, la complexité dans le pire cas est de l'ordre de $O(1+\ldots+n-1) = O(\frac{n\cdot(n-1)}{2}) = O(n^2)$. Cependant, une question se pose quant à la complexité constante de la comparaison de deux mots. En effet, si deux lettres sont comparées, alors il est trivial de montrer que la complexité de cette comparaison est de l'ordre de $O(1)$. Par contre, si deux mots de longueur supérieure ou égale à 2 sont comparés, alors cela devient moins évident. L'algorithme 1 permet de déterminer si le mot u et supérieur ou égal au mot v suivant l'ordre lexicographique $\leq_{\text{lex}}$. De plus, cet algorithme a une complexité dans le pire cas de l'ordre du $O(\min(|u|, |v|))$ qui n'est pas constant. Pour pouvoir comparer deux facteurs de $w \in \mathcal{A}^{\geq 1}$ en temps constant, une idée sera de disposer d'un tableau reprenant les plus long préfixes de toutes les paires de suffixes. De ce fait, en prenant deux facteurs du mot w, il suffirait de regarder dans ce tableau pour avoir la longueur du préfixe commun aux deux facteurs et de comparer, si possible, la lettre de la position suivante dans les deux facteurs. Le cas où c'est impossible reviendrait à dire qu'un facteur est préfixe de l'autre. Pour ce faire, introduisons les trois notions de "suffix array", de "LCP array" et de "range minimum query" (RMQ).

Suffix array ou tableau des suffixes

Le tableau des suffixes (ou suffix array) est un tableau reprenant tous les suffixes d'un mot w donné au départ. Cette structure de données a été introduit en 1990, lors d'une conférence à San Francisco, par Udi Manber et Gene Myers (cf. [10]) et avait, entre autre, comme objectif de pouvoir localiser un facteur f de w et même de pouvoir compter le nombre de fois où ce facteur f apparaissait. Les tableaux de suffixes ont comme avantage d'être moins encombrant que les arbres des suffixes (suffix tree) qui étaient utilisés, entre autre, pour chercher une chaine de caractères précise S dans un mot w ou pour avoir le plus grand palindrome apparaissant dans un mot w donné. D'après l'article [10], les tableaux de suffixes utilisent cinq fois

moins d'espaces que les arbres des suffixes ce qui est non négligeables. De plus, il est possible de construire les tableaux de suffixes en $O(n)$ et ils possèdent une complexité spatiale de l'ordre de $O(n)$. En effet, si un mot w est de longueur n, alors le tableau des suffixes est de longueur n. Maintenant, définissons rigoureusement le tableau des suffixes.

Définition 5.2. (Tableau des suffixes ou suffix array) Soient $\mathcal{A}$ un alphabet totalement ordonné, $\leq_{\text{lex}}$ l'ordre lexicographique sur $\mathcal{A}$ et w un mot de $\mathcal{A}^+$ de longueur n. Le tableau des suffixes T de w est construit tel que, pour tout $1 \leq i \leq n$, on a $T[i]$ est l'indice de début du $i^{\text{ème}}$ suffixe le plus petit pour l'ordre $\leq_{\text{lex}}$.

Exemple 5.3. Soient $\mathcal{A} = \{a, b\}$ avec $a < b$ et $w = aabbabba$. Le tableau de la figure 5.1 est construit de la manière suivante : la première colonne est l'indice de début du suffixe, la deuxième colonne est le suffixe en question et la troisième colonne contient le rang (ou la place) du suffixe selon l'ordre lexicographique.

Indice du suffixe	Suffixe	Rang
1	*aabbabba*	2
2	*abbabba*	4
3	*bbabba*	8
4	*babba*	6
5	*abba*	3
6	*bba*	7
7	*ba*	5
8	*a*	1

FIGURE 5.1 – Ordre des suffixes de $w = aabbabba$

Grâce à cette figure 5.1, le tableau des suffixes T est le suivant : $T = [8, 1, 5, 2, 7, 4, 6, 3]$.

LCP array ou tableau LCP

Le tableau LCP est une structure annexe du tableau des suffixes. LCP veut dire Longest Common Prefix, plus long préfixe commun. Ce tableau va permettre de stocker les longueurs des plus longs préfixes de deux suffixes consécutifs. Il est défini de la manière suivante.

Définition 5.4. (Tableau LCP ou LCP array) Soient $\mathcal{A}$ un alphabet totalement ordonné, w un mot de $\mathcal{A}^*$ de longueur n et T le tableau des suffixes de w. Alors L est un tableau de longueur n tel que $L[1]$ n'est pas défini et, pour tout $2 \leq i \leq n$, $L[i] = lcp(w[T[i-1] :], w[T[i] :])$ où $lcp(u, v)$ est la longueur du plus long préfixe commun entre u et v.

Exemple 5.5. Reprenons le mot $w = aabbabba$ de l'exemple 5.3. Le tableau des suffixes T est $[8, 1, 5, 2, 7, 4, 6, 3]$. Construisons maintenant le tableau LCP L. Par définition, $L[1] = \nexists$. Ensuite, $L[2] = lcp(w[T[1] :], w[T[2] :]) = lcp(a, aabbabba) = 1$. En effectuant ce calcul pour chaque paire de suffixes consécutifs, on obtient le tableau LCP $L = [\nexists, 1, 1, 4, 0, 2, 1, 3]$.

Range minimum query (RMQ)

En informatique, il est souvent demandé de déterminer, dans un tableau, l'indice de l'élément minimal de ce tableau. Il existe un moyen d'étendre ce problème en cherchant l'élément minimal pour un tableau d'élément comparable comme des nombres entiers par exemple. Pour résoudre ce genre de problème, il existe, en informatique, le range minimum query (RMQ).

Définition 5.6. (Range minimum query) Soient L un tableau de taille n d'objetz comparablez à l'aide de l'ordre $\prec$ et deux nombres l, r tels que $1 \leq l \leq r \leq n$. Alors le range minimum query de T entre l et r, noté $RMQ_L(l, r)$, est l'indice dans L de l'élément minimal de $L[l : r]$

Exemple 5.7. Reprenons une nouvelle fois l'exemple 5.3 et le mot $w = aabbabba$. Si l'on considère le tableau LCP L et que l'on cherche $RMQ_L(2, 8)$, alors on obtient $RMQ_L(2, 8) = 5$ qui est l'indice correspondant à l'élément 0.

Comparaison des mots de $\mathcal{A}^*$

Avant d'exécuter la fonction de l'algorithme 11, pour que la comparaison se fasse en $O(1)$, il faut créer une structure qui permet de stocker les RMQ du tableau LCP L pour tous les indices $1 \leq l \leq r \leq n$ avec n la longueur du mot w donné en argument à l'algorithme 11. Cela peut se faire via une matrice $n \times n$ mais il existera des cases inutilisées ce qui n'est pas très optimal. En effet, la matrice sera une matrice diagonale supérieure vu la définition de RMQ. Par contre, en utilisant un tableau dont chaque élément sont des tableaux (tableau multidimensionnel), l'espace mémoire utilisé sera minimal. Notons la structure choisie R. Donc, avant de lancer l'algorithme 11, il faut calculer le tableau des suffixes T, l'inverse du tableau des suffixes T^{-1} ($T^{-1}[j] = i \Leftrightarrow T[i] = j$) et le tableau LCP L. Ensuite, on stocke les RMQ dans un tableau multidimensionnel par exemple. Dès que l'on a tous ces objets, on peut lancer l'algorithme 11. À la place de comparer les deux éléments de la liste $NylF$, on compare les suffixes démarrant aux indices correspondants. Par exemple, si on doit comparer $w_{n-3}w_{n-2}$ avec $w_{n-1}w_n$, on peut comparer les suffixes $w[n-3:]$ et $w[n-1:]$. Pour ce faire, il suffit d'aller dans R pour voir la valeur de $RMQ_H(T^{-1}[n-3]+1, T^{-1}[n-1])$ si $T^{-1}[n-3]+1 \leq T^{-1}[n-1]$ ou, dans le cas contraire, $RMQ_H(T^{-1}[n-1]+1, T^{-1}[n-3])$. Après avoir obtenu cette information notée r, en $O(1)$, il suffit de comparer r et $\min(|NylF(1)|, |NylF(2)|)$. Si $r \geq \min(|NylF(1)|, |NylF(2)|)$, alors $NylF(1) \leq_{\text{lex}} NylF(2)$ si $\min(|NylF(1)|, |NylF(2)|) = |NylF(1)|$ et $NylF(2) \leq_{\text{lex}} NylF(1)$ sinon. Par contre, si $r < \min(|NylF(1)|, |NylF(2)|)$, alors il suffit de comparer $NylF(1)[r+1]$ et $NylF(2)[r+1]$. Toutes ces opérations s'effectuent en $O(1)$ d'où la complexité de l'algorithme 11.

Une étude un peu plus poussée de cet algorithme permet même de diminuer cette complexité dans le pire cas à $O(n)$ avec n la longueur du mot w donné en argument. En effet, en appliquant cet algorithme 11, on se rend compte que le nombre de comparaison avant concaténation est égal n. Cela correspond à la comparaison des lettres pour créer les facteurs de la factorisation. De plus, le nombre de comparaison supplémentaire permettant à partir de deux facteurs de la factorisation de créer un facteur plus grand est au plus $n-1$. Dès lors, l'algorithme 11 effectue au plus $2 \cdot n - 1$ comparaisons. Comme ces comparaisons sont effectuées en $O(1)$ à l'aide du tableau

des suffixes, du tableau LCP et du RMQ, la complexité dans le pire cas est de l'ordre du $O(n)$.

5.2.3 Algorithme de Mélançon

Un autre algorithme présenté dans ce mémoire est l'algorithme de Mélançon. Pour rappel, ce dernier prend en argument un mot primitif $w \in \mathcal{A}^*$ et, dans notre cas, renvoit le mot de Nyldon appartenant à la classe de conjugaison de w. Le raisonnement derrière cette procédure résulte de la démonstration du corollaire 4.35. L'algorithme 12 se présente de la manière suivante.

Require: $w \in \mathcal{A}^+$ primitif
Ensure: NylC est le mot de Nyldon conjugué de w
  NylC $\leftarrow$ liste des lettres de w
  $T \leftarrow$ liste des lettres de w
  while |NylC| $>$ **do**
    if $T[1] = \min_{<_{\text{lex}}}$ NylC **and** $T[1] <_{\text{lex}} T(-1)$ **then**
      $T \leftarrow (T[2], \ldots, T[-2], T[-1] \cdot T[1])$
    end if
    $i \leftarrow 2$, $j \leftarrow 2$
    while $j \leq |T|$ **do**
      while $i \leq |T|$ **and** $T[i] \neq \min_{<_{\text{lex}}}$ NylC **do**
        $i \leftarrow i + 1$
      end while
      if $i \leq |T|$ **and** $T[i] <_{\text{lex}} T[i-1]$ **then**
        $T \leftarrow (T[1], \ldots, T[i-1] \cdot T[i], \ldots, T[-1])$
      end if
      $j \leftarrow i + 1$, $i \leftarrow i + 1$
    end while
    NylC $\leftarrow T$
  end while
  return NylC

Algorithm 12: Mélançon(w)

En ce qui concerne la complexité dans le pire cas de cette fonction *Mélançon*, elle est de l'ordre de $O(n^2)$ avec n la longueur du mot w. En effet, le pire cas correspond à une concaténation par itération de la boucle *while* sur la longueur de NylC. En d'autres termes, la taille de la liste NylC diminue d'un cran à chaque tour de boucle. De plus, pour chaque valeur de |NylC|, les boucles *while* sur i et j sont effectuées au plus $|\text{NylC}| - 1$ fois. De plus, une fois que la boucle *while* sur i effectue |NylC| itération, elle n'est plus jamais effectuée car la condition d'entrée n'est plus respectée. Au final, si $n = |\text{NylC}|$, alors le nombre de fois que les boucles *while* sur i et j sont effectuées est $(n-1) + \ldots + 1 = \frac{n \cdot (n-1)}{2}$ d'où le fait que la complexité dans le pire cas de l'algorithme 12 est de l'ordre de $O(n^2)$.

Il existe une variante de l'algorithme 12 présenté dans l'article [17]. Cette variante a pour but de donner la factorisation croissante en mots de Nyldon.

Require: $w \in \mathcal{A}^*$ primitif
Ensure: Fact contient la factorisation croissante en mots de Nyldon de w.
NylC $\leftarrow$ liste des lettres de w
$T \leftarrow$ liste des lettres de w
while |NylC| $>$ **do**
if $T[1] = \min_{<_{\text{lex}}}$ NylC **and** $T[1] <_{\text{lex}} T(-1)$ **then**
Fact $\leftarrow$ Fact + $T[1]$
$T \leftarrow T[2:]$
continue
end if
$i \leftarrow 2$, $j \leftarrow 2$
while $j \leq |T|$ **do**
while $i \leq |T|$ **and** $T[i] \neq \min_{<_{\text{lex}}}$ NylC **do**
$i \leftarrow i + 1$
end while
if $i \leq |T|$ **and** $T[i] <_{\text{lex}} T[i-1]$ **then**
$T \leftarrow (T[1], \ldots, T[i-1] \cdot T[i], \ldots, T[-1])$
end if
$j \leftarrow i + 1$, $i \leftarrow i + 1$
end while
NylC $\leftarrow T$
end while
Fact $\leftarrow$ Fact + NylC
return Fact

Algorithm 13: Factorisation avec Mélançon

Le raisonnement de cet algorithme 13 est similaire à celui de l'algorithme 12. La différence intervient avant de concaténer deux éléments de la liste T. Si le premier élément de T est l'élément minimal de la liste, alors, au lieu de le concaténer avec le dernier élément, il est placé dans une liste *Fac* qui va contenir la factorisation croissante en mots de Nyldon. Après ce placement dans Fac, le premier élément n'est plus pris en compte et on applique l'algorithme de Mélançon au reste de la liste T. Pour ce qui est de la complexité dans le pire cas, elle est identique à celle de l'algorithme 12 et est donc de l'ordre de $O(n^2)$.

Bibliographie

[1] Jean Berstel et Dominique Perrin, *The origins of combinatorics on words*, European Journal of Combinatorics **28** (2007), p. 996-1022, disponible via l'URL <http://www-igm.univ-mlv.fr/~berstel/Articles/2007Origins.pdf>, consulté le 12 octobre 2023.

[2] Émilie Charlier, Manon Philibert et Manon Stipulanti, *Nyldon words*, Journal of Combinatorial Theory. Series A **167** (2019), p. 60-90, disponible via l'URL <https://orbi.uliege.be/bitstream/2268/232399/1/Nyldon-final.pdf>, consulté le 26 juin 2023.

[3] Kuo-Tsai Chen, Ralph Fox et Roger Lyndon, *Free differential calculus iv. The quotient groups of the lower central series*, The Annals of Mathematics **68** (1958), n° 1, p. 81-95.

[4] Joseph Dallemagne, *Histoire des mathématiques : Théorème de Chen-Fox-Lyndon*, 2023.

[5] Jean-Pierre Duval, *Mots de lyndon et périodicité*, Informatique théorique et Applications **14** (1980).

[6] Robert Ferréol, *L'encyclopédie en ligne des suites d'entiers*, disponible via l'URL <https://tangente-mag.com/article.php?id=6019>, consulté le 29 septembre 2023.

[7] Darij Grinberg, *"Nyldon words" : understanding a class of words factorizing the free monoid increasingly*, 18 novembre 2014, disponible via l'URL <https://mathoverflow.net/questions/187451/nyldon-words-understanding-a-class-of-words-factorizing-the-free-monoid-incre>, consulté le 24 octobre 2023.

[8] M Lothaire, *Combinatorics on words : Chapiter 1 : words*, Cambridge University Press (2009), n° 2, p. 1-17, disponible via l'URL <https://www.cambridge.org/core/books/combinatorics-on-words/words/AA7AA0D89303EB7CDFCD6BC5 consulté le 2 octobre 2023.

[9] M. Lothaire, *Combinatorics on words : Chapter 5 : factorizations of free monoids*, Cambridge University Press (2009), n° 2, p. 63-104, disponible via l'URL <https://www.cambridge.org/core/services/aop-cambridge-core/content/view/2533601DFCB010FD0DB0A27C85FED8B9/9780511566097c5_p63-104_CBO.pdf/factorizations_of_free_monoids.pdf>, consulté le 26 juin 2023.

[10] Udi Manber et Gene Myers, *Suffix arrays : a new method for on-line string searches*, First annual acm-siam symposium on discrete algorithms, 1990, p. 319-327.

[11] Udi Manber et Gene Myers, *Suffix arrays : a new method for on-line string searches*, SIAM Journal on Computing **22** (1993).

[12] Encyclopedia of Mathematics, *Hall set*, anglais, 4 décembre 2014, disponible via l'URL <http://encyclopediaofmath.org/index.php?title=Hall_set&oldid=35333>, consulté le 27 juin 2023.

[13] Guy Melançon, *Combinatorics of hall trees and hall words*, Journal of Combinatorial Theory. Series A **59** (1992), p. 285-308.

[14] Dominique Perrin et Christophe Reutenauer, *Hall sets, lazard sets and comma-free codes*, Discrete Mathematics **341** (2018), p. 232-243.

[15] Dominique Perrin, Christophe Reutenauer et Jean Berstel, *Codes and automata : Factorizations of free monoids*, Cambridge University Press, 2009.

[16] Neil Sloane, *The On-Line Encyclopedia of Integer Sequences*, disponible via l'URL <https://oeis.org/?language=french>, consulté le 29 septembre 2023.

[17] Garg Swapnil, *New results on nyldon words and nyldon-like sets*, Advances in Applied Mathematics **131** (2021).

[18] Gérard Viennot, *Algèbres de lie libres et monoïdes libres : Bases des algèbresde lie libres et factorisations des monoïdes libres*. Vol. 691, Springer, Berlin, 1978.

[19] Wikipédia, *Encyclopédie en ligne des suites de nombres entiers*, 8 août 2023, disponible via l'URL <https://fr.wikipedia.org/wiki/Encyclop%C3%A9die_en_ligne_des_suites_de_nombres_entiers>, consulté le 29 septembre 2023.

[20] Wikipédia, *Hall words*, anglais, 12 mars 2023, disponible via l'URL <https://en.wikipedia.org/wiki/Hall_word#>, consulté le 27 juin 2023.