Explorations combinatoires des structures arborescentes et libres

Christophe Cordero

Thèse de doctorat

Explorations combinatoires des structures arborescentes et libres

pour l'obtention du grade de

Docteur de l'université Paris-Est

Spécialité Informatique

École Doctorale de Mathématiques et des Sciences et Techniques de l'Information et de la Communication

Présentée et soutenue publiquement par

Christophe Cordero

le 9 décembre 2019

Devant le jury composé de

Marie-Pierre Béal	Examinatrice
Bérénice Delcroix-Oger	Examinatrice
Samuele Giraudo	Directeur de thèse
Jean-Gabriel Luque	Examinateur
Dominique Manchon	Rapporteur
Jean-Christophe Novelli	Directeur de thèse
Michaël Rao	Examinateur
Christophe Reutenauer	Rapporteur

Laboratoire d'informatique Gaspard-Monge
UMR 8049 LIGM
5, bd Descartes, Champs-sur-Marne, 77454 Marne-la-Vallée Cedex 2, France

Résumé et contributions

Notre travail aborde trois axes de la combinatoire algébrique et énumérative.

Le premier concerne principalement l'exploration informatique de la théorie des codes. Étant donné un alphabet binaire $\mathcal{A} := \{a, b\}$, on note $\mathcal{A}^*$ l'ensemble des mots finis sur cet alphabet. Un ***code*** est un sous-ensemble $\mathcal{X}$ de $\mathcal{A}^*$ tel que pour tout $x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathcal{X}$ l'égalité

$$x_1 \cdots x_n = y_1 \cdots y_m$$

implique que

$$n = m \text{ et } x_i = y_i$$

pour tout $i \in [1, n]$. On dit qu'un code est ***préfixe*** si aucun de ses éléments n'est le préfixe d'un autre élément. Plus généralement, on dit qu'un ensemble $\mathcal{X}$ est ***commutativement préfixe*** s'il existe un code préfixe $\mathcal{P}$ tel que

$$\sum_{x \in \mathcal{X}} \alpha(x) = \sum_{p \in \mathcal{P}} \alpha(p),$$

où α est l'application qui associe à un mot son image commutative.

Dans les années soixante, Schützenberger a conjecturé que tout code maximal (c'est-à-dire non inclus dans un autre code) fini est commutativement préfixe. Cette conjecture, nommée *commutativement équivalent*, est considérée comme l'une des principales conjectures du domaine de la théorie des codes. Nous sommes motivés par la recherche d'un contre-exemple. Cependant, le seul exemple de code non commutativement préfixe (et non maximal) qu'on connaissait avait été trouvé par Shor en 1984. La stratégie que nous adoptons est de d'abord trouver des codes non commutativement préfixes puis de chercher des codes maximaux finis qui les contiendraient.

Nous donnons dans le théorème 2.1.1 une suite d'inégalités qui raffine l'inégalité de Kraft-Redheffer et qui caractérise les ensembles commutativement préfixes. Cette caractéristique fournit un algorithme quadratique en la longueur du plus long mot pour tester si un ensemble (*a fortiori* un code) donné

est commutativement préfixe. Celui-ci nous a permis avec l'aide de Michaël Rao de montrer par une recherche exhaustive à l'ordinateur qu'il n'existe pas de code non commutativement préfixe dont les mots sont de longueurs au plus 6.

L'espace de recherche étant trop vaste, nous nous sommes restreints aux codes ***baïonnettes***, c'est-à-dire dont les mots sont de la forme $a^i b a^j$. Nous justifions dans la proposition 2.1.4 un algorithme de graphes qui nous a permis de découvrir 69 nouveaux codes baïonnettes non commutativement préfixes. Nous exhibons, entre autres, dans la table 2.1.2 et dans la table 4.36 de l'annexe A, tous les codes baïonnettes non commutativement préfixes dont les mots sont de longueurs inférieures ou égales à 15. Certains de ces codes ont amélioré la borne inférieure qui était connue pour le problème du *ratio* de Shor (2.8).

Il suffit qu'un de ces codes soit inclus dans un code maximal fini pour contredire la conjecture *commutativement équivalent*. Cependant, on ne sait pas s'il existe un algorithme capable de déterminer si un code donné est inclus dans un code maximal fini. On sait cependant que certains codes ne sont pas inclus dans des codes maximaux finis, en particulier le code

$$\{aaaaa, ab, b, baa\}$$

que l'on doit à Antonio Restivo. Nous proposons dans la conjecture 2.2.12 un plus petit (au sens de la longueur de son plus long mot) code de ce type et montrons qu'il n'en existe pas de plus petit.

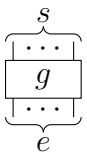
On sait que pour être inclus dans un code maximal fini, un code doit nécessairement se projeter dans une *factorisation d'un groupe cyclique*. On ignore toujours si c'est le cas pour le code de Shor. Nous montrons néanmoins que 7 de nos codes vérifient cette propriété en se projetant dans les factorisations (2.10) et (2.11). En revanche, nous échouons à trouver des codes maximaux finis les contenant. Grâce aux raisonnements classiques de la théorie des factorisations des groupes cycliques, nous avons pu calculer des bornes inférieures sur les longueurs des plus longs mots des codes maximaux finis susceptible de les contenir. Nous les exposons dans la table 2.2.1.

Nous proposons une nouvelle approche pour la conjecture *commutativement équivalent* restreinte aux codes baïonnettes dans la partie 2.2.2. Nous y introduisons la notion de *code modulaire baïonnette complet*. Cette notion nous permet notamment de calculer des bornes inférieures sur les longueurs des plus longs mots des codes maximaux finis pouvant contenir nos codes et d'obtenir, dans le théorème 2.2.7, les premiers exemples de codes non commutativement préfixes et non inclus dans des codes maximaux finis.

Le deuxième axe de recherche porte sur la combinatoire des *prographes*.

Les ***prographes*** sont classiquement définis par la grammaire récursive suivante :

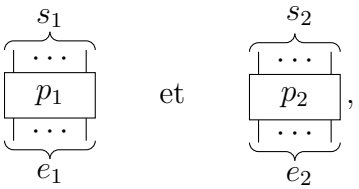
— un ***générateur***



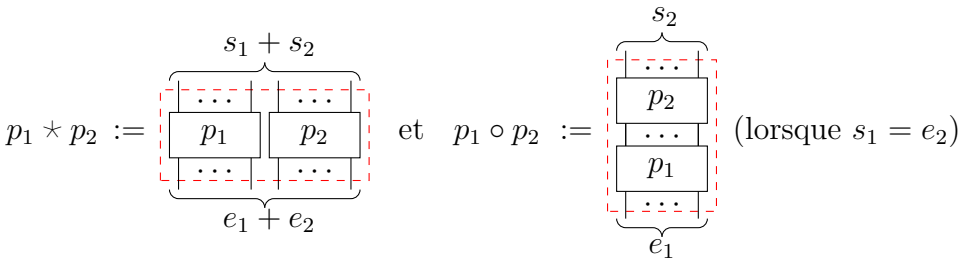
à e entrées et s sorties est un prographe à e entrées et s sorties ;

— le fil | est un prographe à une entrée et une sortie ;

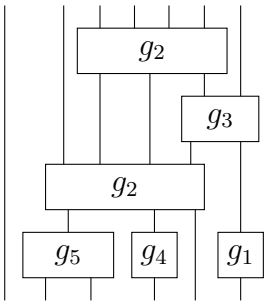
— étant donnés deux prographes



les assemblages



sont des prographes. Le prographe $p_1 \star p_2$ a $e_1 + e_2$ entrées et $s_1 + s_2$ sorties. Si $s_1 = e_2$ alors $p_1 \circ p_2$ est le prographe à e entrées et s sorties où on a branché la première sortie de p_1 à la première entrée de p_2 et ainsi de suite. Par exemple, le prographe



possède 6 entrées, 7 sorties et est composé de 6 générateurs (g_1, deux générateurs g_2, g_3, g_4 et g_5).

Nous proposons au cours du chapitre 3 diverses réponses à la question : étant donnés un ensemble de générateurs $\mathbb{G}$ et des entiers e, n et s, combien y a-t-il de prographes à e entrées, s sorties et composés d'exactement n générateurs tous issus de l'ensemble $\mathbb{G}$?

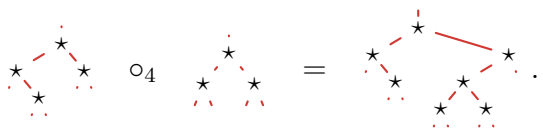
La principale difficulté pour répondre à cette question vient de l'ambiguïté de la grammaire qui engendre les prographes. Nous présentons dans la partie 3.1.2 une grammaire non ambiguë qui engendre les prographes en montrons au passage, dans le théorème 3.1.13, qu'ils forment des objets algébriques libres appelés *pros* libres.

Nous déduisons de cette nouvelle grammaire le théorème 3.2.4 qui exhibe une bijection entre les prographes et des familles de chemins colorés en trois dimensions, formalisées dans la définition 3.2.3. Par une étude classique de la combinatoire des chemins, nous obtenons des formules de récurrence sur les prographes dans la proposition 3.3.1 et dans le théorème 3.3.2. Ainsi qu'une équation fonctionnelle pour la série génératrice des prographes (3.23) dans le théorème 3.3.6. Nous exhibons dans l'exemple 3.3.7, une manière de modéliser par les prographes la notion biologique d'arbre tandem de duplication. Enfin, nous découvrons deux formules closes pour les prographes, n'utilisant qu'un type de générateurs, dans les théorèmes 3.3.8 et 3.3.9.

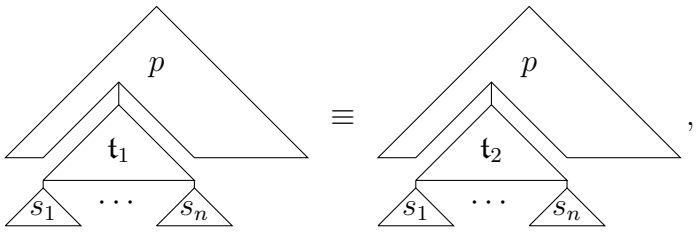
Le dernier axe de recherche de ce manuscrit porte sur la réécriture dans des *quotients magmatiques*. L'***opérade magmatique*** **Mag** est l'ensemble de tous les arbres binaires, doté des compositions partielles $\circ_i$ suivantes : étant donnés deux arbres $\mathfrak{t}$ et $\mathfrak{s}$, l'arbre

$$\mathfrak{t} \circ_i \mathfrak{s}$$

est l'arbre binaire obtenu en greffant l'arbre $\mathfrak{s}$ sur la i^e feuille de l'arbre $\mathfrak{t}$. Par exemple,



Si n est le nombre de feuilles des arbres $\mathfrak{t}_1$ et $\mathfrak{t}_2$ alors on appelle ***relation de congruence*** engendrée par $\mathfrak{t}_1 \equiv \mathfrak{t}_2$ la plus petite relation d'équivalence définie sur les arbres par



pour tous les arbres binaires $s_1, \ldots, s_n$ et p. On note

$$\mathbf{Mag}\Big/_{\langle \mathfrak{t}_1 \equiv \mathfrak{t}_2 \rangle}$$

l'espace ***quotient*** de **Mag** par la relation de congruence engendrée par $\mathfrak{t}_1 \equiv \mathfrak{t}_2$.

Pour tout entier $\gamma \geq 1$, les arbres ***peigne gauche*** et ***peigne droit*** de degré γ sont respectivement les arbres

$\mathfrak{p}^{(\gamma)} :=$  et $\mathfrak{p}^{(\gamma)} :=$  .

Nous appelons ***opérade peigne*** de taille γ l'opérade quotient

$$\mathbf{CAs}^{(\gamma)} := \mathbf{Mag}\Big/_{\left\langle \mathfrak{p}^{(\gamma)} \equiv \mathfrak{p}^{(\gamma)} \right\rangle}.$$

Nous caractérisons dans la proposition 4.2.3 tous les morphismes entre les opérades peignes. Cette caractéristique fournit une structure de treillis sur ces opérades que nous exposons dans le théorème 4.2.4. Nous dessinons une partie de ce treillis dans la figure (4.5).

Nous avons ensuite étudié individuellement des opérades magmatiques. Nous montrons dans le théorème 4.2.5 une présentation finie, confluente et terminante de $\mathbf{CAs}^{(3)}$. Nous déduisons de cette présentation la proposition 4.2.6 qui explicite la série de Hilbert de $\mathbf{CAs}^{(3)}$. En développant cette série nous obtenons la formule close (4.20) pour le nombre de classes d'équivalences d'arité n. Nous terminons notre étude de $\mathbf{CAs}^{(3)}$ en exhibant ses formes normales (4.22) et (4.23) pour notre présentation.

Dans la partie 4.2.2, nous conjecturons à partir de plusieurs explorations informatiques qu'il n'existe pas de présentation finie, confluente et terminante de $\mathbf{CAs}^{(\gamma)}$ quand $\gamma \geq 4$ et où les membres gauches et droits des règles de réécriture appartiennent à **Mag**.

Nous effectuons dans la partie 4.3 une étude exhaustive des 10 quotients magmatiques du type

$$\mathbf{Mag}\Big/_{\langle \mathfrak{t}_1 \equiv \mathfrak{t}_2 \rangle},$$

où $\mathfrak{t}_1$ et $\mathfrak{t}_2$ sont des arbres binaires à 3 nœuds. Nous répartissons ces quotients en six classes d'équivalence (4.24). L'une d'entre elles est réduite à $\mathbf{CAs}^{(3)}$. Pour quatre autres, nous donnons des présentations finies, confluentes et terminantes. Nous exposons également leurs séries de Hilbert (4.3.2) et des formules closes (4.3.3). Nous concluons l'étude de ces quatre classes d'équivalence en donnant des réalisations combinatoires dans les théorèmes 4.3.4, 4.3.5,

4.3.6 et 4.3.7. Pour la dernière classe d'équivalence, nous conjecturons dans la partie 4.3.3 qu'elle n'admet pas de présentation finie, confluente et terminante.

Table des matières

Résumé et contributions **3**

Préface **11**

1 Introduction **13**
1.1 Un problème de communication 13
1.1.1 L'inégalité de Kraft-Redheffer 16
1.1.2 Non commutativement préfixe 17
1.1.3 Code maximal 18
1.2 Combinatoire des circuits 19
1.2.1 Les arbres tandems de duplication 22
1.2.2 Chemins combinatoires 24
1.3 Relations arborescentes 28
1.3.1 Relation d'associativité 29
1.3.2 Espace quotient 30
1.3.3 Réécriture dans les quotients 32
Plan du manuscrit 34

2 Une exploration informatique de la théorie des codes **37**
2.1 Codes non commutativement préfixes 39
2.1.1 Le cas général 39
2.1.2 Les codes baïonnettes 41
2.2 Inclusion dans un code maximal fini 46
2.2.1 Factorisations des groupes cycliques 46
2.2.2 Codes baïonnettes modulaires complets 49
2.2.3 Plus petits codes non inclus dans un code maximal fini 54

3 Combinatoire des prographes **57**
3.1 Une construction des prographes 60
3.1.1 Intuition sur la construction 60
3.1.2 Présentation de la construction 63

3.1.3 Démonstration de la construction 66
3.2 Prographes et chemins combinatoires 72
3.2.1 Chemins combinatoires 72
3.2.2 Une bijection entre les prographes et des chemins . . . 73
3.3 Conséquences combinatoires 75
3.3.1 Formules de récurrence 75
3.3.2 Équation fonctionnelle 78
3.3.3 Formules closes 81

4 Réécriture dans des quotients magmatiques 85
4.1 Opérades et réécriture 85
4.1.1 Opérade 85
4.1.2 Arbre binaire et opérade magmatique 87
4.1.3 Réécriture dans les arbres binaires 88
4.2 Opérades peignes 90
4.2.1 Le treillis des opérades peignes 90
4.2.2 Complétions des opérades peignes 94
4.3 Quotients cubiques 100
4.3.1 Classes anti-isomorphes 100
4.3.2 Réalisations combinatoires 101
4.3.3 Quotients aux présentations possiblement infinies . . . 107

Perspectives 109

Annexes 111

Bibliographie 117

Préface

Combinatoire

La ***combinatoire*** est un vaste domaine. Il n'est pas simple d'en donner une définition. S'il nous fallait en énoncer une, nous dirions sans doute que la combinatoire est l'ensemble des techniques qui servent à dénombrer ou énumérer des ensembles paramétrés finis.

Elle se scinde historiquement en plusieurs branches dont la combinatoire ***énumérative*** et la combinatoire ***algébrique***. La combinatoire énumérative s'intéresse principalement au dénombrement. Un de ses objectifs est de proposer des formules pour compter des objets combinatoires. Évaluer la pertinence d'une de ces formules n'est en général pas automatique. Cela relève le plus souvent d'un art qui prend en compte divers paramètres tels que la complexité de la formule, ses liens avec d'autres objets combinatoires, sa concision, ... Une de ses autres branches, la combinatoire algébrique, fait le pont entre la combinatoire et l'algèbre. Elle applique, entre autres, les techniques combinatoires aux structures algébriques et inversement dote les objets combinatoires de structures algébriques.

Contexte et projet

Nos travaux empruntent ces deux branches de la combinatoire sur trois types d'objets différents. Du plus spécifique au plus général, il s'agit des mots, des arbres (binaires) et des circuits (ou prographes). Ces objets sont les éléments de structures algébriques respectivement appelées ***monoïdes***, ***opérades*** et ***pros***. La recherche combinatoire sur celles-ci est un sujet vaste et varié. On peut classer ces structures en au moins deux catégories : les *libres* et les non *libres*.

Une structure (telle qu'un monoïde, une opérade, un pro, ...) est ***libre*** quand chaque relation entre ses éléments découle des axiomes définissant son type de structure. Les monoïdes libres sont en quelque sorte les versions les plus génériques des monoïdes. Les autres monoïdes peuvent être fabriqués en ajoutant des relations aux monoïdes libres. En algèbre, on dit qu'on « quo-

tiente » un monoïde libre par des relations afin d'en définir un autre. Les dénombrements des monoïdes libres et des opérades libres reviennent respectivement à nombrer les mots sur un alphabet donné et les arbres pour un ensemble de nœuds donné. Il s'agit d'exercices combinatoires élémentaires, bien connus et maîtrisés. Cependant, le dénombrement des pros libres qui engendrent les circuits est plus ardu et n'avait pas été traité. Nous y répondons dans le chapitre 3 qui reprend en partie notre publication [Cor18].

L'étude énumérative des monoïdes, opérades et pros non libres est plus difficile et éclectique. Même en se restreignant aux monoïdes, on est le plus souvent contraint de les étudier au cas par cas. Ceci est devenu un domaine de la combinatoire des mots [Lot97]. La combinatoire énumérative des opérades a généralisé aux arbres des théories, comme celle de la réécriture [BN99], originellement développées sur les mots. Nous apportons notre contribution à ce domaine dans le chapitre 4 en étudiant, principalement sous l'angle de la réécriture, des familles de quotients d'opérades. Cet apport provient de nos articles [CCG19, CCG18]. La réécriture dans les pros [Laf09] ou plus généralement l'étude énumérative des pros non libres reste à notre connaissance peu étudiée.

Toutes ces structures peuvent contenir des sous-structures. Par exemple, les monoïdes inclus dans d'autres monoïdes sont appelés des sous-monoïdes. Une partie de la combinatoire algébrique tente d'exhiber les critères qui garantissent que la liberté d'une structure se transmette à ses sous-structures. C'est-à-dire, sous quelles conditions les monoïdes, opérades et pros inclus dans d'autres structures libres du même type sont libres. Ce domaine restreint aux monoïdes est appelé la *théorie des codes*. Elle a été édifiée entre les années 60 et 90 [BPR10]. Certains problèmes originels qui ont motivé le développement de cette théorie sont toujours ouverts. Cependant, nous avons de nos jours une puissance de calcul qui nous permet de nouvelles approches. Nous exposons cette démarche dans le chapitre 2 qui englobe notre publication [Cor19]. La théorie des codes a été en partie généralisée aux opérades [MR01] et n'a, à notre connaissance, pas encore été extrapolée aux pros.

La programmation informatique va nous accompagner sous diverses formes au cours de ce manuscrit. Pour la combinatoire énumérative, elle nous aidera à conjecturer des formules et des liens entre des objets avant de les démontrer ; tandis qu'elle nous servira en combinatoire algébrique à trouver des objets vérifiant certaines contraintes ou à prouver leurs inexistences.

Chapitre 1

Introduction

Commençons par l'étude d'un problème de combinatoire algébrique issu de la théorie de la communication. Bien que connu, le lien entre ces deux domaines est souvent évoqué de manière lacunaire ou est tout simplement ignoré dans la littérature.

1.1 Un problème de communication

Nous souhaitons organiser la transmission de messages écrits dans une langue alphabétique à travers un canal (sans perte) n'acceptant que des suites binaires. Appelons S et JC [1] les deux personnages souhaitant communiquer à travers ce canal. Pour cela, ils peuvent se mettre d'accord, a priori, sur une table qui encode chaque caractère de leur alphabet en une suite binaire finie. Par exemple, si S et JC choisissent la table d'encodage

Lettre	Encodage
a	0011
b	01000
c	1
d	10

(1.1)

alors pour transmettre le message *bac* il leur faudra envoyer sur le canal la suite binaire

$$0100000111.$$

De même, si S envoie *cbcc* à JC alors JC recevra la suite binaire

$$10100011. \tag{1.2}$$

1. Les lettres a et b étant déjà utilisées dans cette introduction, nous avons tiré ces initiales au hasard.

Cependant, cette suite binaire est également l'encodage du message *dda*. La communication est de ce fait ambiguë ! Ils ne peuvent donc pas utiliser n'importe quelle table d'encodage. Nous appelons ***code*** un ensemble de suites binaires tel que chaque suite binaire admette au plus une décomposition en éléments de l'ensemble. L'ensemble

$$\{0011, 01000, 1, 10\},$$

issu de la table (1.1), n'est donc pas un code puisque que nous savons que la suite (1.2) admet au moins deux décompositions différentes. En effet,

$$10100011 = (1)(01000)(1)(1) = (10)(10)(0011).$$

La communication sera non-ambiguë si et seulement s'ils choisissent une table d'encodage dont l'ensemble des suites binaires d'encodage forme un code.

Une façon simple de construire un code est d'utiliser la notion de *préfixe*. On dit qu'une suite binaire x est ***préfixe*** d'une autre suite binaire y s'il existe une suite binaire u telle que

$$y = xu.$$

Par exemple, la suite 01 est un préfixe de la suite 011101 mais pas de la suite 110. Par extension, on dit qu'un ensemble de suites binaires est ***préfixe*** si aucune de ses suites n'est le préfixe d'une autre suite de l'ensemble. Par exemple, l'ensemble infini

$$\{1, 01, 001, 0001, 00001, \dots\}$$

est un ensemble préfixe. En effet, aucun élément de la forme

$$\underbrace{0 \cdots 0}_{n} 1 \qquad (\text{pour } n \geq 0)$$

n'est le préfixe d'un autre élément de cette forme. Un ensemble préfixe est nécessairement un code car si une suite binaire a deux décodages différents alors le premier élément d'une des décompositions est le préfixe du premier élément de l'autre décomposition. Par exemple, dans la table (1.1), l'encodage de la lettre c est un préfixe de l'encodage de la lettre d.

Nous savons maintenant, par exemple, que les tables

Lettre	Encodage
a	1
b	01
c	001
d	0001

(1.3) et

Lettre	Encodage
a	00
b	01
c	10
d	11

(1.4)

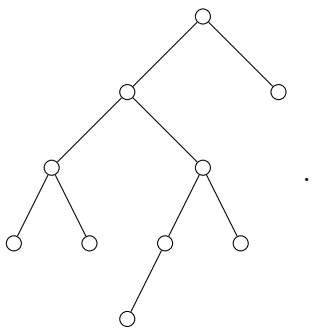
permettent une communication non-ambiguë. Un code n'est pas nécessairement préfixe. Par exemple, l'ensemble

$$\{1, 101\} \tag{1.5}$$

est un code non préfixe. Démontrons que l'ensemble (1.5) est un code. Si un message encodé contient plus de deux chiffres et que le deuxième est un 0 alors nécessairement la première lettre est un b sinon c'est un a. Il suffit de retirer l'encodage de la lettre que l'on vient de décoder au message et de recommencer l'opération. Chaque suite binaire aura donc au plus une décomposition en éléments de l'ensemble (1.5).

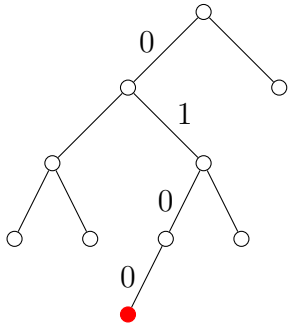
La table (1.4) est issue d'une famille particulière de codes préfixes. En effet, on peut plus généralement encoder toutes les lettres par des suites binaires de longueur n, où n est le plus petit entier tel que 2^n soit supérieur ou égal au nombre de lettres. Cependant, on ne se contente pas de ce type de code car le fait d'autoriser des suites d'encodages de longueurs différentes peut permettre de compresser les messages. Par exemple, en français la lettre e est plus fréquente que la lettre w. Il serait donc judicieux de choisir une suite plus courte pour encoder la lettre e que pour encoder la lettre w.

On peut facilement engendrer les codes préfixes. Il suffit de prendre un arbre binaire, par exemple :



. (1.6)

Puis pour chaque chemin de la racine à une feuille on produit une suite binaire grâce au mécanisme suivant : on écrit 0 quand on emprunte une branche gauche et 1 quand on emprunte une branche droite. Par exemple, la feuille colorée en rouge



va produire la suite 0100. Le code préfixe engendré par un arbre sera l'ensemble des suites ainsi produites. L'arbre (1.6) engendre donc le code préfixe

$$\{000, 001, 0100, 011, 1\}\,.$$

Il n'est pas difficile de montrer que tous les codes préfixes peuvent être ainsi engendrés.

Un des intérêts de l'utilisation des codes préfixes est qu'ils permettent de décoder à la volée. C'est-à-dire qu'il n'est pas nécessaire de recevoir l'intégralité du message pour commencer à le décoder. En effet, si la table d'encodage utilise un code préfixe alors JC peut décoder le message par le procédé suivant : la première lettre du message est nécessairement l'unique lettre dont la suite d'encodage est un préfixe du message. Il peut ainsi retirer l'encodage de la lettre et recommencer le processus. Par exemple, si S et JC utilisent la table (1.3) et que JC reçoit l'octet

$$00010100$$

en provenance de S alors même si le message est nécessairement incomplet, JC sait déjà qu'il commence par *db*.

Notons qu'il existe en général des algorithmes, comme celui de Sardinas–Patterson [SP53], pour tester si un ensemble est un code. Mais quel code choisir pour communiquer ? Les codes préfixes sont-ils suffisants ? Comme nous allons le voir dans les trois sous-parties suivantes, cela dépend des contraintes que l'on souhaite satisfaire.

1.1.1 L'inégalité de Kraft-Redheffer

En 1949, Leon Kraft et Raymond Redheffer [2] [Kra49] ont démontré que si l'ensemble

$$\{x_1, x_2, \ldots, x_n\}$$

2. Le résultat qui suit est souvent nommé *inégalité de Kraft* dans la littérature. Cependant, à la page 20 de sa thèse [Kra49], Kraft attribue l'inégalité à Redheffer.

est un code alors nécessairement

$$\sum_{i=1}^{n} 2^{-|x_i|} \leq 1,$$

où la notation $|x|$ désigne la longueur de la suite x. Ils montrent réciproquement, qu'étant donnés des entiers $\ell_1, \ldots, \ell_n \geq 0$ tels que

$$\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$$

alors il existe un code préfixe

$$\{p_1, p_2, \ldots, p_n\}$$

tel que

$$|p_i| = \ell_i$$

pour tout $i \in [1, n]$.

On en déduit que pour tout code il existe un code préfixe dont les suites ont les mêmes longueurs. Si on souhaite optimiser un code par la longueur de ses suites alors on peut tant qu'à faire choisir un code préfixe et profiter de leurs propriétés, vues dans la partie précédente.

1.1.2 Non commutativement préfixe

Il se peut cependant que la transmission sur le canal binaire des chiffres 0 et 1 ne soit pas homogène. Par exemple, l'envoi d'un 1 peut correspondre à une impulsion électrique et celui d'un 0 à une pause. La transmission d'un 1 consomme donc plus d'énergie que celle d'un 0. Peut-on tout de même se restreindre aux codes préfixes ? C'est-à-dire, étant donné un code, existe-t-il nécessairement un code préfixe avec le même coût énergétique ?

Plus formellement, Perrin et Schützenberger ont demandé oralement dans les années soixante-dix si étant donné un code

$$\{x_1, x_2, \ldots, x_n\},$$

il existait nécessairement un code préfixe

$$\{p_1, p_2, \ldots, p_n\}$$

tel que

$$|x_i|_0 = |p_i|_0 \text{ et } |x_i|_1 = |p_i|_1$$

pour tout $i \in [1, n]$. La notation $|x|_j$ désigne le nombre d'occurrences du chiffre j dans la suite x.

En 1984, Shor a répondu non en fournissant un exemple [Sho85]. On appelle ce type de code des codes non ***commutativement préfixes***, car on ne peut pas construire de code préfixe à partir de ces codes en permutant les chiffres de leurs suites. Jusqu'à notre article [Cor19], l'exemple de Shor était le seul code non commutativement préfixe connu. Nous l'expliquons plus en détail dans la partie 2.1.

1.1.3 Code maximal

Si on peut ajouter des éléments à la table d'encodage de telle façon que la communication reste non ambiguë (c'est-à-dire que l'ensemble des suites d'encodages reste un code) alors on ne profite pas de toute la « capacité » de la communication. Même si le code qu'on utilise a suffisamment d'éléments pour encoder notre alphabet, il peut être intéressant d'en ajouter. Par exemple, en français la lettre q est très souvent suivie par la lettre u, on pourrait donc encoder la suite qu par une suite plus courte que la suite concaténant les encodages de q et de u.

Un code qui n'est pas inclus dans un autre code est dit ***maximal***. Par exemple, le code

$$\{00, 01, 10, 11\}$$

issu de la table d'encodage (1.4) est maximal. En effet, on ne peut pas y ajouter une suite de longueur paire car celle-ci serait décomposable en élément du code et on ne peut pas non plus y ajouter une suite de longueur impaire car la concaténation de la suite avec elle-même serait de longueur paire et donc décomposable. On ne peut donc pas ajouter de lettre à la table d'encodage (1.4).

Bien évidement, tout sous-ensemble d'un code (maximal) est un code. Pour cette raison, une grande partie de la théorie des codes se ramène à l'étude des codes maximaux.

Cependant, on ne sait pas s'il existe un algorithme qui prend en entrée un code et répond si le code est inclus ou non dans un code maximal fini. Ce problème est appelé le *problème de l'inclusion*. Il existe en effet des codes finis non inclus dans des codes maximaux finis. Par exemple, Antonio Restivo a démontré que le code

$$\{00000, 01, 1, 100\}$$

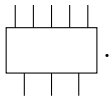
n'est pas inclus dans un code maximal fini [Res77].

Schützenberger a conjecturé[3] que tous les codes maximaux finis sont commutativement préfixes. On l'appelle la conjecture *commutativement équivalent*. Si elle est vraie alors cela signifie que pour des communications utilisant des codes maximaux finis, on peut tant qu'à faire choisir un code préfixe et cela même si la transmission sur le canal n'est pas homogène. Le chapitre 2 de ce manuscrit est principalement consacré à la recherche d'un contre-exemple à la conjecture. Remarquons que sans l'hypothèse *maximal*, la conjecture est fausse d'après l'exemple de Shor, énoncé dans la partie 1.1.2.

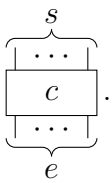
Nous verrons au début du chapitre 2 que la théorie combinatoire des codes coïncide avec la théorie algébrique des *monoïdes* et *sous-monoïdes libres*. Le deuxième thème que nous étudierons dans ce manuscrit est la combinatoire des *circuits*. Les circuits sont également des objets libres d'une structure algébrique et ils sont plus généraux que les suites finies de bits que nous venons d'étudier.

1.2 Combinatoire des circuits

Les circuits sont composés d'éléments insécables que l'on appelle *composants*. Un ***composant*** est une boîte munie d'entrées et de sorties. Voici un composant avec 3 entrées et 5 sorties :

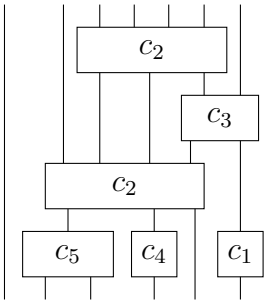


Plus généralement, un composant c qui possède e entrées et s sorties se dessine ainsi :



3. Nous ne connaissons pas la date précise de découverte de la conjecture. Cependant, Schützenberger l'a présentée en 1965 [Sch65].

On peut combiner des composants de façon planaire (sans que les fils ne se croisent) afin d'obtenir des circuits. Par exemple,

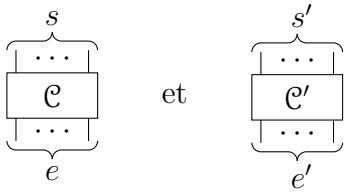


est un circuit qui possède 6 entrées, 7 sorties et qui est constitué de 6 composants (c_1, deux composants c_2, c_3, c_4 et c_5).

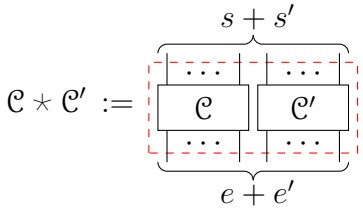
Les circuits permettent de modéliser des objets aussi variés que des circuits électroniques imprimés (le fait que les connections ne se croisent pas permet d'éviter les courts-circuits), des calculs algébriques ou bien des systèmes de réécriture [Laf09]. Nous verrons également dans cette introduction qu'ils permettent de modéliser la notion biologique d'arbres tandems de duplication [Gas05].

On peut redéfinir les ***circuits*** de façon un peu plus rigoureuse à l'aide de la grammaire récursive :

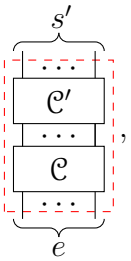
— un composant à e entrées et s sorties est un circuit à e entrées et s sorties ;
— le fil | est un circuit à une entrée et une sortie ;
— étant donnés deux circuits



l'accolement



forme un circuit à $e+e'$ entrées et $s+s'$ sorties ; si $s = e'$ alors le branchement

$\mathcal{C} \circ \mathcal{C}' :=$ 

,

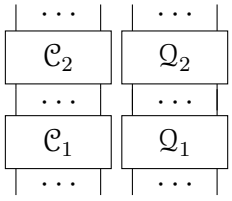
où on branche la première sortie de $\mathcal{C}$ avec la première entrée de $\mathcal{C}'$ et ainsi de suite, forme un circuit à e entrées et s' sorties.

Nous souhaitons dénombrer et énumérer les circuits. C'est-à-dire qu'étant donnés un ensemble de composants $\mathbb{C}$ et trois entiers e, n et s, nous souhaitons avoir des formules ou des méthodes pour compter et engendrer les circuits à e entrées, s sorties et composés d'exactement n composants tous issus de l'ensemble $\mathbb{C}$.

La principale difficulté que l'on va rencontrer vient du fait que la grammaire qui définit les circuits est ambiguë. C'est-à-dire qu'il y a en général des façons différentes de construire un même circuit. Formellement, étant donnés quatre circuits $\mathcal{C}_1, \mathcal{C}_2, \mathcal{Q}_1$ et $\mathcal{Q}_2$ tels que $\mathcal{C}_1 \circ \mathcal{C}_2$ et $\mathcal{Q}_1 \circ \mathcal{Q}_2$ soient bien définis, on a

$$(\mathcal{C}_1 \circ \mathcal{C}_2) \star (\mathcal{Q}_1 \circ \mathcal{Q}_2) = (\mathcal{C}_1 \star \mathcal{Q}_1) \circ (\mathcal{C}_2 \star \mathcal{Q}_2)\,.$$

Cette relation montre qu'il y a deux manières de construire le circuit



à partir des circuits $\mathcal{C}_1, \mathcal{C}_2, \mathcal{Q}_1$ et $\mathcal{Q}_2$. La première signifie que l'on peut d'un côté brancher $\mathcal{C}_1$ avec $\mathcal{C}_2$ et d'un autre brancher $\mathcal{Q}_1$ avec $\mathcal{Q}_2$, puis accoler les deux assemblages. La deuxième manière consiste à accoler $\mathcal{C}_1$ avec $\mathcal{Q}_1$ d'un côté et d'accoler $\mathcal{C}_2$ avec $\mathcal{Q}_2$ d'un autre côté, puis brancher les deux assemblages ensemble.

Il y a d'autres types d'ambiguïtés possibles lors de la construction d'un circuit. Il y a par exemple, deux manières d'accoler ou de brancher trois circuits ensemble. En algèbre, on dit que les opérations $\star$ et $\circ$ sont associatives. C'est-à-dire, que

$$(\mathcal{C}_1 \star \mathcal{C}_2) \star \mathcal{Q}_1 = \mathcal{C}_1 \star (\mathcal{C}_2 \star \mathcal{Q}_1) \text{ et que } (\mathcal{C}_1 \circ \mathcal{C}_2) \circ \mathcal{Q}_1 = \mathcal{C}_1 \circ (\mathcal{C}_2 \circ \mathcal{Q}_1)$$

lorsque ces opérations sont possibles. Il y a une dernière ambiguïté venant du fait que brancher un simple fil sur un autre ne change rien. Plus formellement, si $\mathcal{C}$ est un circuit avec e entrées et s sorties alors

$$\mathcal{C} \circ \overbrace{|\cdots|}^{s} = \mathcal{C} = \underbrace{|\cdots|}_{e} \circ \mathcal{C}.$$

Notre problème d'énumération des circuits est très général. Commençons par un cas particulier.

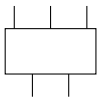
1.2.1 Les arbres tandems de duplication

Étudions le cas le plus simple parmi les cas encore non décrits dans la littérature. Car il y a en effet des cas particuliers de circuits que l'on connaît déjà. Par exemple, si on se restreint aux circuits dotés d'une seule entrée et n'utilisant que des composants

 (1.7)

alors les circuits que nous obtenons sont identiques aux arbres binaires [4], dont nous avons déjà parlé au sujet des codes préfixes et dont nous reparlerons plus en détail dans la partie suivante.

Commençons donc notre étude combinatoire sur les circuits possédant 2 entrées et n'utilisant que le composant

 . (1.8)

Ces circuits auront nécessairement $n+2$ sorties, où n est le nombre de composants dont il est constitué. En effet, ces circuits ont deux entrées et à chaque fois que nous leurs branchons un nouveau composant du type (1.8), celui-ci ajoute une sortie.

Dans ce cas très particulier, notre problème de dénombrement de circuits revient à étudier la suite d'ensembles A_n, indexée par le paramètre n et que nous définissons ainsi : pour un entier $n \geq 0$, nous appelons A_n l'ensemble des circuits possédant 2 entrées, $n+2$ sorties et exactement n composants, tous du type (1.8). Nous notons a_n le nombre d'éléments de A_n.

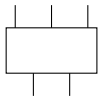
N'ayant a priori aucune connaissance sur cette suite nous allons essayer de mieux comprendre sa « mécanique » en calculant ses premiers termes. Le

4. avec la racine en bas et les feuilles en haut, comme dans la nature.

premier terme est l'ensemble A_0, celui-ci ne contient que le circuit réduit à 2 fils côte à côte

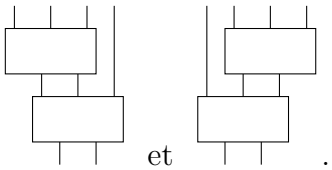
.

Nous en déduisons que a_0 vaut 1. De même, a_1 vaut 1 car l'ensemble A_1 ne contient que le circuit réduit à un composant



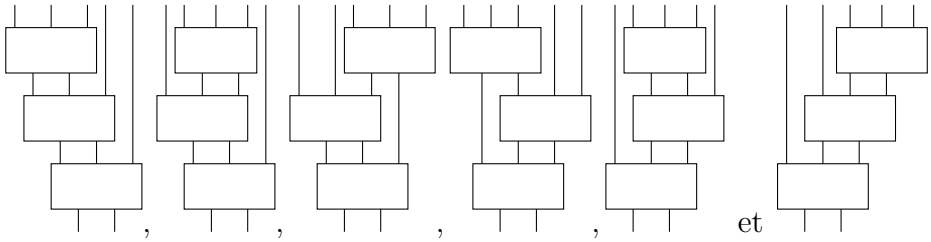
.

L'ensemble A_2 contient les deux circuits



et .

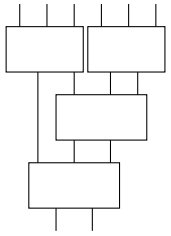
L'entier a_2 vaut donc 2. Nous trouvons les 6 circuits



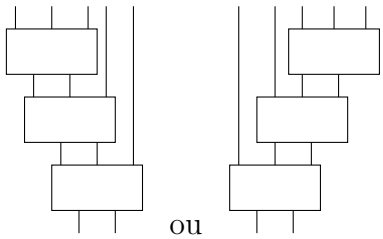
, , , , et

pour l'ensemble A_3, a_3 vaut donc 6.

À ce stade, on pourrait penser que la suite d'entiers a_n correspond tout simplement à la suite des factorielles n. Certes, nous avons jusqu'à présent construit les circuits de A_n en prenant ceux de A_{n-1} et en y ajoutant un composant de n manières possibles. Ce qui correspondrait aux factorielles. Cependant, à partir de $n = 4$, les problèmes d'ambiguïtés liés à la construction des circuits que nous avons exposé précédemment apparaissent. Par exemple, le circuit



sera construit deux fois, en ajoutant un composant à un des deux circuits



de l'ensemble A_3. Notre suite a_n est donc terme à terme plus petite (au sens large) que la suite factorielle.

Plus n sera grand et plus nous aurons du mal à gérer les ensembles A_n. Programmons donc cette énumération et récupérons tout simplement les premiers termes de la suite a_n. Voici les 13 premiers :

$$1, 1, 2, 6, 22, 92, 420, 2042, 10404, 54954, 298648, 1660714, 9410772. \tag{1.9}$$

Nous allons utiliser *l'encyclopédie des suites d'entiers* (dont le sigle anglais est OEIS) qui se trouve à l'adresse

https://oeis.org.

Comme son nom l'indique, il s'agit d'une encyclopédie collaborative dont l'objectif est de recenser les suites d'entiers utilisées par la communauté scientifique. Le site dispose d'un moteur de recherche, en y entrant notre suite (1.9), nous ne trouvons pas de référence aux circuits mais nous trouvons la référence $A264868$:

Number of rooted tandem duplication trees on n gene segments.

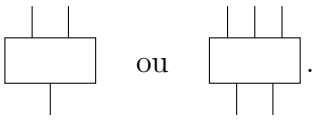
Cela signifie que pour les 13 premiers termes, il y a a_n arbres tandem enracinés de duplication sur $n+1$ gènes [Gas05]. Cela peut être une simple coïncidence. Le site OEIS nous indique des articles contenant la définition des arbres tandems et des formules combinatoires pour les dénombrer [GHJMM03]. Une de ces formules nous a d'ailleurs inspiré pour produire le théorème 3.3.2 sur le dénombrement des circuits en toute généralité. Nous démontrerons le lien entre les circuits et les arbres tandem dans l'exemple 3.3.7.

Étudions un autre cas particulier afin de trouver des liens avec d'autres objets combinatoires.

1.2.2 Chemins combinatoires

Rentabilisons notre programme qui énumère les circuits en recommençant l'énumération sur un exemple un peu plus complexe. Étudions la suite

des ensembles B_n qui contiennent les circuits à une entrée, $n+1$ sorties et composés de n composants du type



De même, nous notons b_n le nombre d'éléments contenus dans l'ensemble B_n. Nous obtenons ce coup-ci la suite

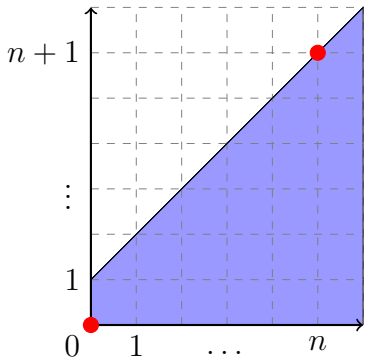
$$1, 1, 3, 14, 83, 568, 4271, 34296, 288946, 2524676, 22695611, 208713400.$$

Une nouvelle fois, ces termes correspondent à une suite enregistrée dans OEIS. La suite correspondante est la suite $A224776$ qui énumère des chemins combinatoires. Cette information est particulièrement alléchante car les chemins sont des objets classiques en combinatoire, facilement généralisables et bien moins exotiques que les circuits (voir le chapitre 10 de [Bón15] pour une synthèse).

La suite $A224776$ est définie comme la suite des nombres c_n qui sont les nombres de chemins du point $(0,0)$ aux points $(n, n+1)$, n'utilisant que les pas

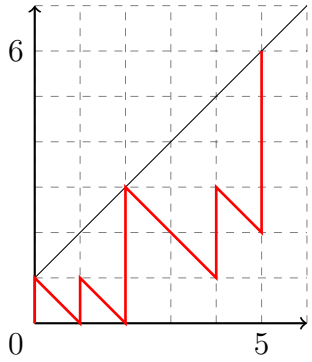
$$(0,1) \ : \ \uparrow \quad \text{et} \quad (1,-1) \ : \ \searrow, \tag{1.10}$$

restant au-dessus de l'axe des abscisses et restant en-dessous de la droite d'équation $y = x+1$. Graphiquement, un chemin va compter pour le nombre c_n s'il relie les points rouges



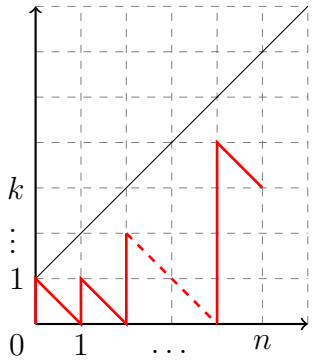
en utilisant les pas (1.10) et en restant dans la zone violette. Par exemple, le

chemin



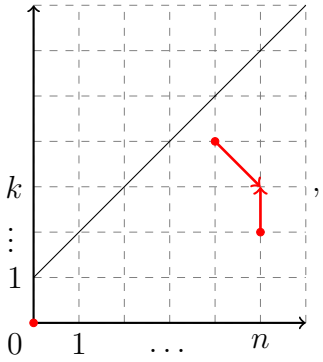
compte pour le nombre c_5.

On peut aisément calculer une formule de récurrence pour ces chemins. Afin de l'obtenir, on généralise la suite c_n en raffinant les chemins qu'elle compte. On définit $p_{n,k}$ comme le nombre de chemins du point $(0,0)$ au point (n,k) n'utilisant que les pas (1.10) et contraint à la même zone que les chemins associés à c_n. Graphiquement, le pseudo-chemin



va compter pour $p_{n,k}$.

La suite doublement indexée $p_{n,k}$ est bien un raffinement de la suite c_n, au sens où $p_{n,n+1} = c_n$. On peut remarquer, comme illustré dans la figure



qu'un de ces chemins non vide qui arrive au point (n, k) provient soit du point $(n-1, k+1)$ par le pas $(1, -1)$ soit du point $(n, k-1)$ par le pas $(0, 1)$. On déduit de cette remarque élémentaire que

$$p_{n,k} = p_{n-1,k+1} + p_{n,k-1},$$

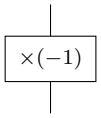
lorsque $n, k \geq 1$. Formellement, on obtient la relation de récurrence

$$p_{n,k} = \begin{cases} 1 & \text{si } n = k = 0, \\ p_{n,k-1} + p_{n-1,k+1} & \text{si } n \geq 0 \text{ et } 0 \leq k \leq n+1, \\ 0 & \text{sinon.} \end{cases} \tag{1.11}$$

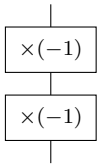
On se munit d'une formule de récurrence pour la suite c_n en spécialisant k à $n+1$.

Nous montrerons un lien formel entre les circuits et les chemins dans le théorème 3.2.4. La formule de récurrence (1.11) nous a également inspiré pour la proposition 3.3.1 qui s'applique à tous les circuits.

Nous venons d'explorer quelques pistes afin de dénombrer les circuits. Cependant, nos dénombrements ne prennent pas en compte les cas où les composants ont des relations entre eux autres que celles données par les règles d'assemblage. En termes algébriques, on dit que nous avons étudié des structures libres. Par exemple, si on choisit le composant



pour modéliser l'opération qui consiste à multiplier l'entrée par -1 alors on ne capture pas le fait que sémantiquement

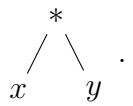


$= |$.

Ces circuits équivalents seront comptés et énumérés plusieurs fois. Dénombrer les circuits en prenant en compte une sémantique est un vaste et éclectique sujet. Nous allons l'aborder dans la dernière partie en nous restreignant aux arbres binaires.

1.3 Relations arborescentes

Soit $*$ une opération binaire associée à un espace $\mathbb{E}$. Pour $x, y \in \mathbb{E}$, on peut modéliser le calcul $x * y$ par l'arbre binaire

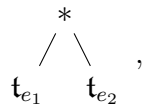


.

Plus généralement, on peut récursivement modéliser chaque calcul de l'espace $(\mathbb{E}, *)$ par un arbre binaire. En effet, une expression de ce type est soit un élément x de $\mathbb{E}$ que l'on représente par une feuille x soit un calcul de la forme

$$e_1 * e_2,$$

où e_1 et e_2 sont des calculs du même type, que l'on représente par l'arbre

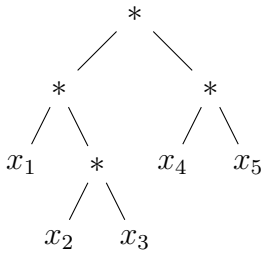


,

où $\mathbf{t}_{e_1}$ et $\mathbf{t}_{e_2}$ sont les arbres modélisant respectivement les expressions e_1 et e_2. Par exemple, l'expression

$$(x_1 * (x_2 * x_3)) * (x_4 * x_5)$$

est modélisée par l'arbre



. (1.12)

Comme pour les circuits dans la partie précédente, nous ne nous intéressons pas aux valeurs des entrées mais uniquement aux différentes compositions possibles des opérations. Pour l'espace $(\mathbb{E}, *)$, il n'y a qu'une opération et elle ne vérifie pas de relation particulière. Le dénombrement de toutes ces compositions revient donc à dénombrer l'ensemble des arbres binaires, que l'on note **Bin**. Comme nous l'avions évoqué, le dénombrement des arbres binaires avec pour paramètre le nombre de feuilles (qui correspond au nombre

d'entrées dans le circuit) est bien connu. Cela correspond à la suite de Catalan, dont la référence OEIS est A000108. Formellement, il y a

$$\frac{1}{n}\binom{2n-2}{n-1} = \frac{(2n-2)!}{(n-1)!n!}$$

arbres binaires à n feuilles. Une manière concise de répondre à la question du dénombrement est de donner la *série de Hilbert* de l'espace qu'on étudie, **Bin** dans le cas présent. C'est-à-dire, la série

$$\mathcal{H}_{\mathbf{Bin}}(t) := \sum_{n\geq 1} h_n t^n,$$

où h_n est le nombre d'arbres binaires à n feuilles. On a donc

$$\mathcal{H}_{\mathbf{Bin}}(t) = \sum_{n\geq 1} \frac{(2n-2)!}{(n-1)!n!}\, t^n = \frac{1-\sqrt{1-4t}}{2}.$$

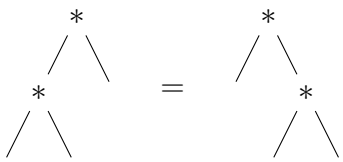
Nous allons maintenant étudier le cas où l'opération $*$ est associative.

1.3.1 Relation d'associativité

Si l'opération $*$ est associative, comme la concaténation de suites [5] ou les opérations $\star$ et $\circ$ d'assemblages des circuits [6], alors

$$(x * y) * z \;=\; x * (y * z) \tag{1.13}$$

pour tous les éléments $x, y, z \in \mathbb{E}$. Notre objectif est de prendre en compte ce type de relations dans nos dénombrements de tous les calculs possibles sur un espace donné, sans compter en double des expressions différentes mais équivalentes. Pour cela nous pouvons modéliser la relation (1.13) par la relation



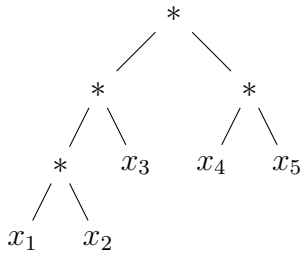
(1.14)

sur les arbres. Nous ne nommons pas les feuilles dans les arbres de la relation (1.14) car elle est vraie pour tous les triplets d'éléments.

Il faut noter qu'en supposant la relation (1.14) vraie, on doit l'appliquer localement dans n'importe quel arbre de n'importe quelle taille. On dit que

5. vue dans la première partie sur les codes.
6. vues dans la deuxième partie.

la relation (1.14) passe au contexte. Par exemple, l'arbre (1.12) est équivalent à l'arbre



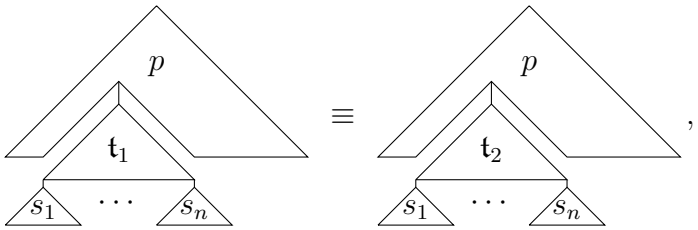
(1.15)

si la relation (1.14) est vraie.

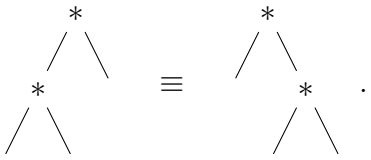
L'ensemble des arbres modélisant des calculs dont les opérations sont soumises à des relations est formalisé par la notion d'*espace quotient*.

1.3.2 Espace quotient

Soit n est le nombre de feuilles des arbres $\mathfrak{t}_1$ et $\mathfrak{t}_2$. On appelle ***relation de congruence*** engendrée par $\mathfrak{t}_1 \equiv \mathfrak{t}_2$ la plus petite relation d'équivalence définie sur les arbres par



pour tous les arbres binaires $s_1, \ldots, s_n$ et p. Par exemple, les arbres (1.12) et (1.15) sont équivalents pour la relation de congruence engendrée par



(1.16)

On note

$$\mathbf{Bin}\Big/_{\langle \mathfrak{t}_1 \equiv \mathfrak{t}_2 \rangle} \tag{1.17}$$

l'espace ***quotient*** de **Bin** par la relation de congruence engendrée par $\mathfrak{t}_1 \equiv \mathfrak{t}_2$. Les arbres (1.12) et (1.15) sont donc égaux dans l'espace quotient

$$\mathbf{Bin}\Big/_{\left\langle \begin{array}{c} * \\ * \end{array} \equiv \begin{array}{c} * \\ * \end{array} \right\rangle} . \tag{1.18}$$

L'espace (1.18) est généralement noté **As**. On l'appelle le ***quotient associatif*** car il modélise tous les calculs possibles qui n'utilisent qu'une opération binaire associative.

Dénombrons les arbres de **As**, c'est-à-dire l'ensemble des calculs possibles avec une opération binaire associative. Montrons par récurrence que pour tout $n \geq 1$, les arbres avec exactement n feuilles sont égaux dans **As**. C'est évident pour $n = 1, 2$ et 3. Supposons maintenant l'assertion vraie pour tous les entiers k tels que $k \leq n$. Soit deux arbres binaires $\mathfrak{t}_1$ et $\mathfrak{t}_2$ avec exactement $n+1$ feuilles chacun, ces arbres se décomposent ainsi

$$\mathfrak{t}_1 = \begin{array}{ccc} & * & \\ / & & \backslash \\ g_{\mathfrak{t}_1} & & d_{\mathfrak{t}_1} \end{array} \quad \text{et} \quad \mathfrak{t}_2 = \begin{array}{ccc} & * & \\ / & & \backslash \\ g_{\mathfrak{t}_2} & & d_{\mathfrak{t}_2} \end{array} ,$$

où $g_{\mathfrak{t}_1}, d_{\mathfrak{t}_1}, g_{\mathfrak{t}_2}$ et $d_{\mathfrak{t}_2}$ sont des arbres binaires. Notons respectivement m_1 et m_2 les nombres de feuilles des arbres $g_{\mathfrak{t}_1}$ et $g_{\mathfrak{t}_2}$. Si $m_1 = m_2$ alors par hypothèse de récurrence on a que $g_{\mathfrak{t}_1} \equiv g_{\mathfrak{t}_2}$ et $d_{\mathfrak{t}_1} \equiv d_{\mathfrak{t}_2}$, donc $\mathfrak{t}_1 \equiv \mathfrak{t}_2$. Si $m_1 \neq m_2$ alors on peut supposer sans perte de généralité que $m_1 > m_2$. Dans ce cas, on peut constater que

$$\mathfrak{t}_1 = \begin{array}{ccccc} & & * & & \\ & / & & \backslash & \\ & * & & & d_{\mathfrak{t}_1} \\ / & & \backslash & & \\ gg_{\mathfrak{t}_1} & & gd_{\mathfrak{t}_1} & & \end{array} \equiv \begin{array}{ccccc} & & * & & \\ & / & & \backslash & \\ gg_{\mathfrak{t}_1} & & & * & \\ & & / & & \backslash \\ & & gd_{\mathfrak{t}_1} & & d_{\mathfrak{t}_1} \end{array} , \tag{1.19}$$

où $gg_{\mathfrak{t}_1}$ et $gd_{\mathfrak{t}_1}$ sont des arbres binaires. On se ramène au cas précédent, en itérant $m_1 - m_2$ fois le procédé (1.19). On obtient dans tous les cas que $\mathfrak{t}_1 \equiv \mathfrak{t}_2$. On conclut la démonstration grâce au principe de récurrence.

Algébriquement, nous venons de prouver que tous les parenthésages du produit, par une opération associative, de n opérandes sont équivalents. Les parenthèses sont donc superflues dans ce cas.

On appelle ***série de Hilbert*** d'un espace quotient $\mathcal{Q}$, la série

$$\mathcal{H}_{\mathcal{Q}}(t) := \sum_{n \geq 1} h_n t^n,$$

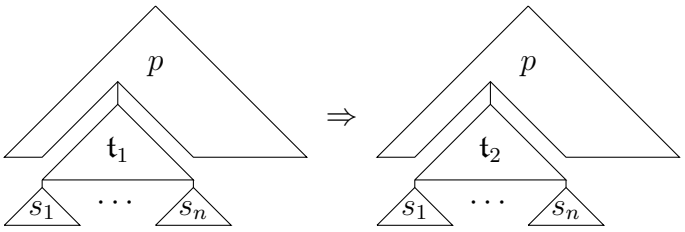
où h_n est le nombre d'arbres dans l'espace $\mathcal{Q}$ avec exactement n feuilles. Nous déduisons de la démonstration qui précède que la série de Hilbert de **As** est

$$\mathcal{H}_{\mathbf{As}}(t) = t + t^2 + t^3 + t^4 + \cdots = \frac{t}{1-t}\,.$$

Afin de montrer que deux arbres sont équivalents dans **As**, nous nous sommes contentés dans (1.19) de remplacer l'arbre à gauche de (1.14) par l'arbre à sa droite. Cette technique est formalisée par la notion de *règle de réécriture*.

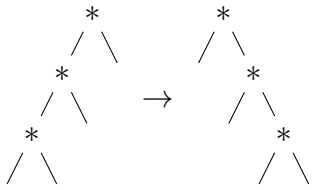
1.3.3 Réécriture dans les quotients

Un ensemble de ***règles de réécriture*** est un ensemble de couples d'arbres avec le même nombre de feuilles. On le note souvent $\to$ et on note $\mathfrak{t}_1 \to \mathfrak{t}_2$ le fait que $(\mathfrak{t}_1, \mathfrak{t}_2) \in \to$. On appelle ***relation de réécriture*** engendrée par $\to$, la plus petite relation $\Rightarrow$ définie sur les arbres par



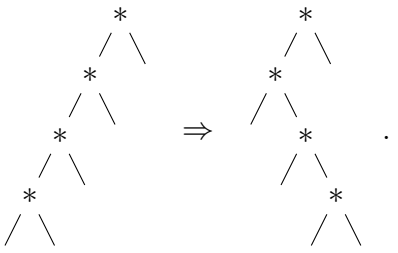
pour tous les couples $\mathfrak{t}_1 \to \mathfrak{t}_2$ et tous les arbres binaires $p, s_1, \ldots, s_n$, où n est le nombre de feuilles de l'arbre $\mathfrak{t}_1$.

Comme pour les congruences, la relation $\Rightarrow$ est le passage au contexte des règles de l'ensemble $\to$. Par exemple, si $\Rightarrow$ est la relation de réécriture engendrée par l'unique règle



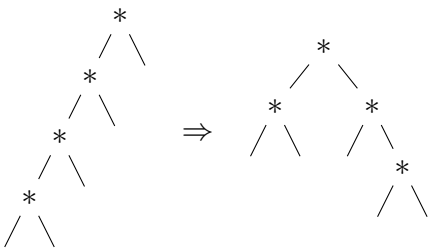
(1.20)

alors



. (1.21)

Un arbre n'a pas nécessairement une unique manière d'être réécrit. En effet, on a également dans ce cas la réécriture



. (1.22)

Deux arbres $\mathfrak{t}$ et $\mathfrak{t}'$ sont égaux dans l'espace quotient

$$\mathbf{Bin}\Big/_{\langle \mathfrak{t}_1 \equiv \mathfrak{t}_2 \rangle} \tag{1.23}$$

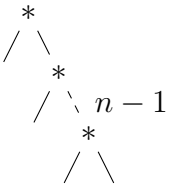
si et seulement s'il existe des arbres binaires $x_1, \ldots, x_k$ tels que

$$\mathfrak{t} \Leftrightarrow x_1 \Leftrightarrow \cdots \Leftrightarrow x_k \Leftrightarrow \mathfrak{t}', \tag{1.24}$$

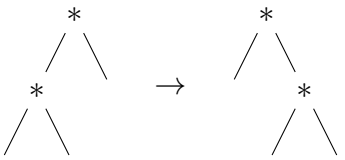
où $x \Leftrightarrow y$ signifie que $x \Rightarrow y$ ou $y \Rightarrow x$ et $\Rightarrow$ est la relation de réécriture engendrée par la règle $\mathfrak{t}_1 \to \mathfrak{t}_2$. Pour tester si deux arbres sont équivalents, il suffit donc de chercher des chaînes de réécritures du type (1.24). Dans le cas de **As**, on sait qu'ils sont égaux si et seulement s'ils ont le même nombre de feuilles.

On note que les arbres à droite des réécritures (1.21) et (1.22) ne peuvent pas être réécrits par la relation de réécriture engendrée par la règle (1.20). On dit qu'ils sont en ***forme normale*** pour cette relation.

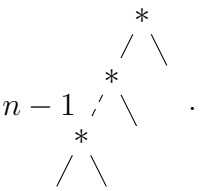
En reprenant le raisonnement (1.19) du point de vue de la réécriture, on obtient que tous les arbres à n feuilles se réécrivent après un certain nombre d'étapes en l'arbre

$d_n \quad := \quad$ 

pour la relation de réécriture engendrée par l'unique règle

 . (1.25)

Les arbres d_n sont appelés les ***peignes droits***. Symétriquement, on appelle ***peignes gauches*** les arbres

$g_n \quad := \quad$  .

On peut de même montrer que la règle de réécriture $d_2 \to g_2$ qui est l'inverse de la règle de réécriture (1.25) donne pour formes normales les peignes gauches.

Nous généraliserons, entre autres, dans la partie 4.2, ces raisonnements aux quotients de **Bin** par les relations $g_n \equiv d_n$.

Plan du manuscrit

Le chapitre 2 est principalement consacré à la recherche d'un contre-exemple à la conjecture *commutativement équivalent*. Notre stratégie est de trouver des codes non commutativement préfixes dans la partie 2.1, puis de voir dans la partie 2.2 si l'un d'entre eux est inclus dans un code maximal fini.

Nous avons trouvé des codes non commutativement préfixes en étudiant d'abord le cas général dans la sous-partie 2.1.1, puis en nous restreignant aux codes baïonnettes dans la sous-partie 2.1.2.

Pour être inclus dans un code maximal fini, un code doit vérifier certaines conditions. Nous étudions dans la sous-partie 2.2.1 une de ces conditions liée aux factorisations des groupes cycliques. Elle nous permet entre autres de calculer des bornes inférieures sur les tailles des codes maximaux finis susceptibles de contenir nos codes non commutativement préfixes. Nous introduisons dans la sous-partie 2.2.2 le concept de *code modulaire baïonnette complet*. Celui-ci nous permet d'améliorer certaines bornes inférieures obtenues dans la sous-partie précédente. Pour conclure ce chapitre, nous proposons dans la sous-partie 2.2.3 un code dont nous conjecturons qu'il est le plus petit code non inclus dans un code maximal fini.

Nous étudions la combinatoire des circuits dans le chapitre 3. Nous présentons et démontrons une construction combinatoire des circuits dans la partie 3.1. Nous déduisons de cette construction, une bijection entre les circuits et des chemins combinatoires que nous exposons dans la partie 3.2. Nous obtenons des résultats de combinatoire énumérative dans la partie 3.3 grâce à une étude de ces chemins. Plus précisément, la sous-partie 3.3.1 rassemble les formules de récurrence, la sous-partie 3.3.2 contient une équation fonctionnelle et la sous-partie 3.3.3 propose deux formules closes pour les circuits n'ayant qu'un seul type de composant.

Nous étudions dans le chapitre 4 la combinatoire énumérative de certains espaces quotients définis par des relations arborescentes. Nous étudions dans la partie 4.2 les quotients peignes. Plus précisément, dans la sous-partie 4.2.1, nous exhibons une structure de treillis sur l'ensemble des quotients peignes. Dans la sous-partie 4.2.2, nous étudions des ensembles de règles de réécritures associés à ces quotients ainsi que leurs séries de Hilbert. Nous concluons ce chapitre par la partie 4.3, où nous étudions les 10 quotients de **Bin** par les relations qui confondent deux arbres possédant exactement 4 feuilles. Pour

chacun de ces quotients, nous donnons entre autres sa série de Hilbert et une réalisation combinatoire.

Nous clôturons ce manuscrit en présentant des pistes de recherches dans la partie *perspectives*.

Chapitre 2

Une exploration informatique de la théorie des codes

Nous avons introduit dans le chapitre 1 des notions de base de la théorie des codes. Nous allons au début de cette partie réintroduire de manière formelle et plus générale ces notions. Nous recommandons, par ailleurs, le livre *Codes and Automata* [BPR10] pour une introduction poussée à la théorie des codes et comme référence pour trouver les démonstrations des propriétés que nous nous contentons d'évoquer. Commençons par quelques rappels.

Étant donné un alphabet fini $\mathcal{A}$, on appelle ***code*** sur cet alphabet un sous-ensemble $\mathcal{X} \subset \mathcal{A}^*$ tel que pour tout $n, m \geq 0$ et $x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathcal{X}$ la condition

$$x_1\, x_2 \cdots x_n \;=\; y_1\, y_2 \cdots y_m$$

implique que

$$n \;=\; m \text{ et } x_i \;=\; y_i \text{ pour tout } i \in [1, n].$$

Par exemple, l'ensemble $\{aabb, abaaa, b, ba\}$ n'est pas un code car

$$(b)(abaaa)(b)(b) \;=\; (ba)(ba)(aabb).$$

On considère qu'il est superflu de préciser l'alphabet associé à un code lorsque celui-ci se déduit du code.

Un code est dit ***maximal*** s'il n'est pas contenu dans un autre code. Un ensemble $\mathcal{X} \subset \mathcal{A}^*$ est dit ***préfixe*** si aucun élément de $\mathcal{X}$ n'est un préfixe d'un autre élément de $\mathcal{X}$. On peut facilement démontrer [1] qu'un ensemble préfixe qui ne contient pas le mot vide est un code. Plus généralement, on dit qu'un code $\mathcal{X}$ est ***commutativement préfixe*** s'il existe un code préfixe $\mathcal{P}$ tel que

$$\sum_{x \in \mathcal{X}} \alpha(x) \;=\; \sum_{p \in \mathcal{P}} \alpha(p),$$

1. comme nous l'avons fait en introduction pour le cas particulier où $\mathcal{A} = \{0, 1\}$.

où α est l'application qui associe à un mot son image commutative. En d'autres termes, cela signifie qu'un ensemble $\mathcal{X}$ est commutativement préfixe si on peut construire un code préfixe à partir de $\mathcal{X}$ en changeant l'ordre des lettres dans ses mots. On sait qu'un ensemble $\mathcal{X} \subset \mathcal{A}^*$ est commutativement préfixe si et seulement si tous les coefficients de la série

$$\frac{1 - \sum_{x \in \mathcal{X}} \alpha(x)}{1 - \sum_{a \in \mathcal{A}} \alpha(a)} \tag{2.1}$$

sont positifs.

La conjecture ***commutativement équivalent*** est l'un des principaux problèmes ouverts de la théorie des codes. Elle a été proposée par Marcel-Paul Schützenberger [Sch65]. Elle s'énonce ainsi : *tous les codes maximaux finis sont commutativement préfixes.*

Étant donné un ensemble $\mathcal{X}$, on note $\underline{\mathcal{X}}$ la somme formelle

$$\sum_{x \in \mathcal{X}} x$$

des éléments de $\mathcal{X}$. Marcel-Paul Schützenberger a conjecturé que pour tout code maximal fini $\mathcal{X}$, il existe $\mathcal{P}, \mathcal{S} \subset \mathcal{A}^*$ tel que

$$\underline{\mathcal{X}} - 1 = \underline{\mathcal{P}} \, (\underline{\mathcal{A}} - 1) \, \underline{\mathcal{S}},$$

où 1 désigne le mot vide. Cette conjecture est nommée la ***conjecture de la factorisation***, elle implique la conjecture *commutativement équivalent*. En 1985, Christophe Reutenauer [Reu85] a démontré la version faible de la *conjecture de la factorisation* suivante : pour tout code maximal fini $\mathcal{X}$, il existe $\mathcal{P}, \mathcal{S} \in \mathbb{Z}\langle\mathcal{A}\rangle$ tel que

$$\underline{\mathcal{X}} - 1 = \mathcal{P} \, (\underline{\mathcal{A}} - 1) \, \mathcal{S}.$$

Un ensemble $\mathcal{X}$ est dit ***baïonnette*** si $\mathcal{X} \subset a^*ba^*$. On appelle un mot de la forme a^*ba^* une baïonnette car c'est la forme de l'arbre binaire (voir le procédé (1.6)) qui l'engendre. La restriction de la conjecture *commutativement équivalent* sur ces ensembles est appelée la ***conjecture du triangle***. Elle stipule qu'un code baïonnette non commutativement préfixe ne peut pas être inclus dans un code maximal fini. Il existe un critère particulièrement simple pour savoir si un code baïonnette est commutativement préfixe. En effet, on sait qu'un code baïonnette $\mathcal{X}$ est commutativement préfixe si et seulement si

$$\left|\mathcal{X} \cap \mathcal{A}^{\leq n}\right| \leq n, \text{ pour tout } n \geq 0. \tag{2.2}$$

En 1985, Shor [Sho85] a trouvé le code baïonnette

$$\left\{b, ba, ba^7, ba^{13}, ba^{14}, a^3b, a^3ba^2, a^3ba^4, a^3ba^6, a^8b, a^8ba^2, a^8ba^4, a^8ba^6, a^{11}b, a^{11}ba, a^{11}ba^2\right\} \quad (2.3)$$

composé de 16 éléments et inclus dans $\mathcal{A}^{\leq 15}$. Il s'agit donc d'un code non commutativement préfixe. Jusqu'à notre article [Cor19], c'était le seul code connu de ce type. On ne sait toujours pas s'il est inclus dans un code maximal fini. Si tel est le cas alors la conjecture *commutativement équivalent* et son cas particulier la *conjecture du triangle* seraient fausses.

On connaît néanmoins des contraintes sur l'existence d'un code maximal fini contenant le code de Shor. En effet, on sait que pour tout code maximal fini $\mathcal{X}$ et pour toute lettre $\mu \in \mathcal{A}$, il existe un unique entier k tel que $\mu^k \in \mathcal{X}$. On appelle cet entier l'***ordre*** de la lettre μ. On peut montrer que si le code de Shor est inclus dans un code maximal fini alors l'ordre de la lettre a est un multiple de 330. Ceci signifie qu'un code maximal fini contenant le code (2.3) contiendrait au moins un mot de longueur au moins 330.

Nous sommes motivés par la recherche d'un contre-exemple à la conjecture *commutativement équivalent*. La stratégie que nous adoptons est de d'abord trouver à l'aide d'ordinateurs des codes non commutativement préfixes puis de voir s'ils sont inclus ou non dans des codes maximaux finis. Commençons par la première étape : la recherche de code non commutativement préfixe sur l'alphabet binaire $\mathcal{A} := \{a, b\}$.

2.1 Codes non commutativement préfixes

2.1.1 Le cas général

Nous avons vu dans les rappels que la littérature fournit le critère (2.1) pour déterminer si un code est commutativement préfixe ou non. Ce critère n'est pas un algorithme car il faut vérifier une infinité de coefficients. Cependant, en le développant, nous obtenons une autre caractérisation.

Théorème 2.1.1. *Soit $\mathcal{X} \subset \{a, b\}^*$, notons*

$$\alpha(\mathcal{X}) = \sum_{i=1}^{m} c_i \boldsymbol{a}^{x_i} \boldsymbol{b}^{y_i}$$

son image commutative, avec $x_1 + y_1 \leq \cdots \leq x_m + y_m$. L'ensemble $\mathcal{X}$ est commutativement préfixe si et seulement si pour tout j tel que $1 \leq j \leq m$,

on a

$$\sum_{i=1}^{j} \binom{x_j + y_j - x_i - y_i}{x_j - x_i} c_i \leq \binom{x_j + y_j}{x_j}.$$

Démonstration. D'après (2.1), l'ensemble $\mathcal{X}$ est commutativement préfixe si et seulement si la série

$$\frac{1 - \sum\limits_{x \in \mathcal{X}} \alpha(x)}{1 - \alpha(a+b)}$$

n'a que des coefficients positifs. Cette série est égale à la série

$$\sum_{n \geq 0} \sum_{k=0}^{n} \binom{n}{k} \left(1 - \sum_{i=1}^{m} c_i \mathbf{a}^{x_i} \mathbf{b}^{y_i}\right) \mathbf{a}^k \mathbf{b}^{n-k}.$$

L'ensemble $\mathcal{X}$ est donc commutativement préfixe si et seulement si pour tous les entiers n, k tels que $0 \leq k \leq n$, on a

$$s_{n,k}(\mathcal{X}) \leq \binom{n}{k}, \tag{2.4}$$

où

$$s_{n,k}(\mathcal{X}) := \sum_{i=1}^{m} \binom{n - x_i - y_i}{k - x_i} c_i.$$

Montrons par récurrence sur n qu'il est suffisant de vérifier la relation (2.4) pour les couples (n, k) appartenant à l'ensemble

$$\{(x_1 + y_1, x_1), \ldots, (x_m + y_m, x_m)\}. \tag{2.5}$$

Pour $n = 0$ et k quelconque, la relation (2.4) est trivialement vérifiée.

Supposons maintenant la relation (2.4) vraie pour n fixé et pour k quelconque. Si $n + 1 \neq x_1 + y_1, \ldots, x_m + y_m$ alors d'après la relation de Pascal :

$$s_{n+1,k}(\mathcal{X}) = s_{n,k-1}(\mathcal{X}) + s_{n,k}(\mathcal{X}).$$

En appliquant l'hypothèse de récurrence, nous obtenons :

$$s_{n+1,k}(\mathcal{X}) \leq \binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}.$$

La relation (2.4) est donc vraie en remplaçant n par $n + 1$.

S'il existe un entier r tel que $n + 1 = x_r + y_r$ alors il existe j et p tels que

$$x_1 + y_1 \leq \cdots < n + 1 = x_j + y_j = \cdots = x_{j+p} + y_{j+p} < \cdots \leq x_m + y_m.$$

Donc

$$s_{n+1,k}(\mathcal{X}) = \sum_{i=1}^{j-1} \binom{n+1-x_i-y_i}{k-1-x_i} c_i + \sum_{i=j}^{j+p} \binom{n+1-x_i-y_i}{k-x_i} c_i.$$

Si de plus $k \neq x_j, \ldots, x_{j+p}$ alors

$$s_{n+1,k}(\mathcal{X}) = \sum_{i=1}^{j-1} \binom{n+1-x_i-y_i}{k-1-x_i} c_i.$$

En appliquant la relation de Pascal puis l'hypothèse de récurrence, nous obtenons

$$s_{n+1,k}(\mathcal{X}) = s_{n,k-1}(\mathcal{X}) + s_{n,k}(\mathcal{X}) \leq \binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}.$$

Dans tous les cas, nous obtenons le fait que si la relation (2.4) est vraie sur les couples de l'ensemble (2.5) et pour n alors elle est vraie pour $n+1$. On conclut la démonstration grâce au principe de récurrence. □

Remarquons que dans le cas où l'ensemble $\mathcal{X}$ est un ensemble baïonnette, le théorème 2.1.1 confirme le critère (2.2).

Ce critère fournit un algorithme quadratique pour tester si un ensemble (*a fortiori* un code) fini est commutativement préfixe ou non. Avec l'aide de Michaël Rao, nous avons ainsi pu vérifier en programmant en C/C++ que tous les codes dont les mots sont de longueurs inférieures ou égales à 6 sont commutativement préfixes. Ce programme étant trop lent pour trouver des codes non commutativement préfixes, nous allons nous restreindre au cas particulier des codes baïonnettes.

2.1.2 Les codes baïonnettes

Étant donné un code baïonnette $\mathcal{X}$, nous appelons ***dual*** du code $\mathcal{X}$ le code baïonnette

$$\delta(\mathcal{X}) := \left\{ a^i b a^j \mid a^j b a^i \in \mathcal{X} \right\}.$$

Nous disons que $\mathcal{X}$ est ***auto-dual*** si $\mathcal{X} = \delta(\mathcal{X})$. Bien évidement, un code baïonnette est commutativement préfixe ou est inclus dans un code maximal fini si et seulement si son dual l'est. Nous considérons donc dans ce travail qu'un code baïonnette et son dual sont identiques. Même s'ils ne peuvent pas être égaux dans le cas qui nous intéresse, d'après la contraposée de la proposition suivante.

Proposition 2.1.2. *Si $\mathcal{X}$ est un code baïonnette auto-dual alors $\mathcal{X}$ est commutativement préfixe.*

Démonstration. Soit $\mathcal{X}$ un code baïonnette auto-dual et n un entier. Soit $\mathcal{E}_i^n$ l'ensemble

$$\left(a^i b a^* \cup a^* b a^i\right) \cap \mathcal{A}^{\leq n},$$

pour $i \geq 0$. Nous remarquons que

$$\mathcal{X} \cap \mathcal{A}^{\leq n} = \bigsqcup_{0 \leq i \leq \lceil \frac{n}{2} \rceil} (\mathcal{X} \cap \mathcal{E}_i^n).$$

Ce qui implique que

$$\left|\mathcal{X} \cap \mathcal{A}^{\leq n}\right| = \sum_{0 \leq i \leq \lceil \frac{n}{2} \rceil} |\mathcal{X} \cap \mathcal{E}_i^n|.$$

Il est suffisant de montrer que

$$|\mathcal{X} \cap \mathcal{E}_i^n| \leq 2$$

pour $i \geq 0$. Démontrons-le par l'absurde. Si

$$|\mathcal{X} \cap \mathcal{E}_i^n| > 2$$

alors il existe j_1 et j_2 tels que $i \leq j_1 < j_2 < n - i$ et

$$a^i b a^{j_1}, a^i b a^{j_2}, a^{j_1} b a^i, a^{j_2} b a^i \in \mathcal{X}.$$

Or

$$\left(a^i b a^{j_1}\right)\left(a^{j_2} b a^i\right) = \left(a^i b a^{j_2}\right)\left(a^{j_1} b a^i\right).$$

Ceci contredit le fait que $\mathcal{X}$ est un code. Nous savons donc que

$$\left|\mathcal{X} \cap \mathcal{A}^{\leq n}\right| \leq n$$

et grâce à (2.2), nous pouvons conclure que $\mathcal{X}$ est commutativement préfixe. □

Les codes baïonnettes auto-duaux ne peuvent donc pas dépasser la borne (2.2) mais ils peuvent cependant l'atteindre. Par exemple, pour un entier n, le code baïonnette auto-dual

$$\left\{a^i b a^{n-1-i} : 0 \leq i < n\right\}$$

atteint la borne (2.2).

Exploration informatique

Notre recherche informatique des codes baïonnettes non commutativement préfixes va nécessiter un algorithme capable de déterminer si un ensemble baïonnette donné est un code. Nous pourrions utiliser un des algorithmes déjà connus [SP53] comme nous l'avons fait dans le cas général. Cependant, nous ne profiterions pas de la restriction aux codes baïonnettes. Nous avons donc programmé un algorithme reposant sur la théorie des graphes afin de tester si un ensemble baïonnette est un code ou non.

Étant donnés un entier n et un ensemble $\mathcal{X} \subset a^*ba^* \cap \mathcal{A}^{\leq n}$, nous définissons le graphe orienté $\mathcal{G}_{\text{abs}}(X)$ [2] comme le graphe dont l'ensemble des nœuds est $[0, n-1]$ et dont les arêtes sont

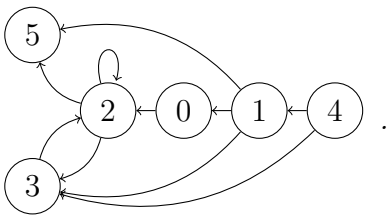
$$\boxed{|i-k|} \longrightarrow \boxed{|j-\ell|},$$

pour tout $a^iba^j, a^kba^\ell \in \mathcal{X}$ tels que $a^iba^j \neq a^kba^\ell$.

Exemple 2.1.3. *Soit $\mathcal{X}$ l'ensemble*

$$\left\{a^4ba^3, a^2ba^5, aba^5, b, ba^2\right\}, \tag{2.6}$$

le graphe $\mathcal{G}_{abs}(\mathcal{X})$ est



.

La proposition suivante caractérise les codes.

Proposition 2.1.4. *L'ensemble baïonnette $\mathcal{X}$ fini est un code si et seulement si le graphe $\mathcal{G}_{abs}(\mathcal{X})$ ne contient pas de chemin non vide de 0 à 0.*

Démonstration. Soit n un entier tel que $\mathcal{X} \subset a^*ba^* \cap \mathcal{A}^{\leq n}$. Il y a une arête de i à j dans le graphe $\mathcal{G}_{\text{abs}}(\mathcal{X})$ si et seulement s'il existe $u, v \in \mathcal{X}$ tels que

$$a^iu = va^j \text{ ou } a^iua^j = v.$$

Par concaténation, il existe un chemin de i à j dans le graphe si et seulement s'il existe $u, v \in \mathcal{X}^*$ tels que

$$a^iu = va^j \text{ ou } a^iua^j = v. \tag{2.7}$$

2. Christophe Reutenauer, nous a signalé l'article [PS81] dans lequel le graphe $\mathcal{G}_{\text{abs}}$ est étudié. Cependant, la démonstration de la proposition 2.1.4 semble inédite.

Supposons qu'il existe un chemin non vide de 0 à 0 passant par $k \neq 0$. D'après (2.7), il existe $u_1, u_2, v_1, v_2 \in \mathcal{X}^*$ tel que

$$u_1 = v_1 a^k \text{ et } u_2 = a^k v_2.$$

Donc

$$u_1 v_2 = v_1 u_2.$$

Ce qui contredit le fait que $\mathcal{X}$ est un code.

Réciproquement, si $\mathcal{X}$ n'est pas code alors il existe

$$a^{i_1} b a^{j_1}, \ldots, a^{i_n} b a^{j_n}, a^{k_1} b a^{\ell_1}, \ldots, a^{k_n} b a^{\ell_n} \in \mathcal{X}$$

tels que $j_1 \neq \ell_1$ et

$$\left(a^{i_1} b a^{j_1}\right)\left(a^{i_2} b a^{j_2}\right) \cdots \left(a^{i_n} b a^{j_n}\right) = \left(a^{k_1} b a^{\ell_1}\right)\left(a^{k_2} b a^{\ell_2}\right) \cdots \left(a^{k_n} b a^{\ell_n}\right).$$

Le graphe $\mathcal{G}_{\text{abs}}(\mathcal{X})$ contient donc le chemin

$$\boxed{0 = |i_1 - k_1|} \to \boxed{|j_1 - \ell_1| = |i_2 - k_2|} \to \cdots \to \boxed{|j_n - \ell_n| = 0}.$$

Ce qui conclut la démonstration. □

On déduit de cette proposition que l'ensemble (2.6) de l'exemple 2.1.3 est un code.

Notre recherche des codes baïonnettes non commutativement préfixes fonctionne par l'algorithme avec *retour sur trace* (*backtracking* en anglais) suivant : étant donné un entier n, nous énumérons en suivant l'ordre d'inclusion les ensembles baïonnettes

$$\mathcal{X} \subset a^* b a^* \cap \mathcal{A}^{\leq n}.$$

L'algorithme retourne sur ses traces lorsque le graphe $\mathcal{G}_{\text{abs}}(\mathcal{X})$ contient un chemin de 0 à 0 et exhibe $\mathcal{X}$ lorsque $|\mathcal{X}| > n$.

Nous avons programmé cet algorithme en C et nous avons obtenu les résultats suivants : il n'y a pas de code non commutativement préfixe pour $n \leq 11$, $n = 13$ et $n = 14$. Il y a 4 codes pour $n = 12$. Nous les affichons dans le tableau ci-dessous en représentant le mot baïonnette $a^i b a^j$ par $x_i x_j$, où x_i est le i^{e} chiffre de la base 17 $(0, \ldots, 9, \text{A}, \ldots, \text{G})$. La base 17 nous sera nécessaire afin d'écrire les codes pour n plus grand que 12.

Nom	Code baïonnette non commutativement préfixe												
$\mathcal{X}_1$	00	02	08	0A	18	1A	40	42	50	53	56	90	92
$\mathcal{X}_2$	01	03	09	0B	18	1A	40	42	50	53	56	90	92
$\mathcal{X}_3$	02	08	0A	10	18	1A	42	50	53	56	60	92	A0
$\mathcal{X}_4$	02	08	0A	18	1A	20	42	53	56	60	70	92	B0

FIGURE 2.1 – Codes non commutativement préfixes pour $n = 12$.

Ces 4 codes sont les plus petits codes (en termes de cardinalité et de la longueur de leurs plus longs mots) non commutativement préfixes que l'on connaisse.

Dans son article de 1985 [Sho85], Shor a demandé quelle est la valeur maximale du ratio

$$\frac{|\mathcal{X}|}{n}, \tag{2.8}$$

où $\mathcal{X}$ est un code baïonnette dont le plus long mot est de longueur inférieure ou égale à n. Intuitivement, plus un code $\mathcal{X}$ a un ratio (2.8) élevé, plus il est efficace pour certaines communications non homogènes (voir la sous-partie 1.1.2 du chapitre *introduction*). Cette question est toujours ouverte.

Cependant, on en connaît une réponse partielle sous la forme d'un encadrement. Hansel [Han82] a découvert une borne supérieure au ratio (2.8) et Shor avait déduit la borne inférieure $\frac{16}{15}$ de son exemple (2.2). Nous déduisons de nos codes ($\mathcal{X}_1$–$\mathcal{X}_4$) que ce ratio est nécessairement supérieur ou égal à $\frac{13}{12}$. Ce qui améliore la borne inférieure trouvée par Shor.

Nous avons également réalisé une recherche exhaustive des codes baïonnettes non commutativement préfixes pour $n = 15$. Après plusieurs semaines de calculs parallélisés sur 4 cœurs, nous avons trouvé 38 codes dont celui de Shor. Nous les exposons dans l'annexe A.

N'ayant pas assez de puissance de calcul pour faire une recherche exhaustive au delà de $n = 15$, nous avons effectué des recherches partielles pour $n = 16$ et $n = 17$. Nous exposons les 25 codes que nous avons trouvés pour $n = 16$ dans l'annexe B. Pour $n = 17$, nous avons constaté qu'il n'existe pas de code non commutativement préfixe contenant le mot b. Mais nous avons tout de même décelé les 3 codes ci-dessous. Il existe donc des exemples même quand n est premier.

01 02 03 0C 0D 3C 3D 51 52 5B 80 82 84 86 D1 D2 D3 G0 (2.9a)

01 02 03 0C 0D 3C 3D 51 52 5B 81 83 85 87 D1 D2 D3 G0 (2.9b)

01 02 03 0C 0D 3C 3D 51 52 5B 82 84 86 88 D1 D2 D3 G0 (2.9c)

Nous avons donc découvert au total 69 codes baïonnettes non commutativement préfixes en plus du code de Shor. Nous rappelons que si un de ces codes est inclus dans un code maximal fini alors la conjecture *commutativement équivalent* est fausse. Nous passons donc à la deuxième phase de notre stratégie qui est la recherche d'un code maximal fini contenant un de ces codes.

2.2 Inclusion dans un code maximal fini

On ne connaît pas de méthode permettant de savoir si un code quelconque est inclus ou non dans un code maximal fini. On ne sait même pas si ce problème est décidable. Malgré cela, il existe des conditions nécessaires à vérifier pour être inclus dans un code maximal fini. Ces conditions, comme celles portant sur les factorisations des groupes cycliques, nous renseignent sur l'existence d'une telle inclusion.

2.2.1 Factorisations des groupes cycliques

Nous supposons dans cette partie que les 70 codes baïonnettes non commutativement préfixes que nous avons trouvés dans la partie précédente sont inclus dans des codes maximaux finis. Nous allons utiliser la théorie des factorisations des groupes cycliques pour calculer des bornes inférieures sur les ordres des lettres a dans ces codes maximaux finis.

Étant donné $n \geq 1$, un couple (L, R) tel que $L, R \subset [0, n-1]$ est une ***factorisation*** du groupe $\mathbb{Z}/n\mathbb{Z}$ si et seulement si pour tout $k \in [0, n-1]$ il existe un unique couple $(\ell, r) \in L \times R$ tel que

$$k = \ell + r \bmod n.$$

Exemple 2.2.1. *Le couple*

$$(\{1, 3, 5\}, \{1, 2, 7, 8\})$$

est une factorisation du groupe $\mathbb{Z}/12\mathbb{Z}$.

Restivo, Salemi et Sportelli ont montré [RSS89] le lien suivant entre les factorisations et la théorie des codes.

Théorème 2.2.2. *Si* $\mathcal{X}$ *est un code maximal fini tel que* $b, a^n \in \mathcal{X}$ *alors le couple*

$$\left(\left\{k \bmod n : a^k b^+ \in \mathcal{X}\right\}, \left\{k \bmod n : b^+ a^k \in \mathcal{X}\right\}\right)$$

est une factorisation du groupe $\mathbb{Z}/n\mathbb{Z}$. *Une telle factorisation est dite* ***associée*** *au code* $\mathcal{X}$.

Sands a démontré le théorème suivant [San00].

Théorème 2.2.3. *Si (L, R) est une factorisation du groupe $\mathbb{Z}/n\mathbb{Z}$ et p un nombre premier avec $|L|$ alors (pL, R) est une factorisation du groupe $\mathbb{Z}/n\mathbb{Z}$.*

Nous appelons une ***factorisation de Sands*** une factorisation (L, R) telle que $0, 1 \in R$ et $0, p, q \in L$, où p et q sont premiers entre eux. On ne connaît toujours pas de factorisation associée au code de Shor. Autrement dit, on ne sait pas s'il existe un entier n et une factorisation (de Sands) (L, R) du groupe $\mathbb{Z}/n\mathbb{Z}$ telle que

$$(L \supseteq \{0, 3, 8, 11\}, R \supseteq \{0, 1, 7, 13, 14\}) .$$

S'il n'existe pas de factorisation de ce type alors le code de Shor ne peut pas être un contre-exemple à la conjecture *commutativement équivalent*.

Nous allons étudier les factorisations associées aux codes baïonnettes non commutativemment préfixes trouvés dans la partie précédente.

Factorisations connues

Nous avons découvert des factorisations associées aux codes $\mathcal{X}_1$, $\mathcal{Y}_6$, $\mathcal{Y}_8$, $\mathcal{Y}_{10}$, $\mathcal{Y}_{12}$, $\mathcal{Y}_{14}$ et $\mathcal{Y}_{16}$[3]. Le lecteur peut par exemple vérifier que pour $n \geq 2$, le couple

$$\left(\{0, 4, 5, 9\}, \bigsqcup_{i \in [0, n-1]} \{8i, 8i + 2\} \right) \tag{2.10}$$

est une factorisation de $\mathbb{Z}/8n\mathbb{Z}$ associée au code $\mathcal{X}_1$. Plus généralement, l'entier n tel qu'il existe une factorisation du groupe $\mathbb{Z}/n\mathbb{Z}$ associée au code $\mathcal{X}_1$ doit être un multiple de 4. En effet, les couples

$$(\{0, 4, 5, 9\}, 2\{0, 2, 8, 10\}) \text{ et } (2\{0, 4, 5, 9\}, \{0, 2, 8, 10\})$$

ne sont pas des factorisations car

$$0 + 2 \times 2 = 4 + 0 \text{ et } 2 \times 4 + 0 = 0 + 8.$$

Donc d'après la contraposée du théorème de Sands 2.2.3, nous savons que $|L|$ et $|R|$ sont des multiples de 2. On en conclut que n est un multiple de 4. Nous n'avons pas trouvé de factorisation associée au code $\mathcal{X}_1$ telle que n soit un multiple de 4 sans être un multiple de 8.

Le lecteur peut pareillement vérifier que pour $n \geq 2$, le couple

$$(\{0, 8, \cdots, 8(n-1)\}, \{0, 1, 2, 3, 4, 7, 13, 14\}) \tag{2.11}$$

3. Nous rappelons que ces codes sont définis dans le tableau 2.1.2 et dans l'annexe A.

est une factorisation de $\mathbb{Z}/8n\mathbb{Z}$ associée aux codes $\mathcal{Y}_6$, $\mathcal{Y}_8$, $\mathcal{Y}_{10}$, $\mathcal{Y}_{12}$, $\mathcal{Y}_{14}$ et $\mathcal{Y}_{16}$. Par un argument similaire, nous pouvons montrer qu'un entier n tel qu'il existe une factorisation de $\mathbb{Z}/n\mathbb{Z}$ associée à ces codes est nécessairement un multiple de 4.

Ces exemples sont les premiers exemples de codes non commutativement préfixes qui possèdent des factorisations associées. Les factorisations associées aux autres codes seront nécessairement de type Sands ou équivalentes à ce type.

Factorisations de Sands

Nous n'avons pas trouvé de factorisations de Sands associées à nos codes mais nous avons obtenu des contraintes sur leurs existences.

Supposons le code $\mathcal{Y}_2$ inclus dans un code maximal fini et soit (L, R) une factorisation associée à ce code. Nous savons donc que

$$L \supseteq \{0, 3, 8\} \text{ et } R \supseteq \{0, 1, 7, 13, 14\}.$$

Remarquons que $(L, 8R)$, $(L, 3R)$ et $(L, 5R)$ ne sont pas des factorisations car

$$8 + 8 \times 0 = 0 + 8 \times 1,\ 8 + 3 \times 0 = 0 + 3 \times 1 \text{ et } 8 + 5 \times 0 = 3 + 5 \times 1.$$

Donc d'après la contraposée du théorème 2.2.3, nous savons que $|R|$ est un multiple de $2 \times 3 \times 5$. L'ordre de la lettre a est donc de la forme $30 \times k$, avec $k \geq 3$. En suivant le même raisonnement nous avons calculé la table suivante.

Codes	Ordre de la lettre a
$\mathcal{Y}_2$, $\mathcal{Y}_4$	$2 \times 3 \times 5 \times k = 30k$, où $k \geq 3$
$\mathcal{Y}_5$, $\mathcal{Y}_7$, $\mathcal{Y}_9$, $\mathcal{Y}_{11}$, $\mathcal{Y}_{13}$, $\mathcal{Y}_{15}$	$2 \times 3 \times 11 \times k = 66k$, où $k \geq 3$
$\mathcal{Y}_1$, $\mathcal{Y}_3$	$2 \times 3 \times 5 \times 11 \times k = 330k$, où $k \geq 4$
$\mathcal{Z}_1$, $\mathcal{Z}_3$, $\mathcal{Z}_5$, $\mathcal{Z}_7$	$2 \times 3 \times 5 \times 13 \times k = 390k$, où $k \geq 4$
$\mathcal{Z}_2$, $\mathcal{Z}_4$, $\mathcal{Z}_6$, $\mathcal{Z}_8$	$2 \times 3 \times 5 \times 13 \times k = 390k$, où $k \geq 3$
$\mathcal{Z}_9$, $\mathcal{Z}_{10}$	$2 \times 5 \times 13 \times k = 130k$, où $k \geq 3$

FIGURE 2.2 – Ordre la lettre a dans les codes maximaux finis.

Remarque 2.2.4. *Un couple (L, R) est une factorisation d'un groupe G si et seulement si $(L, R - r)$ est une factorisation du même groupe, où $r \in R$. Donc si*

$$(L \supseteq \{0, 5, 13\}, R \supseteq \{0, 2, 11, 12\})$$

est une factorisation associée aux codes $\mathcal{Z}_9$ *et* $\mathcal{Z}_{10}$ *alors* $(L, R-11)$ *est une factorisation de Sands.*

Nous savons donc que si un des codes cités dans la table (2.2.1) est un contre-exemple à la conjecture *commutativement équivalent* alors pour produire un de ces contre-exemples il nous faudra chercher un code maximal fini possédant des mots de longueurs supérieures ou égales à 90. Or ceci est hors de portée de nos programmes informatiques.

Nonobstant ces résultats, le lien (et la condition qui en résulte) entre la théorie des codes et la théorie des factorisations des groupes cycliques, tel qu'énoncé dans le théorème 2.2.2, souffre selon nous d'au moins trois défauts lorsqu'il est appliqué aux codes baïonnettes. Premièrement, nous avons trouvé des codes non commutativement préfixes qui vérifient cette condition sans pour autant trouver de codes maximaux finis les contenant. Cette condition est donc trop faible. Deuxièmement, le lien ne concerne que les codes contenant le mot b ; le lien exclut de facto 43 des 70 codes. [4] Et pour finir, le lien ne prend en compte que les mots appartenant à l'ensemble $ba^* \cup a^*b$.

Nous introduisons dans la partie suivante un nouvel objet que nous appelons *code baïonnette modulaire* afin de pallier ces défauts.

2.2.2 Codes baïonnettes modulaires complets

Dans [PS77], Perrin et Schützenberger ont démontré le théorème suivant.

Théorème 2.2.5. *Soit* $\mathcal{X}$ *un code maximal fini. Soit* $\mu \in \mathcal{A}$ *une lettre et* n *son ordre. Pour tout* $\omega \in \mathcal{A}^*$*, l'ensemble*

$$C_\mu(\omega) := \left\{(i \bmod n,\ j \bmod n) : \mu^i \omega \mu^j \in \mathcal{X}^*\right\}$$

contient exactement n *éléments.*

Nous appelons code baïonnette n-***modulaire*** un code $\mathcal{X} \subset a^{<n}ba^{<n}$ tel que $\{a^n\} \cup \mathcal{X}$ soit un code et nous disons qu'il est ***complet*** si $|\mathcal{X}| = n$. D'après le cas particulier du théorème 2.2.5 où $\omega = b$ et $\mu = a$, on sait que pour être inclus dans un code maximal fini, un code baïonnette doit être inclus dans un code baïonnette n-modulaire complet, pour un certain n.

Exemple 2.2.6. *Nous appelons code de permutation un sous-ensemble* $\mathcal{X}$ *de* $a^{<n}ba^{<n}$ *tel que la matrice carrée binaire* $\mathcal{M}$ *de taille* n *définie par*

$$\mathcal{M}_{i,j} = 1 \text{ si et seulement si } a^iba^j \in \mathcal{X}$$

4. Pour être plus précis, le lien a été étendu par Lam dans [Lam96] aux codes ne contenant pas nécessairement le mot b. Cependant, ce nouveau lien engendre une condition qui est trivialement vérifiée par tous nos codes. Elle est donc trop faible.

est une matrice de permutation. On laisse au lecteur le soin de montrer qu'un code de permutation est un code baïonnette n-modulaire complet.

Nous allons chercher à l'ordinateur si un de nos codes est inclus dans un code baïonnette n-modulaire complet.

Exploration informatique

Nous avons légèrement modifié l'algorithme de la première partie pour tester si un ensemble donné est un code baïonnette n-modulaire.

Étant donné un ensemble $\mathcal{X} \subset a^{<n}ba^{<n}$, nous appelons $\mathcal{G}_{\mathrm{mod}}(\mathcal{X})$ le graphe orienté dont les nœuds sont les entiers de $[0, n-1]$ et les arêtes sont

$$\boxed{i-k \bmod n} \longrightarrow \boxed{\ell - j \bmod n},$$

pour tout a^iba^j, $a^kba^\ell \in X$, où $a^iba^j \neq a^kba^\ell$. Par un raisonnement similaire à celui de la proposition 2.1.4, nous obtenons qu'un ensemble $\mathcal{X}$ est un code n-modulaire si et seulement si le graphe $\mathcal{G}_{\mathrm{mod}}(\mathcal{X})$ ne contient pas de chemin non vide de 0 à 0.

Dans la partie précédente, nous n'avions pas écarté la possibilité que les codes $\mathcal{X}_1$, $\mathcal{Y}_6$, $\mathcal{Y}_8$, $\mathcal{Y}_{10}$, $\mathcal{Y}_{12}$, $\mathcal{Y}_{14}$ et $\mathcal{Y}_{16}$ puissent être inclus des codes maximaux finis dont l'ordre de la lettre a est de la forme $4k$, avec $k \geq 4$. Sans oublier les 43 autres codes ne contenant pas le mot b pour lesquels nous n'avions rien pu dire.

Cependant, grâce à une recherche exhaustive à l'ordinateur, nous obtenons qu'aucun des 70 codes n'est inclus dans un code n-modulaire complet, pour $n \leq 32$. On en déduit que si un des 70 codes est un contre-exemple à la conjecture *commutativement équivalent* alors il sera inclus dans un code maximal fini qui aura nécessairement au moins un de ses mots de longueur strictement supérieure à 32. En particulier, si le code $\mathcal{X}_1$ est inclus dans un code maximal fini alors l'ordre de sa lettre a est de la forme $4k$, où $k \geq 10$ (il n'y a pas de factorisation de $\mathbb{Z}/n\mathbb{Z}$ associée à ce code pour $32 < n < 40$). De même, si un des codes $\mathcal{Y}_6$, $\mathcal{Y}_8$, $\mathcal{Y}_{10}$, $\mathcal{Y}_{12}$, $\mathcal{Y}_{14}$ et $\mathcal{Y}_{16}$ est inclus dans un code maximal fini alors l'ordre de sa lettre a est de la forme $4k$ avec $k \geq 9$ (il n'y a pas de factorisation de $\mathbb{Z}/n\mathbb{Z}$ associée à ces codes pour $32 < n < 36$).

Cette exploration informatique nous révèle en outre le théorème suivant.

Théorème 2.2.7. *Si $\mathcal{X}$ est un des codes ($\mathcal{X}_1$– $\mathcal{X}_4$), alors*

$$\mathcal{X} \cup \{a^{16}\} \tag{2.12}$$

est un code non commutativement préfixe et non inclus dans un code maximal fini.

Démonstration. Si $\mathcal{X}$ est un des codes ($\mathcal{X}_1$- $\mathcal{X}_4$) alors on vérifie aisément que $\mathcal{X} \cup \{a^{16}\}$ est un code. De plus, nous avons vérifié qu'aucun de ces codes n'était inclus dans un code 16-modulaire complet. Nous concluons la démonstration grâce au théorème 2.2.5. □

Les quatre codes (2.12) sont les premiers exemples de codes non commutativement préfixes et non inclus dans un code maximal fini. Avec le même procédé et en utilisant les 66 autres codes, nous arrivons à produire de nouveaux exemples de ce type. Cependant, les codes (2.12) sont les plus petits (au sens de la longueur de leurs plus longs mots et de la cardinalité) que nous trouvons.

Transformations

Afin de mieux comprendre les codes modulaires complets, nous allons énoncer et étudier certaines de leurs transformations.

Lemme 2.2.8. *Si $\mathcal{X}$ est un code n-modulaire alors pour tout $r \in [0, n-1]$, l'ensemble*

$$s_r(\mathcal{X}) := \left\{ a^i b a^j : a^i b a^p, a^q b a^j \in \mathcal{X} \text{ et } p + q = r \mod n \right\}$$

est un code n-modulaire.

Démonstration. Étant donné un entier $r \in [0, n-1]$, si $s_r(\mathcal{X})$ n'est pas un code n-modulaire alors il existe

$$a^{i_1} b a^{j_1}, \ldots, a^{i_m} b a^{j_m}, a^{k_1} b a^{\ell_1}, \ldots, a^{k_m} b a^{\ell_m} \in s_r(\mathcal{X})$$

avec $j_1 \neq \ell_1$, tel que

$$\begin{cases} i_1 & = & k_1 & \\ j_1 + i_2 & = & \ell_1 + k_2 & \mod n \\ & \vdots & & \\ j_{m-1} + i_m & = & \ell_{m-1} + k_m & \mod n \\ j_m & = & \ell_m & \end{cases}.$$

Par définition de $s_r(\mathcal{X})$, il existe

$$a^{i_1} b a^{p_1}, a^{q_1} b a^{j_1}, \ldots, a^{i_m} b a^{p_m}, a^{q_m} b a^{j_m} \in \mathcal{X}$$

et

$$a^{k_1} b a^{p'_1}, a^{q'_1} b a^{\ell_1}, \ldots, a^{k_m} b a^{p'_m}, a^{q'_m} b a^{\ell_m} \in \mathcal{X}$$

tels que $p_t + q_t = p'_t + q'_t = r \bmod n$, pour tout $t \in [1, m]$. On en déduit que

$$\begin{cases} i_1 & = & k_1 & \\ p_1 + q_1 & = & p'_1 + q'_1 & \bmod n \\ j_1 + i_2 & = & \ell_1 + k_2 & \bmod n \\ & \vdots & & \\ j_{m-1} + i_m & = & \ell_{m-1} + k_m & \bmod n \\ p_m + q_m & = & p'_m + q'_m & \bmod n \\ j_m & = & \ell_m & \end{cases}.$$

L'ensemble $\mathcal{X}$ n'est donc pas un code n-modulaire car il engendre une double décomposition. Nous concluons la preuve par contraposition. □

Nous avons à l'origine utilisé ce lemme afin de démontrer le théorème 2.2.9. Cependant, nous nous sommes rendu compte après l'avoir publié dans l'article [Cor19], qu'il était équivalent à la proposition 2.1 issue de l'article [DFR85] de De Felice et de Restivo. Proposition qui est elle-même équivalente à la proposition 1 de Perrin et Schützenberger de l'article [PS81], que Christophe Reutenauer nous a signalé... Nous donnons notre démonstration.

Théorème 2.2.9. *Si $\mathcal{X}$ est un code n-modulaire alors $|\mathcal{X}| \leq n$.*

Démonstration. Supposons que $\mathcal{X}$ est un code n-modulaire tel que $|\mathcal{X}| > n$. D'après le lemme 2.2.8, nous savons que pour tout entier $r \in [0, n-1]$, l'ensemble $s_r(\mathcal{X})$ est un code donc

$$\sum_{r \in [0,n-1]} |s_r(\mathcal{X})| = |\mathcal{X}|^2.$$

Il existe donc un entier $r_1 \in [0, n-1]$ tel que

$$|s_{r_1}(\mathcal{X})| \geq \left\lceil \frac{|\mathcal{X}|^2}{n} \right\rceil \geq n + 1.$$

En itérant, nous obtenons le fait qu'il existe des entiers $r_2, \ldots, r_{n^2} \in [0, n-1]$ tels que

$$\left| s_{r_{n^2}} \left(\cdots s_{r_2} \left(s_{r_1}(X) \right) \cdots \right) \right| > n^2.$$

Ce qui contredit le fait que $\mathcal{X}$ appartienne à $a^{<n}ba^{<n}$. □

Grâce à ce résultat nous pouvons maintenant exhiber cinq transformations sur les codes modulaires qui préservent la complétude.

Théorème 2.2.10. *Si $\mathcal{X}$ est un code n-modulaire (respectivement complet) alors*

1. *pour tous les entiers α et β, l'ensemble*

$$\tau_{\alpha,\beta}(\mathcal{X}) := \left\{a^{i+\alpha \bmod n} b a^{j+\beta \bmod n} : a^i b a^j \in \mathcal{X}\right\}$$

est un code n-modulaire (respectivement complet) ;

2. *pour tout q premier avec n, l'ensemble*

$$\rho_q(\mathcal{X}) := \left\{a^{qi \bmod n} b a^{qj \bmod n} : a^i b a^j \in \mathcal{X}\right\}$$

est un code n-modulaire (respectivement complet) ;

3. *l'ensemble*

$$\iota(\mathcal{X}) := \left\{a^{n-1-i} b a^j : a^i b a^j \in \mathcal{X}\right\}$$

est un code n-modulaire (respectivement complet) ;

4. *le code dual $\delta(\mathcal{X})$ est un code n-modulaire (respectivement complet) ;*

5. *pour tout entier $r \in [0, n-1]$, l'ensemble $s_r(\mathcal{X})$ est un code n-modulaire (respectivement complet).*

Démonstration.

1. Pour tout α, $\beta \in [0, n-1]$, le graphe $\mathcal{G}_{\mathrm{mod}}(\tau_{\alpha,\beta}(\mathcal{X}))$ est identique au graphe $\mathcal{G}_{\mathrm{mod}}(\mathcal{X})$.

2. Pour tout entier q premier avec n, la fonction qui associe à $i \in [0, n-1]$ l'entier $qi \bmod n$ est un isomorphisme du graphe $\mathcal{G}_{\mathrm{mod}}(\mathcal{X})$ au graphe $\mathcal{G}_{\mathrm{mod}}(\rho_q(\mathcal{X}))$.

3. Si $\iota(\mathcal{X})$ n'est pas un code n-modulaire alors le graphe $\mathcal{G}_{\mathrm{mod}}(\iota(\mathcal{X}))$ contient des chemins du type

$$\boxed{0} \longrightarrow \boxed{i_1} \longrightarrow \boxed{i_2} \longrightarrow \cdots \boxed{i_m} \longrightarrow \boxed{0}$$

et

$$\boxed{0} \longrightarrow \boxed{-i_1 \bmod n} \longrightarrow \boxed{-i_2 \bmod n} \longrightarrow \cdots \boxed{-i_m \bmod n} \longrightarrow \boxed{0}$$

pour $i_1, \ldots, i_m \in [0, n-1]$. Le graphe $\mathcal{G}_{\mathrm{mod}}(\mathcal{X})$ contient donc le chemin

$$\boxed{0} \longrightarrow \boxed{i_1} \longrightarrow \boxed{-i_2 \bmod n} \longrightarrow \boxed{i_3} \longrightarrow \boxed{-i_4 \bmod n} \longrightarrow \cdots \longrightarrow \boxed{0}.$$

Ce qui contredit le fait que $\mathcal{X}$ est un code n-modulaire. Nous concluons par contraposition.

4. Le graphe $\mathcal{G}_{\mathrm{mod}}(\delta(\mathcal{X}))$ est le graphe $\mathcal{G}_{\mathrm{mod}}(\mathcal{X})$ dont on a inversé les arêtes.

5. Si $\mathcal{X}$ est un code n-modulaire alors d'après le lemme 2.2.8, l'ensemble $s_r(X)$ est un code n-modulaire. Il faut démontrer que si
$$|\mathcal{X}| = n$$
alors pour tout $r \in [0, n-1]$ on a $|s_r(\mathcal{X})| = n$. Supposons que
$$|s_r(\mathcal{X})| \neq n$$
pour $r \in [0, n-1]$, il existe donc un entier $r' \in [0, n-1]$ tel que
$$|s_{r'}(\mathcal{X})| > n.$$
Ce qui contredit le théorème 2.2.9.

□

Nous nous demandons si la conjecture suivante (version forte du théorème 2.2.10.2) est vraie.

Conjecture 2.2.11. *Si $\mathcal{X}$ est un code n-modulaire complet alors pour tout q premier avec n, l'ensemble*
$$\varphi_q(\mathcal{X}) := \left\{a^{qi \bmod n} b a^j : a^i b a^j \in \mathcal{X}\right\}$$
est un code n-modulaire complet.

Si cette conjecture est vraie alors en utilisant sa contraposée, on pourrait améliorer les bornes inférieures exposées dans le tableau (2.2.1) mais également proposer des bornes inférieures du même ordre de grandeur pour les cas où le mot b n'appartient pas au code. Remarquons que le théorème 2.2.10 implique le cas $q = n-1$ de la conjecture 2.2.11, en effet $\varphi_q(\mathcal{X}) = \tau_{1,0}(\iota(\mathcal{X}))$.

2.2.3 Plus petits codes non inclus dans un code maximal fini

Le plus petit code (au sens de la longueur du plus long mot et du cardinal) connu à ce jour qui ne soit pas inclus dans un code maximal fini est le code
$$\{aaaaa, ab, b, baa\}$$
d'Antonio Restivo [Res77]. Il l'a trouvé grâce au lien entre les factorisations des groupes cycliques et la théorie de codes.

Nous avons, par une exploration informatique, vérifié que tous les codes dont le plus long mot est de longueur inférieure ou égale à 3 sont inclus dans des codes maximaux finis. Par ailleurs, on ne sait toujours pas s'il existe des codes composés de seulement trois éléments qui ne sont pas inclus dans des codes maximaux finis. Nous proposons pour conclure ce chapitre la conjecture suivante.

Conjecture 2.2.12. *Le code*

$$\{aa, abab, baaa, b\} \tag{2.13}$$

est un des plus petits codes, au sens de la longueur du plus long mot et du cardinal, non inclus dans un code maximal fini.

Nous laissons le lecteur vérifier que ce code résiste aux outils que nous avons utilisés ou développés au cours de cette partie. En effet, on trouve facilement des factorisations de groupes cycliques qui lui sont associées mais également des codes baïonnettes 2-modulaires complets qui le contiennent.

Chapitre 3

Combinatoire des prographes

Les prographes sont les éléments de certaines structures algébriques libres appelées pros libres. Un pro[1] est défini en théorie des catégories[2] comme une catégorie monoïdale stricte [Laf09, Lei04]. On attribut rétrospectivement cette notion à Mac Lane car il a introduit dans son article de 1965 [ML65] la version symétrique des pros, appelée prop.

À notre connaissance, les pros apparaissent pour la première fois en homotopie sous le nom de *catégorie d'opérateurs sans permutations* [BV68]. Depuis, l'étude et l'usage des pros se sont étendus à d'autres domaines comme la topologie algébrique [BV06], la combinatoire algébrique [Bor17, BG16a] et énumérative [Cor18], la théorie de la réécriture [Laf09], la théorie des automates [LLMN19], etc.

Comme rappelé dans [BG16b], nous pouvons reformuler la définition des pros sans utiliser le langage de la théorie des catégories.

Définition 3.0.1. *Un* ***pro*** *est un quadruplet* $(\mathcal{P}, \circ, \star, \mathbb{1})$ *tel que*

- *l'ensemble* $\mathcal{P}$ *soit un ensemble bigradué de la forme*

$$\bigsqcup_{e,s\geq 0} \mathcal{P}(e,s).$$

Nous notons $\downarrow(x)$ *et* $\uparrow(x)$ *les entiers tels que* $x \in \mathcal{P}(\downarrow(x), \uparrow(x))$ *et nous posons* $\Delta(x) := \uparrow(x) - \downarrow(x)$*. Les entiers* $\downarrow(x)$ *et* $\uparrow(x)$ *sont respectivement appelés le nombre d'entrées et le nombre de sorties de* x *;*

1. On trouve le plus souvent dans la littérature, *pro* en lettres capitales. Cependant, pro est l'abréviation de *produit de catégories*. Cela ne nécessite selon nous et les auteurs de [LLMN19] pas de traitement typographique particulier.

2. Nous recommandons aux lecteurs intéressés par la théorie des catégories le livre de référence [ML13]. Pour une simple introduction, nous préconisons le livre [Awo10].

- *l'opération* $\circ$ *est une opération binaire associative de la forme*

$$\circ : \mathcal{P}(e, s_1) \times \mathcal{P}(s_1, s_2) \to \mathcal{P}(e, s_2),$$

pour tout $e, s_1, s_2 \geq 0$ *;*

- *l'opération* $\star$ *est une opération binaire associative de la forme*

$$\star : \mathcal{P}(e_1, s_1) \times \mathcal{P}(e_2, s_2) \to \mathcal{P}(e_1 + e_2, s_1 + s_2),$$

pour tout $e_1, e_2, s_1, s_2 \geq 0$ *;*

- *pour tout* $x \in \mathcal{P}(e, s)$*,*

$$x \circ \mathbb{1}^s = x = \mathbb{1}^e \circ x, \tag{3.1}$$

où pour tout $n \geq 0$*,* $\mathbb{1}^n$ *désigne* $\underbrace{\mathbb{1} \star \ldots \star \mathbb{1}}_{n}$ *;*

- *pour tout* $x_1, y_1, x_2, y_2 \in \mathcal{P}$ *tel que* $\uparrow(x_1) = \downarrow(y_1)$ *et* $\uparrow(x_2) = \downarrow(y_2)$*,*

$$(x_1 \circ y_1) \star (x_2 \circ y_2) = (x_1 \star x_2) \circ (y_1 \star y_2). \tag{3.2}$$

Nous utiliserons cette définition des pros dans le reste de ce manuscrit.

Exemple 3.0.2. *Le quadruplet*

$$\left(\bigsqcup_{p,q \geq 0} \mathcal{M}_{p,q}(\mathbb{N}), \times, \oplus, [1] \right),$$

où $\mathcal{M}_{p,q}(\mathbb{N})$ *désigne les matrices d'entiers de taille* pq*,* $\times$ *le produit matriciel et* $\oplus$ *la somme directe, est un pro. Ce pro est étudié dans le chapitre* 6 *du livre [Gir18a].*

Dans certains pros, tous les éléments ont le même nombre d'entrées que de sorties.

Exemple 3.0.3. *Le quadruplet*

$$\left(\mathbb{N}^* = \bigsqcup_{n \geq 0} \mathcal{N}(n, n), +, \cdot, 0 \right)$$

est un pro, où $\mathcal{N}(n, n)$ *est l'ensemble des suites d'entiers naturels de longueur* n*,* $+$ *est la somme terme à terme des suites de même longueur et* $\cdot$ *est la concaténation.*

Informellement, un *pro libre* est un pro où les relations vérifiées par ses éléments sont nécessairement induites par les axiomes des pros. Ceci est formalisé par le fait qu'un pro libre est la solution d'un problème universel. Pour introduire ce formalisme, nous devons d'abord introduire la notion de *morphisme de pro.*

Définition 3.0.4. *Soit* $(\mathcal{P}, \circ, \star, \mathbb{1})$ *et* $\left(\overline{\mathcal{P}}, \overline{\circ}, \overline{\star}, \overline{\mathbb{1}}\right)$ *deux pros. On appelle* ***morphisme de pro*** *toute application* f *définie sur*

$$\mathcal{P}(e, s) \to \overline{\mathcal{P}}(e, s), \qquad e, s \geq 0,$$

telle que

$$f(\mathbb{1}) = \overline{\mathbb{1}}$$

et telle que pour tout $x, y \in \mathcal{P}$*,*

$$f(x \star y) = f(x) \,\overline{\star}\, f(y)$$

et

$$f(x \circ y) = f(x) \,\overline{\circ}\, f(y)$$

lorsque $\uparrow(x) = \downarrow(y)$*.*

Nous pouvons maintenant définir les pros libres.

Définition 3.0.5. *Soit* $\mathbb{G}$ *un ensemble bigradué de générateurs. Nous appelons* ***pro libre*** *engendré par* $\mathbb{G}$*, noté* $\mathrm{FP}(\mathbb{G})$ *ou* $\mathrm{FP}(\mathbb{G}, \circ, \star, \mathbb{1})$*, le pro qui est solution du problème universel suivant : il existe une application*

$$\pi : \mathbb{G} \to \mathrm{FP}(\mathbb{G})$$

telle que pour toute application f *(respectant la bigraduation) de* $\mathbb{G}$ *dans* $\mathcal{P}$*, il existe un unique morphisme de pro*

$$\overline{f} : \mathrm{FP}(\mathbb{G}) \to \mathcal{P}$$

tel que le diagramme

$$\begin{array}{ccc} \mathrm{FP}(\mathbb{G}) & \xrightarrow{\overline{f}} & \mathcal{P} \\ {\scriptstyle\pi}\uparrow & \nearrow_{f} & \\ \mathbb{G} & & \end{array} \tag{3.3}$$

commute.

Si $\mathbb{G}$ est un ensemble fini de générateurs tel que $\uparrow(x) \neq 0$ et $\downarrow(x) \neq 0$ pour tout $x \in \mathbb{G}$ alors $\mathrm{FP}(\mathbb{G})$ est également surnommé le *pro des circuits* engendré par $\mathbb{G}$ [LLMN19], ses éléments sont appelés des ***prographes*** ou des circuits.

Remarque 3.0.6. *Deux objets qui sont solutions du problème universel (3.3) sont nécessairement égaux à isomorphisme près [Bou70], donc si* $\mathrm{FP}(\mathbb{G})$ *existe alors il est unique à isomorphisme près.*

Comme esquissé en introduction, nous nous intéressons ici aux prographes en tant qu'objets combinatoires. Dans la partie suivante, nous présentons une construction combinatoire des prographes. Cette construction utilise l'opération suivante. Étant donné un pro $(\mathcal{P}, \circ, \star, \mathbb{1})$, pour tout $x, y \in \mathcal{P}$ et $i \in \mathbb{N}$ tel que $i + \downarrow(y) - 1 \leq \uparrow(x)$ nous introduisons l'opération

$$x \circ_i y := x \circ \left(\mathbb{1}^{i-1} \star y \star \mathbb{1}^{\uparrow(x)-i+1-\downarrow(y)}\right). \tag{3.4}$$

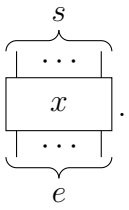
Un concept similaire est utilisé dans [Laf09].

3.1 Une construction des prographes

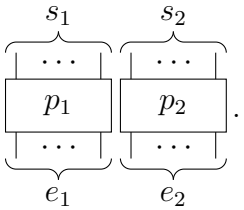
Nous présentons et démontrons dans cette partie une construction combinatoire des prographes. Commençons par en donner l'intuition.

3.1.1 Intuition sur la construction

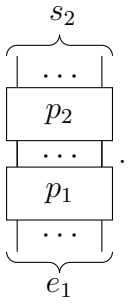
Étant donné un pro $(\mathcal{P}, \circ, \star, \mathbb{1})$, un élément $x \in \mathcal{P}(e, s)$ est souvent [Bor17, BG16b, Laf09, LLMN19, Cor18] représenté par une boîte



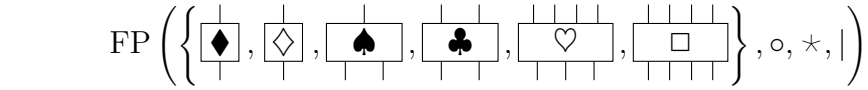
L'élément $\mathbb{1}$ est simplement représenté par un fil |. Étant donnés deux éléments $p_1 \in \mathcal{P}(e_1, s_1)$ et $p_2 \in \mathcal{P}(e_2, s_2)$, nous représentons l'élément $p_1 \star p_2$ par la composition parallèle



Finalement, si $p_1 \circ p_2$ est bien défini alors on le représente par la composition en série

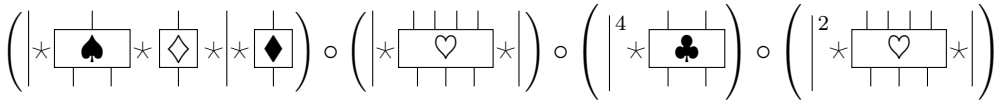


Exemple 3.1.1. *Le pro libre*

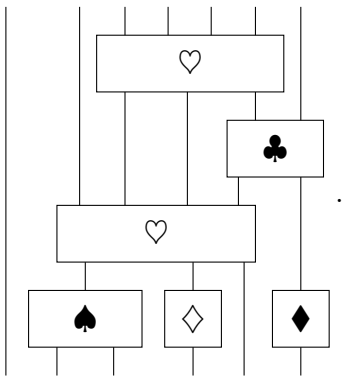


(3.5)

contient le prographe

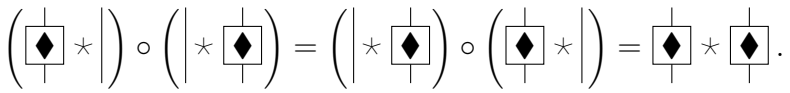


qu'on représente par



(3.6)

La description des prographes avec les opérations $\star$ et $\circ$ est en générale ambiguë. Par exemple, on a dans le pro libre (3.5) les relations



Nous cherchons une grammaire non-ambiguë pour engendrer les prographes. Premièrement, nous remarquons que le prographe (3.6) peut être construit par l'opération $\circ_i$. En effet,

$$(3.6) = \Big(\Big(\Big(\Big(\Big(\mathbb{1}^6 \circ_2 \spadesuit\Big) \circ_3 \diamondsuit\Big) \circ_2 \heartsuit\Big) \circ_6 \blacklozenge\Big) \circ_5 \clubsuit\Big) \circ_3 \heartsuit.$$

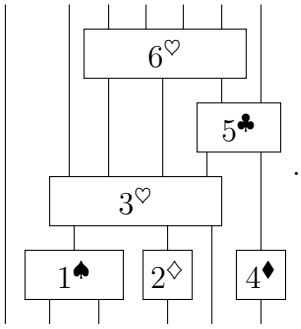
Nous montrerons dans la prochaine partie qu'il en est ainsi pour tous les prographes. Cependant, cette construction n'est pas unique en général. En effet, le prographe (3.6) est tout autant égal à

$$\Big(\Big(\Big(\Big(\big(\mathbb{1}^6 \circ_6 \blacklozenge\big) \circ_4 \diamondsuit\Big) \circ_2 \spadesuit\Big) \circ_2 \heartsuit\Big) \circ_5 \clubsuit\Big) \circ_3 \heartsuit.$$

L'ordre des générateurs dans ce type de décomposition peut être vu comme un parcours de prographe. Dans [Bor17], Borie introduit un parcours de prographes pour décrire les éléments de l'ensemble

$$\mathrm{FP}\left(\left\{\square, \square\right\}\right)(1,1).$$

Informellement, il s'agit d'un parcours en profondeur-gauche qui peut visiter un générateur à la condition que tous les générateurs connectés à ses entrées aient déjà été visités[3]. Son parcours se généralise sans difficulté à tous les prographes. Par exemple, ce parcours visite les générateurs du prographe (3.6) dans cet ordre



.

Informellement, nous avons opté pour la construction suivante : on place les générateurs dans l'ordre donné par le parcours étendu de Borie et les indices i des opérations $\circ_i$ sont uniquement déterminés par le nombre de fils à la gauche leurs générateurs plus un. Par exemple, le prographe (3.6) ne sera construit que par l'expression

$$\Big(\cdots\Big(\big|^6 \circ_2 \spadesuit\big) \circ_3 \diamondsuit\Big) \circ_2 \heartsuit\Big) \circ_6 \blacklozenge\Big) \circ_5 \clubsuit\Big) \circ_3 \heartsuit.$$

Dans la prochaine partie, nous encoderons ces expressions en des objets que nous appelons *prosuites*. Dans le but de formaliser notre construction combinatoire des prographes et de montrer sa concordance avec la définition algébrique.

3. Certains esthètes appellent ce parcours une *boriefication*.

3.1.2 Présentation de la construction

Présentons formellement notre construction des prographes. Soit $\mathbb{G}$ un ensemble bigradué de générateurs

$$\bigsqcup_{e,s\geq 1} \mathcal{G}(e,s).$$

Étant donné un générateur $x \in \mathbb{G}$, nous notons $\downarrow(x)$ et $\uparrow(x)$ les entiers tels que $x \in \mathcal{G}(\downarrow(x), \uparrow(x))$ et nous posons $\Delta(x) := \uparrow(x) - \downarrow(x)$.

Définition 3.1.2. *Nous notons* $\mathrm{PS}_{e,n,s}(\mathbb{G})$ *l'ensemble des couples*

$$\Big(e, (i_k, x_k)_{k\in[1,n]}\Big) \in \mathbb{N}_{\geq 1} \times (\mathbb{N}_{\geq 1} \times \mathbb{G})^n$$

tel que

$$s = e + \sum_{\ell=1}^{n} \Delta(x_\ell)$$

et tel que pour tout $k \in [1,n]$,

$$i_{k-1} < i_k + \downarrow(x_k) \leq e + \sum_{\ell=1}^{k-1} \Delta(x_\ell) + 1,$$

où $i_0 = 0$.

Nous notons $\mathrm{PS}(\mathbb{G})$ l'ensemble

$$\bigsqcup_{e,s\,\in\,\mathbb{N}_{\geq 1}} \bigsqcup_{n\,\in\,\mathbb{N}} \mathrm{PS}_{e,n,s}(\mathbb{G})$$

et nous baptisons ses éléments des ***prosuites***. Étant donnée une prosuite p, nous notons $\downarrow(p), |p|$ et $\uparrow(p)$ les entiers tels que $p \in \mathrm{PS}_{\downarrow(p),|p|,\uparrow(p)}(\mathbb{G})$.

Intuitivement, la prosuite

$$\Big(e, (i_k, x_k)_{k\in[1,n]}\Big)$$

va encoder le prographe

$$(\cdots((\mathbb{1}^e \circ_{i_1} x_1) \circ_{i_2} x_2)\cdots) \circ_{i_n} x_n.$$

Exemple 3.1.3. *Le prographe (3.6) sera dans notre construction des prographes la prosuite*

$$(6, (2,\spadesuit)(3,\diamondsuit)(2,\heartsuit)(6,\blacklozenge)(5,\clubsuit)(3,\heartsuit)). \tag{3.7}$$

Nous allons maintenant munir l'ensemble des prosuites $\mathrm{PS}(\mathbb{G})$ de deux opérations, notées $\star$ et $\bullet$. Ces opérations vont encoder les compositions parallèle et série des prosuites.

Définition 3.1.4. *Nous définissons l'opération $\star$ par*

$$\begin{array}{ccccc} \mathrm{PS}_{e,\,n,\,s}(\mathbb{G}) & \times & \mathrm{PS}_{e',\,m,\,s'}(\mathbb{G}) & \longrightarrow & \mathrm{PS}_{e+e',\,n+m,\,s+s'}(\mathbb{G}) \\ (e,u) & & \Big(e',(j_k,y_k)_{k\in[1,m]}\Big) & \mapsto & \Big(e+e',u\cdot(j_k+s,y_k)_{k\in[1,m]}\Big) \end{array}$$

pour tout $e,n,s,e',m,s'\in\mathbb{N}$, où le point $\cdot$ est l'opération de concaténation.

Exemple 3.1.5. *Si p_1 est la prosuite*

$$(3,(1,\diamondsuit)(2,\spadesuit)(1,\clubsuit))$$

et p_2 la prosuite

$$(2,(1,\blacklozenge)(1,\diamondsuit))$$

alors $p_1\star p_2$ est la prosuite

$$(5,(1,\diamondsuit)(2,\spadesuit)(1,\clubsuit)(3,\blacklozenge)(3,\diamondsuit))\,.$$

Afin de définir l'opération $\bullet$, nous introduisons d'abord l'opération d'insertion $\bullet^1$. Cette opération est partiellement définie sur

$$\mathrm{PS}_{e,\,n,\,s}(\mathbb{G})\times(\mathbb{N}_{\geq 1}\times\mathbb{G})\to\mathrm{PS}_{e,\,n+1,\,s+\Delta(x)}(\mathbb{G})$$

pour tout $e,n,s\in\mathbb{N}$. Si p est la prosuite

$$\Big(e,(i_k,x_k)_{k\in[1,n]}\Big)\in\mathrm{PS}_{e,\,n,\,s}(\mathbb{G})$$

alors pour tout $(i,x)\in(\mathbb{N}_{\geq 1}\times\mathbb{G})$ tel que $i+{\downarrow}(x)<s$ la prosuite $p\bullet^1(i,x)$ est calculée par l'algorithme suivant :

Algorithme 1 Calculer $p\bullet^1(i,x)$

Entrées : $i>0$ **et** $i+{\downarrow}(x)\leq s+1$
  $k:=n$
  $(i_{n+1},x_{n+1}):=(i,x)$
  tant que $k\geq 1$ **et** $i_k\geq i+{\downarrow}(x)$ **faire**
    $(i_{k+1},x_{k+1})\leftarrow(i_k+\Delta(x),x_k)$
    $(i_k,x_k)\leftarrow(i,x)$
    $k\leftarrow k-1$
  Fin tant que
  Retourner $(e,(i_k,x_k)_{k\in[1,n+1]})$

En d'autres termes, $p \bullet^1 (i,x)$ est la prosuite

$$(e,(i_1,x_1)\dots(i_t,x_t)(i,x)(i_{t+1}+\Delta(x),x_{t+1})\dots(i_n+\Delta(x),x_n)) \tag{3.8}$$

où

$$t = \max\left(\{k\in[1,n],\, i_k < i + \downarrow(x)\}\right),$$

par convention $\max(\emptyset) = 0$. Nous disons que (i,x) s'est inséré dans p à la position $t+1$.

Exemple 3.1.6. *L'opération* $\bullet^1$ *insère le couple* $(2,\spadesuit)$ *dans la prosuite*

$$(5,(2,\diamondsuit)(4,\spadesuit)(4,\blacklozenge))$$

à la position 2. *Autrement dit,*

$$(5,(2,\diamondsuit)(4,\spadesuit)(4,\blacklozenge)) \bullet^1 (2,\spadesuit)$$

est égal à la prosuite

$$(5,(2,\diamondsuit)(2,\spadesuit)(3,\spadesuit)(3,\blacklozenge)).$$

Nous pouvons maintenant définir l'opération $\bullet$.

Définition 3.1.7. *Soit* $p_1 \in \mathrm{PS}_{e,\,n,\,s}(\mathbb{G})$ *et*

$$p_2 := \left(s,(j_k,y_k)_{k\in[1,m]}\right) \in \mathrm{PS}_{s,\,m,\,s'}(\mathbb{G})$$

deux prosuites. Nous définissons l'opération $\bullet$ *sur*

$$\mathrm{PS}_{e,\,n,\,s}(\mathbb{G}) \times \mathrm{PS}_{s,\,m,\,s'}(\mathbb{G}) \to \mathrm{PS}_{e,\,n+m,\,s'}(\mathbb{G})$$

pour tous les entiers e,n,s,m,s' *par*

$$p_1 \bullet p_2 := \left(\cdots\left(\left(p_1 \bullet^1 (j_1,y_1)\right)\bullet^1 (j_2,y_2)\right)\cdots\right)\bullet^1 (j_m,y_m).$$

Exemple 3.1.8. *La prosuite (3.7) est égale à*

$$(6,(2,\spadesuit)(5,\blacklozenge)) \bullet (5,(3,\diamondsuit)(2,\heartsuit)(5,\clubsuit)(3,\heartsuit)).$$

Dans la prochaine partie, nous allons démontrer la cohérence entre notre construction des prographes et la définition algébrique de ces derniers.

3.1.3 Démonstration de la construction

Nous allons démontrer dans cette partie que l'ensemble des prosuites doté des opérations $\bullet$ et $\star$ est un pro libre. Pour cela, nous allons utiliser les deux propositions suivantes.

Proposition 3.1.9. *Pour tous les couples (i,x) et $(j,y) \in \mathbb{N}_{\geq 1} \times \mathbb{G}$ et pour toute prosuite p tels que*

$$\Big(p \bullet^1 (i,x)\Big) \bullet^1 (j,y) \text{ et } i \geq j + {\downarrow}(y),$$

on a

$$\Big(p \bullet^1 (i,x)\Big) \bullet^1 (j,y) = \Big(p \bullet^1 (j,y)\Big) \bullet^1 (i + \Delta(y), x).$$

Démonstration. Soit

$$\Big(e, (i_k, x_k)_{k\in[1,n]}\Big)$$

la prosuite p et t l'entier tel que (i,x) s'insère dans p à la position $t+1$. D'après la remarque (3.8), pour tout $k \in [t+1, n]$, on a $i_k \geq i + {\downarrow}(x)$ et par hypothèse $i \geq j + {\downarrow}(y)$ donc

$$j + {\downarrow}(y) \leq i \leq i_k - {\downarrow}(x) \leq i_k + \Delta(x).$$

Ceci implique que (j,y) s'insère dans $(p \bullet^1 (i,x))$ à la même position que dans p, cette position est plus petite que $t+1$. De plus, d'après l'hypothèse

$$i_k + \Delta(y) < i + \Delta(y) + {\downarrow}(x),$$

pour tout $k \in [t+1, n]$. On en déduit que le couple $(i + \Delta(y), x)$ est inséré dans $(p \bullet^1 (j,y))$ à la position $t+2$. D'où le résultat. □

Proposition 3.1.10. *Soit p_1 et p_2 deux prosuites. Pour tous les couples (i,x) tels que*

$$p_1 \bullet^1 (i,x)$$

soit bien défini, on a les deux propriétés élémentaires suivantes :

i) $(p_1 \bullet^1 (i,x)) \star p_2 = (p_1 \star p_2) \bullet^1 (i,x)$;

ii) $p_2 \star (p_1 \bullet^1 (i,x)) = (p_2 \star p_1) \bullet^1 (i + {\uparrow}(p_2), x)$.

Démonstration. i) Par hypothèse,

$$i + {\downarrow}(x) \leq 1 + {\uparrow}(p_1).$$

Donc (i,x) s'insère dans $p_1 \star p_2$ à la même position que dans p_1. Ce qui conclut la démonstration.

ii) Pour tout $k \in [1,n]$, on a $i_k \geq i + \downarrow(x)$ si et seulement si

$$i_k + \uparrow(p_2) \geq i + \uparrow(p_2) + \downarrow(x).$$

Donc

$$\uparrow(p_2) < i + \uparrow(p_2) + \downarrow(x).$$

On en déduit que si (i,x) est inséré dans p_1 à la position t alors $(i+\uparrow(p_2),x)$ est inséré dans $p_2 \star p_1$ à la position $t+|p_2|$. Ce qui conclut la démonstration. □

Soit ε la suite vide. Nous démontrons maintenant que les prosuites forment un pro.

Lemme 3.1.11. *Le quadruplet*

$$(\mathrm{PS}(\mathbb{G}), \bullet, \star, (1,1,\varepsilon))$$

est un pro.

Démonstration. Il suffit de montrer qu'il vérifie les axiomes des pros énoncés dans la définition 3.0.1. Nous démontrerons seulement l'associativité de l'opération $\bullet$ et la relation (3.2), nous laissons le lecteur prouver les autres relations élémentaires.

Premièrement, montrons que

$$(p_2 \bullet p_1) \bullet^1 (i,x) = p_2 \bullet \left(p_1 \bullet^1 (i,x)\right)$$

et par récurrence immédiate nous aurons l'associativité de l'opération $\bullet$. Soit p_1 la prosuite

$$\left(e, (i_k, x_k)_{k\in[1,n]}\right).$$

Si

$$p_2 \bullet \left(p_1 \bullet^1 (i,x)\right)$$

est bien défini alors il est égal à

$$\Big(\cdots\Big(p_2 \bullet^1 (i_1,x_1)\Big)\cdots \bullet^1 (i_t,x_t)\Big) \bullet^1 (i,x)\Big) \bullet^1 (i_{t+1}+\Delta(x), x_{t+1})\Big)\cdots \bullet^1 (i_n+\Delta(x), x_n), \quad (3.9)$$

où $t+1$ est la position d'insertion de (i,x) dans p_1. En itérant $n-t$ fois la proposition 3.1.9, la prosuite (3.9) devient

$$(p_2 \bullet p_1) \bullet^1 (i,x), \quad (3.10)$$

ce qui prouve l'associativité de l'opération $\bullet$.

Nous allons maintenant démontrer que ce quadruplet vérifie la relation (3.2). Soit $p_1, p_2, p'_1, p'_2 \in \mathrm{PS}(\mathbb{G})$ tels que

$$(p_1 \bullet p_2) \star (p'_1 \bullet p'_2) \tag{3.11}$$

soit bien défini. En itérant $|p'_2|$ fois la proposition 3.1.10.ii, la prosuite (3.11) devient

$$((p_1 \bullet p_2) \star p'_1) \bullet ((\uparrow(p_2), \uparrow(p_2), \varepsilon) \star p'_2). \tag{3.12}$$

En itérant $|p_2|$ fois la proposition 3.1.10.i, la prosuite (3.12) devient

$$((p_1 \star p'_1) \bullet p_2) \bullet ((\uparrow(p_2), \uparrow(p_2), \varepsilon) \star p'_2),$$

qui se simplifie en

$$(p_1 \star p_2) \bullet (p'_1 \star p'_2).$$

Ceci conclut la démonstration du lemme. □

Nous voulons également montrer que ce pro est libre. Pour cela nous allons utiliser la proposition suivante.

Proposition 3.1.12. *Soit un pro $(\mathcal{P}, \circ, \star, \mathbb{1})$, pour tous les entiers $i, j \in \mathbb{N}_{\geq 1}$ et pour tout $p, x, y \in \mathcal{P}$ tels que*

$$(p \circ_i x) \circ_j y$$

soit bien défini, on a les propriétés élémentaires :

i) si $i \geq j + \downarrow(y)$ alors $(p \circ_i x) \circ_j y = (p \circ_j y) \circ_{i+\Delta(x)} x$;

ii) $(p \circ_i x) \star \mathbb{1}^e = (p \star \mathbb{1}^e) \circ_i x$;

iii) $\mathbb{1}^e \star (p \circ_i x) = (\mathbb{1}^e \star p) \circ_{i+e} x$.

Démonstration. i) Si $i \geq j + \downarrow(y)$ alors

$$\left(\mathbb{1}^{\uparrow(p)} \circ_i x\right) \circ_j y$$

se factorise en

$$\left(\mathbb{1}^{j+\downarrow(y)} \star \left(\mathbb{1}^{i-j-\downarrow(y)} \star x \star \mathbb{1}^{\uparrow(p)-i-\downarrow(x)}\right)\right) \circ \left(\left(\mathbb{1}^{j} \star y\right) \star \mathbb{1}^{\uparrow(p)+\Delta(x)-j-\downarrow(y)}\right). \tag{3.13}$$

D'après (3.2), l'élément (3.13) vaut

$$\left(\mathbb{1}^{j+\downarrow(y)} \circ \left(\mathbb{1}^{j} \star y\right)\right) \star \left(\left(\mathbb{1}^{i-j-\downarrow(y)} \star x \star \mathbb{1}^{\uparrow(p)-i-\downarrow(x)}\right) \circ \mathbb{1}^{\uparrow(p)+\Delta(x)-j-\downarrow(y)}\right),$$

que l'on peut simplifier en

$$\left(\left(\mathbb{1}^{j} \star y\right) \circ \mathbb{1}^{j+\uparrow(y)}\right) \star \left(\mathbb{1}^{\uparrow(p)-j-\downarrow(y)} \circ \left(\mathbb{1}^{i-j-\downarrow(y)} \star x \star \mathbb{1}^{\uparrow(p)-i-\downarrow(x)}\right)\right). \tag{3.14}$$

Toujours d'après (3.2), l'élément (3.14) est égal à

$$\left(\left(\mathbb{1}^j \star y\right) \star \mathbb{1}^{\uparrow(p)-j-\downarrow(y)}\right) \circ \left(\mathbb{1}^{j+\uparrow(y)} \star \left(\mathbb{1}^{i-j-\downarrow(y)} \star x \star \mathbb{1}^{\uparrow(p)-i-\downarrow(x)}\right)\right),$$

que l'on peut simplifier en

$$\left(\mathbb{1}^{\uparrow(p)} \circ_j y\right) \circ_{i+\Delta(y)} x.$$

Nous concluons cette démonstration grâce à l'associativité de l'opération $\circ$.

ii) Par définition,

$$(p \circ_i x) \star \mathbb{1}^e$$

est égal à

$$\left(p \circ (\mathbb{1}^{i-1} \star x \star \mathbb{1}^{\uparrow(p)-i+1-\downarrow(x)})\right) \star \mathbb{1}^e. \tag{3.15}$$

D'après (3.2), l'élément (3.15) est égal à

$$(p \star \mathbb{1}^e) \circ \left(\mathbb{1}^{i-1} \star x \star \mathbb{1}^{\uparrow(p)+e-i+1-\downarrow(x)}\right),$$

que l'on peut simplifier en

$$(p \star \mathbb{1}^e) \circ_i x.$$

iii) De même, on sait que par définition

$$\mathbb{1}^e \star (p \circ_i x)$$

est égal à

$$\mathbb{1}^e \star \left(p \circ (\mathbb{1}^{i-1} \star x \star \mathbb{1}^{\uparrow(p)-i+1-\downarrow(x)})\right). \tag{3.16}$$

D'après (3.2), l'élément (3.16) est égal à

$$(\mathbb{1}^e \star p) \circ \left(\mathbb{1}^e \star (\mathbb{1}^{i-1} \star x \star \mathbb{1}^{\uparrow(p)-i+1-\downarrow(x)})\right),$$

que l'on peut simplifier en

$$(\mathbb{1}^e \star p) \circ_{i+e} x.$$

□

Nous allons maintenant montrer que le pro des prosuites est libre. Ce qui démontrera l'exactitude de notre construction.

Théorème 3.1.13. *Les pros* $(\mathrm{PS}(\mathbb{G}), \bullet, \star, (1, 1, \varepsilon))$ *et* $\mathrm{FP}(\mathbb{G})$ *sont isomorphes.*

Démonstration. Il nous suffit de montrer que le pro

$$(\mathrm{PS}(\mathbb{G}), \bullet, \star, (1, 1, \varepsilon))$$

est une solution du problème universel (3.3). Car d'après la remarque 3.0.6 les solutions à ce problème sont nécessairement isomorphes.

Soit un pro $(\mathcal{P}, \circ, \star, \mathbb{1})$ et f une application qui préserve la bigraduation de $\mathbb{G}$ dans $\mathcal{P}$. Nous définissons l'application π par

$$\begin{array}{rccc} \pi & : & \mathbb{G} & \to & \mathrm{PS}(\mathbb{G}) \\ & & g & \mapsto & (\downarrow(g), (1, g)) \end{array}.$$

Prouvons que l'application $\overline{f}$ définie par

$$\begin{array}{rccc} \overline{f} & : & \mathrm{PS}_{e,n,s}(\mathbb{G}) & \to & \mathrm{FP}(\mathbb{G}) \\ & & \Big(e, (i_k, x_k)_{k\in[1,n]}\Big) & \mapsto & (\dots(\mathbb{1}^e \circ_{i_1} f(x_1))\dots) \circ_{i_n} f(x_n) \end{array}, e, n, s \geq 0,$$

prolonge f en un morphisme de pro. Par construction, on a $\overline{f} \circ \pi = f$. Soit

$$p_1 := \Big(e_1, (i_k, x_k)_{k\in[1,n]}\Big) \in \mathrm{PS}_{e_1,n,s_1}(\mathbb{G})$$

et

$$p_2 := \Big(e_2, (j_k, y_k)_{k\in[1,m]}\Big) \in \mathrm{PS}_{e_2,m,s_2}(\mathbb{G})$$

deux prosuites. D'après (3.1) et (3.2), on a que

$$\overline{f}(p_1) \star \overline{f}(p_2)$$

est égal à

$$\Big(\overline{f}(p_1) \star \mathbb{1}^{s_1}\Big) \circ \Big(\mathbb{1}^{e_2} \star \overline{f}(p_2)\Big). \tag{3.17}$$

En itérant n fois la proposition 3.1.12.ii et m fois la proposition 3.1.12.iii, l'élément (3.17) devient

$$\begin{aligned} &\Big(\Big(\cdots\Big(\mathbb{1}^{e_1+e_2} \circ_{i_1} f(x_1)\Big)\dots\Big) \circ_{i_n} f(x_n)\Big) \\ &\qquad \circ \Big(\Big(\cdots\Big(\mathbb{1}^{s_1+e_2} \circ_{s_1+j_1} f(y_1)\Big)\cdots\Big) \circ_{s_1+j_m} f(y_m)\Big). \end{aligned} \tag{3.18}$$

Grâce à l'associativité de l'opération $\circ$ et à la définition de $\overline{f}$, l'expression (3.18) se simplifie en

$$\overline{f}\left(\Big(e_1 + e_2, (i_k, x_k)_{k\in[1,n]} \cdot (j_k + s_1, y_k)_{k\in[1,m]}\Big)\right).$$

Donc

$$\overline{f}(p_1) \star \overline{f}(p_2) = \overline{f}(p_1 \star p_2).$$

De plus, si

$$\overline{f}\left(p_1 \bullet^1 (i,x)\right)$$

est bien défini alors, d'après (3.8), il est égal à

$$(\dots(\mathbb{1}^e \circ_{i_1} f(x_1))\dots) \circ_{i_t} f(x_t)) \circ_i f(x)) \circ_{i_{t+1}+\Delta(x)} f(x_{t+1}))\cdots \circ_{i_n+\Delta(x)} f(x_n), \quad (3.19)$$

où $t+1$ est la position d'insertion de (i,x) dans p_1. En itérant $n-t$ fois la proposition 3.1.12.i, l'expression (3.19) se simplifie en

$$\overline{f}(p_1) \circ_i f(x).$$

Par récurrence immédiate, on obtient

$$\overline{f}(p_1 \bullet p_2) = \overline{f}(p_1) \circ \overline{f}(p_2).$$

Ceci démontre que $\overline{f}$ est un morphisme de pro.

Montrons par l'absurde que ce prolongement de f est unique. Supposons qu'il existe un morphisme de pro $\overline{g}$ qui prolonge l'application f alors

$$\begin{aligned}
\overline{f}(p_1) &= (\dots((\mathbb{1}^{e_1} \circ_{i_1} f(x_1)) \circ_{i_2} f(x_2))\dots) \circ_{i_n} f(x_n)\\
&= (\dots((\mathbb{1}^{e_1} \circ_{i_1} \overline{g}(\pi(x_1))) \circ_{i_2} \overline{g}(\pi(x_2)))\dots) \circ_{i_n} \overline{g}(\pi(x_n))\\
&= \overline{g}((e_1,(i_1,x_1)) \bullet \cdots \bullet (s_1 - \Delta(x_n), s_1, (i_n,x_n)))\\
&= \overline{g}(p_1).
\end{aligned}$$

Donc $\overline{f}$ est égal à $\overline{g}$. Ceci conclut la démonstration du théorème. □

Notons

$$\mathrm{PG}_{e,n,s}(\mathbb{G})$$

l'ensemble des prographes appartenant au pro libre $\mathrm{FP}(\mathbb{G})$ et composés de e entrées, s sorties et n générateurs. Le théorème 3.1.13 prouve que cet ensemble est bien défini et que son cardinal vaut

$$\left|\mathrm{PS}_{e,n,s}(\mathbb{G})\right|.$$

Un des principaux objectifs de ce chapitre est de dénombrer cet ensemble. Plus formellement, nous allons répondre au problème suivant.

Probleme. *Étant donné un ensemble de générateurs $\mathbb{G}$ et des entiers e, n, s, quel est le cardinal de l'ensemble $\mathrm{PG}_{e,n,s}(\mathbb{G})$?*

Dans la partie suivante, nous allons encoder les prosuites (et donc les prographes) par des chemins combinatoires afin d'en déduire des propriétés combinatoires.

3.2 Prographes et chemins combinatoires

Nous exhibons dans cette partie une bijection entre les prographes et des familles de chemins combinatoires colorés en trois dimensions.

3.2.1 Chemins combinatoires

Un ***chemin*** est un couple

$$(p_{\mathrm{i}}, w_1 \cdots w_n) \in \mathbb{Z}^3 \times \left(\mathbb{Z}^3 \times \mathbb{N}\right)^n,$$

où p_{i} est l'origine du chemin et $w_1 \cdots w_n$ est la suite des pas colorés en trois dimensions. Étant donné un pas $v \in \mathbb{Z}^3 \times \mathbb{N}$, nous notons respectivement $\gamma_1, \gamma_2, \gamma_3$ et γ_c l'abscisse, l'ordonnée, la cote et la couleur du pas v. Plus formellement, on a

$$v = (\gamma_1(v), \gamma_2(v), \gamma_3(v), \gamma_c(v)).$$

Nous utilisons les mêmes notations pour $v \in \mathbb{Z}^3$, c'est à dire que

$$v = (\gamma_1(v), \gamma_2(v), \gamma_3(v)).$$

Définition 3.2.1. *Pour $M \subset (\mathbb{Z}^3 \times \mathbb{N})^*$ et $p_{\mathrm{i}}, p_{\mathrm{f}} \in \mathbb{Z}^3$, notons*

$$\mathcal{L}(M, p_{\mathrm{i}}, p_{\mathrm{f}})$$

l'ensemble des chemins $(p_{\mathrm{i}}, w_1 \cdots w_n)$ tels que

i) $w_1 \cdots w_n \in M$;

ii) *pour* $j = 1, 2, 3$, $\gamma_j(p_{\mathrm{i}}) + \sum_{\ell=1}^{n} \gamma_j(w_\ell) = \gamma_j(p_{\mathrm{f}})$;

iii) *pour* $k \in [0, n]$, $1 \leq \gamma_2(p_{\mathrm{i}}) + \sum_{\ell=1}^{k} \gamma_2(w_\ell) \leq \gamma_3(p_{\mathrm{i}}) + \sum_{\ell=1}^{k} \gamma_3(w_\ell)$.

En d'autres termes, l'ensemble $\mathcal{L}(M, p_{\mathrm{i}}, p_{\mathrm{f}})$ est l'ensemble des chemins de p_{i} à p_{f} dont la suite des pas appartient à M et tel qu'en chaque point du chemin l'ordonnée est comprise entre 1 et la cote.

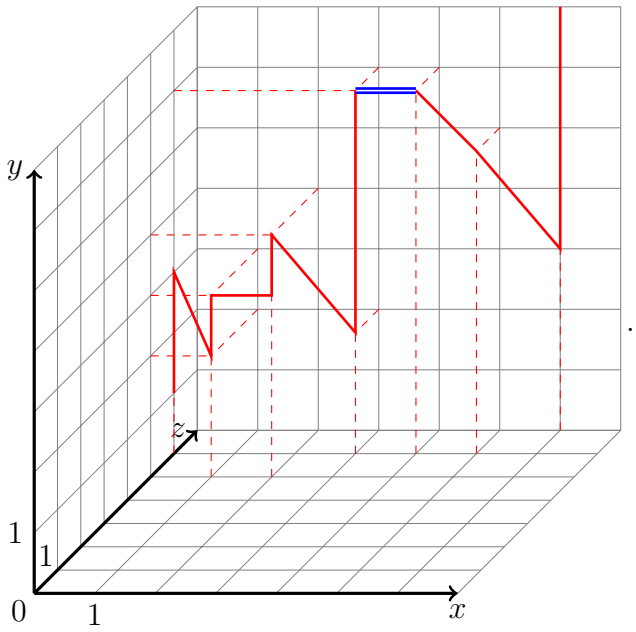
Exemple 3.2.2. *Nous dessinons la couleur* 0 *en rouge (ligne simple) et la couleur* 1 *en bleu (ligne double). L'ensemble*

$$\mathcal{L}\left(\left\{\begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\-1\\-1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\-2\\1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\\0\\0\end{bmatrix}\right\}^*, \begin{bmatrix}0\\1\\6\end{bmatrix}, \begin{bmatrix}6\\7\\7\end{bmatrix}\right)$$

contient le chemin

$$\left(\begin{bmatrix}0\\1\\6\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}^2\begin{bmatrix}1\\-1\\-1\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\begin{bmatrix}1\\-2\\1\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}^4\begin{bmatrix}1\\0\\0\\1\end{bmatrix}\begin{bmatrix}1\\-1\\0\\0\end{bmatrix}\begin{bmatrix}1\\-2\\1\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}^4\right) \tag{3.20}$$

dont la représentation est



. (3.21)

Nous allons encoder les prosuites en une famille particulière de chemins que nous appellerons *prochemins*.

3.2.2 Une bijection entre les prographes et des chemins

Nous notons $\mathbb{G}$ l'ensemble des générateurs

$$\left\{\overbrace{\begin{array}{c}\cdots\\\boxed{1}\\\cdots\end{array}}^{\alpha_1}, \ldots, \overbrace{\begin{array}{c}\cdots\\\boxed{m_1}\\\cdots\end{array}}^{\alpha_1}, \ldots, \overbrace{\begin{array}{c}\cdots\\\boxed{1}\\\cdots\end{array}}^{\alpha_d}, \ldots, \overbrace{\begin{array}{c}\cdots\\\boxed{m_d}\\\cdots\end{array}}^{\alpha_d}\right\}^{\beta_1 \quad \beta_1 \quad \beta_d \quad \beta_d},$$

où $\alpha_1, \ldots, \alpha_d, \beta_1, \ldots, \beta_d, m_1, \ldots, m_d \in \mathbb{N}_{\geq 1}$ et nous définissons l'application ω par

$$\begin{array}{rccc} \omega : & \mathbb{G} & \longrightarrow & \mathbb{Z}^3 \times \mathbb{N} \\ & \overbrace{\begin{array}{|c|}\hline \cdots \\ \hline j \\ \hline \cdots \\ \hline \end{array}}^{\beta_i} \atop \underbrace{\qquad\qquad}_{\alpha_i} & \mapsto & (1, 1-\alpha_i, \beta_i - \alpha_i, j) \end{array}.$$

Nous notons U le pas $(0, 1, 0, 0)$, D le pas $(0, -1, 0, 0)$ et $\mathcal{W}$ l'ensemble de pas $\{\omega(x) \,|\, x \in \mathbb{G}\}$.

Définition 3.2.3. *Notons* $\mathrm{PP}_{e,n,k,s}(\mathbb{G})$ *l'ensemble*

$$\mathcal{L}\left((U + \mathcal{W})^*, (0, 1, e), (n, k, s)\right)$$

et nous nommons ***prochemins*** *ses éléments.*

Les prochemins sont des raffinements des prographes.

Théorème 3.2.4. *Pour tout* $e, n, s \in \mathbb{N}$, *on a*

$$|\mathrm{PP}_{e,n,s,s}(\mathbb{G})| = |\mathrm{PG}_{e,n,s}(\mathbb{G})|.$$

Démonstration. Soit ϕ la fonction définie sur

$$\mathrm{PS}_{e,n,s}(\mathbb{G}) \mapsto \mathrm{PP}_{e,n,s,s}(\mathbb{G}), \quad e, n, s \in \mathbb{N},$$

qui associe à la prosuite

$$(e, (i_k, x_k))_{k \in [1,n]}$$

le chemin

$$\left((0, 1, e),\, U^{i_1 + \downarrow(x_1) - 1 - i_0} \omega(x_1) \cdots U^{i_n + \downarrow(x_n) - 1 - i_{n-1}} \omega(x_n)\, U^{s - i_n}\right).$$

La fonction ϕ est une bijection, nous concluons la preuve grâce au théorème 3.1.13. □

Intuitivement, l'abscisse d'un prochemin encode les types des générateurs du prographe qui lui est associé, son ordonnée encode les indices des opérations (3.4), sa cote encode les sorties et la couleur permet de distinguer les générateurs du même type.

Exemple 3.2.5. *Le prographe (3.6) est encodé par le prochemin*

$$\left((0, 1, 6),\, U^2\, \omega(\spadesuit)\, U\, \omega(\diamondsuit)\, U\, \omega(\heartsuit)\, U^4\, \omega(\blacklozenge)\, \omega(\clubsuit)\, \omega(\heartsuit)\, U^4\right),$$

qui est égal au chemin (3.20).

Les chemins sont des objets classiques en combinatoire (voir le chapitre 10 de [Bón15] pour une synthèse). Dans la prochaine partie, nous allons étudier les prochemins afin d'obtenir des résultats énumératifs sur les prographes.

3.3 Conséquences combinatoires

Nous allons faire une étude combinatoire classique des chemins sur les prochemins.

3.3.1 Formules de récurrence

Afin d'alléger les notations, nous posons

$$p(n,k,s) := |\mathrm{PP}_{e,n,k,s}(\mathbb{G})|,$$

où $e \in \mathbb{N}_{\geq 1}$ est un entier fixé.

Proposition 3.3.1. *Les prochemins vérifient la relation de récurrence :*

$$p(n,k,s) = \begin{cases} 1 & \text{si } n=0,\ k=1 \text{ et } s=e\,; \\ p(n,k-1,s) + \sum_{i=1}^{d} m_i\, p(n-1,k-1+\alpha_i,s-\beta_i+\alpha_i) & \\ \qquad \text{si } n \geq 0 \text{ et } 1 \leq k \leq s\,; & \\ 0 & \text{sinon.} \end{cases}$$

Démonstration. La relation de récurrence est une traduction directe de la grammaire non ambiguë suivante : un prochemin qui appartient à

$$\mathrm{PP}_{e,n,k,s}(\mathbb{G})$$

est

- soit un chemin réduit au point $(0,1,e)$;
- soit un chemin de
$$\mathrm{PP}_{e,n,k-1,s}(\mathbb{G})$$
concaténé avec le pas U ;
- soit un chemin de
$$\mathrm{PP}_{e,n-1,k-1+\alpha_i,s-\beta_i+\alpha_i}(\mathbb{G})$$
concaténé à un pas du type
$$(1, 1-\alpha_i, \beta_i-\alpha_i, c),$$
où c est l'une des m_i couleurs.

□

D'après le théorème 3.2.4, il suffit de spécialiser k à s dans la proposition 3.3.1 afin d'obtenir une relation de récurrence sur les prographes. Le théorème ci-dessous nous fournit une relation de récurrence sur les prographes en nous débarrassant du paramètre de raffinement k.

Théorème 3.3.2. *La suite*

$$a_{n,s} := |\mathrm{PP}_{e,n,s,s}(\mathbb{G})| = |\mathrm{PG}_{e,n,s}(\mathbb{G})|$$

vérifie la relation de récurrence :

$$a_{n,s}=\begin{cases} 1 & \text{si } n=0 \text{ et } s=e\,; \\ \displaystyle\sum_{\ell=1}^{n}(-1)^{\ell+1}\sum_{c_1+\cdots+c_d=\ell}\binom{\ell}{c_1,\ldots,c_d}\binom{s+\ell-\sum_{i=1}^{d}c_i\beta_i}{\ell}m_1^{c_1}\ldots m_d^{c_d}a_{n-\ell,\,s-\sum_{i=1}^{d}c_i(\beta_i-\alpha_i)} & \\ & \text{si } n,s\geq 1\,; \\ 0 & \text{sinon.} \end{cases}$$

Démonstration. Soit $n, s \geq 1$. Pour $\ell \in [0, n]$, nous notons A_ℓ l'ensemble

$$\mathcal{L}\left((U^*\mathcal{W})^{n-\ell}\,(DD^*\mathcal{W})^{\ell}\,U^*, (0,1,e), (n,s,s)\right).$$

On a que $|A_n| = 0$, $a_{n,s} = |A_0|$ et pour $0 \leq \ell \leq n-1$,

$$|A_\ell| = |A_\ell \sqcup A_{\ell+1}| - |A_{\ell+1}|.$$

Nous obtenons donc par itération que

$$a_{n,s} = \sum_{\ell=1}^{n}(-1)^{\ell+1}|A_{\ell-1} \sqcup A_\ell|.$$

De plus, pour $\ell \in [1, n]$, $A_{\ell-1} \sqcup A_\ell$ est égal à

$$\mathcal{L}\left((U^*\mathcal{W})^{n-\ell}(U^* + DD^*)\mathcal{W}(DD^*\mathcal{W})^{\ell-1}U^*, (0,1,e), (n,s,s)\right).$$

Donc $|A_{\ell-1} \sqcup A_\ell|$ se décompose en

$$\sum_{j\geq 0} a_{n-\ell,j}\,\left|\mathcal{L}\left(D^*\mathcal{W}(DD^*\mathcal{W})^{\ell-1}D^*, (n-\ell,j,j), (n,1,s)\right)\right|.$$

Calculons le cardinal de l'ensemble

$$\mathcal{L}\left(D^*\mathcal{W}(DD^*\mathcal{W})^{\ell-1}D^*, (n-\ell,j,j), (n,1,s)\right). \tag{3.22}$$

Pour construire un élément de l'ensemble (3.22), nous choisissons en premier ℓ pas de $\mathcal{W}$. Pour une composition

$$c_1 + \cdots + c_d = \ell,$$

nous prenons c_i pas

$$(1, 1-\alpha_i, \beta_i - \alpha_i, c),$$

où c est une des m_i couleurs. Il y a donc

$$\binom{\ell}{c_1,\ldots,c_d} m_1^{c_1}\ldots m_d^{c_d}$$

façons de choisir les pas de $\mathcal{W}$. Nous devons maintenant ajouter les pas D. Notons b le nombre de pas que nous devons ajouter pour produire un chemin de (3.22). Nécessairement,

$$j = s - \sum_{i=1}^{d} c_i(\beta_i - \alpha_i)$$

et

$$b = \ell - 1 + s - \sum_{i=1}^{d} c_i\beta_i.$$

Les $\ell - 1$ premiers pas de $\mathcal{W}$ dans (3.22) sont suivis par un pas D. Il nous reste donc à placer

$$s - \sum_{i=1}^{d} c_i\beta_i$$

pas D dans $\ell + 1$ emplacements possibles, ce qui fait

$$\binom{s+\ell-\sum_{i=1}^{d} c_i\beta_i}{\ell}$$

possibilités. Donc $|A_{\ell-1} \sqcup A_\ell|$ vaut

$$\sum_{c_1+\cdots+c_d=\ell} \binom{\ell}{c_1,\ldots,c_d} \binom{s+\ell-\sum_{i=1}^{d} c_i\beta_i}{\ell} m_1^{c_1}\ldots m_d^{c_d}\, a_{n-\ell,s-\sum_{i=1}^{d} c_i(\beta_i-\alpha_i)}.$$

Ceci conclut la démonstration. □

Exemple 3.3.3. *Le nombre de sorties d'un prographe dont les générateurs sont de l'ensemble*

est déterminé par son nombre d'entrées et de générateurs. En effet, en ajoutant un générateur à un de ces prographes on augmente de 1 *son nombre de sorties. L'étude combinatoire de ces prographes à une entrée revient à étudier la suite*

$$a_n := \left|\mathrm{PG}_{1,n,n+1}\left(\left\{\,,\,\right\}\right)\right|,$$

où $n \geq 0$.

La suite $(a_n)_{n\geq 0}$ *est égale à la suite de Heinz, dont la référence est A*224776 *dans [Inc]. En effet, elles vérifient les mêmes relations de récurrence et les mêmes cas initiaux (voir la proposition 3.3.1 et la référence A*224776 *de [Inc]). D'après le théorème 3.3.2, nous obtenons le nouveau résultat suivant*

$$a_n = \begin{cases} 1 & si\ n = 0; \\ \sum_{k=1}^{n} (-1)^{k+1} q(n,k) a_{n-k} & si\ n \geq 1; \\ 0 & sinon, \end{cases}$$

où

$$q(n,k) = \sum_{\ell=1}^{k} \binom{n+1+\ell-2k}{k} \binom{k}{\ell}.$$

La suite

$$(q(n+2k-1,k))_{k\geq 0}$$

est connue sous le nom anglais de crystal ball sequence pour le treillis cubique de dimension n [CS97].

3.3.2 Équation fonctionnelle

Nous allons dans cette partie calculer une équation fonctionnelle vérifiée par la série génératrice des prographes

$$A_e(x,z) := \sum_{(n,s)\in\mathbb{N}^2} |\mathrm{PG}_{e,n,s}(\mathbb{G})| x^n z^s, \tag{3.23}$$

pour $e \in \mathbb{N}_{\geq 1}$. Nous fixons e pour le reste de cette partie et notons plus simplement $A(x,z)$ la série $A_e(x,z)$. Nous supposons jusqu'à la fin de ce chapitre que $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_d$.

Notre première equation fonctionnelle implique $A(x,z)$ et la série génératrice des prochemins

$$P(x,y,z) := \sum_{(k,n,s)\in\mathbb{N}^3} |\mathrm{PP}_{e,n,k,s}(\mathbb{G})| x^n y^k z^s.$$

Proposition 3.3.4. *Nous avons la relation fonctionnelle*

$$\left(1 - \sum_{j=1}^{d} m_j x y^{1-\alpha_j} z^{\beta_j-\alpha_j} - y\right) P(x,y,z) = yz^e - yA(x,yz) - \sum_{j=1}^{d} m_j x y^{1-\alpha_j} z^{\beta_j-\alpha_j} \sum_{k=1}^{\alpha_j-1} y^k L_k(x,z), \tag{3.24}$$

où

$$L_k(x,z) := \sum_{(n,s)\in\mathbb{N}^2} p(n,k,s)x^n z^s$$

pour tout $k \in [1, \alpha_1 - 1]$.

Démonstration. Par une traduction directe dans le monde des séries de la grammaire exhibée dans la démonstration de la proposition 3.3.1, nous obtenons que

$$P(x,y,z) = x^0 y z^e + y\left(P(x,y,z) - A(x,yz)\right) + \sum_{j=1}^{d} m_j x y^{1-\alpha_j} z^{\beta_j-\alpha_j}\left(P(x,y,z) - \sum_{k=1}^{\alpha_j-1} y^k L_k(x,z)\right).$$

On conclut la démonstration en simplifiant cette dernière relation. □

Nous allons maintenant utiliser la méthode du noyau [Knu68, BMP00] afin de transformer la relation (3.24) en une équation fonctionnelle n'impliquant que la série $A(x,z)$. La première étape de cette méthode est de calculer les racines de l'expression

$$1 - \sum_{j=1}^{d} m_j x y^{1-\alpha_j} z^{\beta_j-\alpha_j} - y, \tag{3.25}$$

où y est l'inconnue. Nous avons trouvé les solutions suivantes sous la forme de séries de Newton-Puiseux.

Lemme 3.3.5. *Posons*

$$y_0(x,z) := 1 - \sum_{n\geq 1} \sum_{n_1+\cdots+n_d=n} \binom{\sum_{i=1}^{d} n_i\alpha_i - 2}{n_1,\ldots,n_d,\ \sum_{i=1}^{d} n_i\alpha_i - n - 1} m_1^{n_1}\cdots m_d^{n_d} z^{\sum_{i=1}^{d} n_i(\beta_i-\alpha_i)} x^n$$

et pour $k \in [1, \alpha_1 - 1]$, *posons*

$$y_k(x,z) := \frac{1}{\alpha_1 - 1} \sum_{n\geq 1} y^{(n)}(z)\left(e^{\frac{2\pi i k}{\alpha_1-1}} x\right)^n,$$

où $y^{(n)}(z)$ *est la somme sur les d-uplets* $(n_1,\ldots,n_d)$ *tels que*

$$n_1 + n_2(\alpha_1 - \alpha_2) + \cdots + n_d(\alpha_1 - \alpha_d) = n - 1$$

de

$$\binom{\frac{n}{\alpha_1-1} + n_1 - 1}{n_1,\ldots,n_d} m_2^{n_2}\cdots m_d^{n_d} z^{\sum_{i=2}^{d} n_i(\beta_i-\alpha_i)} \left(m_1 z^{\beta_1-\alpha_1}\right)^{\frac{n}{\alpha_1-1}-n_2-\cdots-n_d}.$$

L'ensemble des racines de l'expression (3.25) est

$$\{y_0(x,z), y_1(x,z), \ldots, y_{\alpha_1-1}(x,z)\}\,.$$

Démonstration. L'équation (3.25) est équivalente à

$$x\left(\sum_{j=1}^{d} m_j (1-y')^{1-\alpha_j} z^{\beta_j-\alpha_j}\right) = y',$$

où $y' = 1-y$. D'après le théorème d'inversion de Lagrange, on sait que

$$[x^n]\, y = -[x^n]\, y' = -\frac{1}{n}\left[(y')^{n-1}\right]\left(\sum_{j=1}^{d} m_j (1-y')^{1-\alpha_j} z^{\beta_j-\alpha_j}\right)^n,$$

pour $n \geq 1$. Cette dernière expression est égale à

$$-\frac{1}{n}\left[(y')^{n-1}\right] \sum_{n_1+\cdots+n_d=n} \binom{n}{n_1,\ldots,n_d} \prod_{i=1}^{d} \left(m_i (1-y')^{1-\alpha_i} z^{\beta_i-\alpha_i}\right)^{n_i}$$

Nous obtenons par simplification la solution $y_0(x,z)$.

De plus, l'équation (3.25) est équivalente à

$$x'\left(\frac{m_1 z^{\beta_2-\alpha_2} + \sum\limits_{j=2}^{d} m_j y^{\alpha_1-\alpha_j} z^{\beta_j-\alpha_j}}{1-y}\right)^{\frac{1}{\alpha_1-1}} = y,$$

où $x' = x^{\frac{1}{\alpha_1-1}}$. D'après le théorème d'inversion de Lagrange

$$\begin{aligned}\left[x^{\frac{n}{\alpha_1-1}}\right] y &= \left[(x')^n\right] e^{\frac{2\pi i k n}{\alpha_1-1}} y \\ &= \frac{e^{\frac{2\pi i k n}{\alpha_1-1}}}{n}\,[y^{n-1}]\left(\frac{m_1 z^{\beta_1-\alpha_1} + \sum\limits_{j=2}^{d} m_j y^{\alpha_1-\alpha_j} z^{\beta_j-\alpha_j}}{1-y}\right)^{\frac{n}{\alpha_1-1}},\end{aligned}$$

pour tout $n \geq 1$, Nous obtenons par simplification les solutions $y_k(x,z)$. □

Nous énonçons enfin une équation fonctionnelle sous forme de déterminant pour la série $A(x,z)$.

Théorème 3.3.6. *Soit*

$$c_i(j,k) := \sum_{r=1}^{j} m_r z^{\beta_r-\alpha_r} y_i^{k-\alpha_r},$$

le déterminant

$$\left|\begin{array}{ccc|c|ccc|c} c_1(d,1) & \cdots & c_1(d,\alpha_d{-}1) & & c_1(1,\alpha_2) & \cdots & c_1(1,\alpha_1{-}1) & z^e{-}A(x,y_0z) \\ & \vdots & & \cdots & & \vdots & & \vdots \\ c_{\alpha_1}(d,1) & \cdots & c_{\alpha_1}(d,\alpha_d{-}1) & & c_{\alpha_1}(1,\alpha_2) & \cdots & c_{\alpha_1}(1,\alpha_1{-}1) & z^e{-}A(x,y_{\alpha_1-1}z) \end{array}\right|$$

est nul.

Démonstration. D'après la proposition 3.3.4 et le lemme 3.3.5, nous savons que pour tout $i \in [0, \alpha_1 - 1]$,

$$\sum_{j=1}^{d} \sum_{k=1}^{\alpha_j - 1} m_j y_i^{k-\alpha_j} z^{\beta_j - \alpha_j} \left(xL_k(x,z)\right) = z^e - A(x, y_i z).$$

Donc

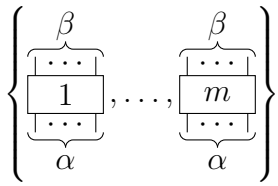
$$\sum_{j=d}^{1} \sum_{k=\alpha_{j+1}}^{\alpha_j - 1} c_i(j,k) \left(xL_k(x,z)\right) = z^e - A(x, y_i z).$$

Les colonnes du déterminant sont donc linéairement dépendantes. Ceci conclut la démonstration. □

Il est possible de développer ce déterminant avec une approche similaire à celle classiquement utilisée pour calculer le déterminant de Vandermonde.

3.3.3 Formules closes

Nous nous restreignons dans cette partie au cas $d = 1$. Ce qui signifie que l'ensemble des générateurs $\mathbb{G}$ est de la forme



Dans ce cas, le nombre de sorties des prographes engendrés par $\mathbb{G}$ est déterminé par les autres paramètres (comme dans l'exemple 3.3.3). En effet, ces prographes avec e entrées et n générateurs vont nécessairement avoir

$$e + (\beta - \alpha)n$$

sorties. Car chaque générateur de $\mathbb{G}$ va retirer α sorties et en créer β nouvelles. Nous allons donc dans cette partie étudier les termes

$$a_n := \left|\mathrm{PP}_{e,\,n,\,e+(\beta-\alpha)n,\,e+(\beta-\alpha)n}(\mathbb{G})\right| = \left|\mathrm{PG}_{e,\,n,\,e+(\beta-\alpha)n}(\mathbb{G})\right|.$$

Exemple 3.3.7. *Grâce à l'encyclopédie OEIS [Inc], nous avons découvert que les prographes pour le cas particulier*

$$m = 1,\ \alpha = 2,\ \beta = 3 \text{ et } e = 2$$

sont en bijection avec les arbres tandem de duplication [Gas05]. Plus précisément, pour tout $n \geq 0$,

$$\left|\mathrm{PG}_{2,n,2+n}\left(\left\{\right\}\right)\right| \tag{3.26}$$

est égal au nombre d'arbres tandem enracinés de duplication sur $n+1$ gènes. En effet, ils vérifient la même relation de récurrence et ont le même cas initial (voir la proposition 3.3.1 et [GHJMM03]).

Le théorème suivant nous fournit une première formule close sous forme de déterminant.

Théorème 3.3.8. *Pour $n \geq 0$, on a*

$$a_n = m^n det\left(\binom{e - i(\alpha-1) + (j-1)(\beta-1)}{i-j+1}_{(i,j)\in[1,n]^2}\right).$$

Démonstration. Pour le cas particulier $d = 1$, le théorème 3.3.2 peut se réécrire sous la forme matricielle suivante :

$$A[a_0, \ldots, a_n]^t = [1, 0, \ldots, 0]^t,$$

où A est la matrice dont les coefficients sont

$$A_{i,j} := (-1)^{i-j} m \binom{e + (i+1)(\alpha-1) + j(\beta-1)}{i-j}.$$

D'après la règle de Cramer, on a

$$a_n = \frac{A_{n+1}}{det(A)},$$

où A_{n+1} est la matrice A dont la dernière colonne est remplacée par le vecteur

$$[1, 0, \ldots, 0].$$

On sait que $det(A) = 1$ donc en développant suivant la dernière colonne on obtient que

$$a_n = (-1)^n B,$$

où B est le déterminant de la matrice

$$\left((-1)^{i-j} m \binom{e-i(\alpha-1)+(j-1)(\beta-1)}{i-j+1}\right)_{(i,j)\in[1,n]^2}.$$

On obtient la formule recherchée en multipliant par -1 toutes les colonnes d'indices impairs et toutes les lignes d'indices pairs de cette matrice. □

Nous concluons ce chapitre en proposant une deuxième formule close sous l'aspect d'une série alternée.

Théorème 3.3.9. *Pour $n \geq 0$, on a*

$$a_n = m^n \sum_{k\geq 1} (-1)^{n+k} \sum_{\substack{t_1+\cdots+t_k=n \\ t_i\neq 0}} \prod_{j=1}^{k} \binom{(\beta-\alpha)(t_1+\cdots+t_{j-1})+e-(\alpha-1)t_j}{t_j}.$$

Démonstration. Pour $i \in [1, n-1]$, notons A_i l'ensemble

$$\mathcal{L}\left(((D^*+U^*)\,\mathcal{W})^i\, DD^*\mathcal{W}\,((D^*+U^*)\,\mathcal{W})^{n-i-1}\, U^*, (0,1,e), (n,s,s)\right)$$

et E l'ensemble

$$\mathcal{L}\left(((D^*+U^*)\,\mathcal{W})^n\, U^*, (0,1,e), (n,s,s)\right).$$

On remarque que

$$a_n = \left|\bigcap_{i=1}^{n-1} A_i\right| = |E| - \left|\bigcup_{i=1}^{n-1} A_i\right|$$

et d'après le principe d'inclusion–exclusion

$$\left|\bigcup_{k=1}^{n-1} A_k\right| = \sum_{k\geq 1}^{n-1} (-1)^{k-1}\Omega_k,$$

où Ω_k est

$$\sum_{1\leq i_1<\cdots<i_k\leq n} |A_{i_1}\cap\cdots\cap A_{i_k}|.$$

Pour $k \in [1, n-1]$, nous notons $A_{t_1,\ldots,t_k}$ l'ensemble

$$\mathcal{L}\left(U^*\,[\mathcal{W}, DD^*]^{t_1}\,(DD^*+U^*)\,[\mathcal{W}, DD^*]^{t_2}\cdots[\mathcal{W}, DD^*]^{t_k}\,U^*, (0,1,e), (n,s,s)\right),$$

où pour $v \in \mathbb{N}_{\geq 1}$, $[\mathcal{W}, DD^*]^v$ désigne $(\mathcal{W}DD^*)^{v-1}\mathcal{W}$. L'entier Ω_k peut se réécrire

$$\sum_{\substack{t_1+\cdots+t_{n-k}=n \\ t_i\neq 0}} |A_{t_1,\ldots,t_{n-k}}|$$

et l'ensemble E est identique à l'ensemble

$$A_{\underbrace{1,\ldots,1}_{n}}.$$

Grâce au changement de la variable k en $n-k$, on obtient que

$$a_n = \sum_{k=1}^{n} (-1)^{n+k} \sum_{\substack{t_1+\cdots+t_k=n \\ t_i\neq 0}} |A_{t_1,\ldots,t_k}|.$$

Cependant $|A_{t_1,\ldots,t_k}|$ est égal à

$$\prod_{j=1}^{k} \left| \mathcal{L}\left(D^*[W, DD^*]^{t_j} D^*, p(j), p(j+1)\right)\right|,$$

où

$$p(j) = \left(\sum_{\ell=1}^{j-1} t_\ell, e + \sum_{\ell=1}^{j-1} t_\ell(\beta-\alpha), e + \sum_{\ell=1}^{j-1} t_\ell(\beta-\alpha)\right).$$

Par un calcul similaire à celui du cardinal de (3.22) nous obtenons que $|A_{t_1,\ldots,t_k}|$ vaut

$$\prod_{j=1}^{k} \binom{e + (\beta-\alpha)\left(t_1+\cdots+t_{j-1}\right) - (\alpha-1)t_j}{t_j}.$$

Ce qui conclut la démonstration. □

Chapitre 4

Réécriture dans des quotients magmatiques

Les opérades peuvent être vues comme des cas particuliers des prographes à une sortie, où tous les générateurs ont également une seule sortie. Cependant, la théorie des opérades et celle des prographes se sont développées relativement séparément. Nous allons donc introduire le concept d'opérade dans le langage classique de la théorie des opérades et non dans celui de la théorie des prographes.

4.1 Opérades et réécriture

4.1.1 Opérade

Une ***opérade*** (non symétrique) est un ensemble gradué

$$\mathcal{O} := \bigsqcup_{n\geq 1} \mathcal{O}(n)$$

qui possède un élément $\mathbb{1} \in \mathcal{O}(1)$ appelé ***unité*** et qui est muni d'applications

$$\circ_i : \mathcal{O}(n) \times \mathcal{O}(m) \to \mathcal{O}(n+m-1), \qquad 1 \leq i \leq n, 1 \leq m,$$

appelées ***compositions partielles***. Ces éléments doivent vérifier les trois relations :

$$(x \circ_i y) \circ_{i+j-1} z = x \circ_i (y \circ_j z), \qquad 1 \leq i \leq n, 1 \leq j \leq m,$$

$$(x \circ_i y) \circ_{j+m-1} z = (x \circ_j z) \circ_i y, \qquad 1 \leq i < j \leq n,$$

$$\mathbb{1} \circ_1 x = x = x \circ_i \mathbb{1}, \qquad 1 \leq i \leq n,$$

pour tout $x \in \mathcal{O}(n)$, $y \in \mathcal{O}(m)$ et $z \in \mathcal{O}$.

Introduisons quelques notations et définitions élémentaires au sujet des opérades. Étant donné un élément $x \in \mathcal{O}$, on appelle l'***arité*** de x et on la note $|x|$ l'entier tel que $x \in \mathcal{O}(|x|)$.

La ***composition complète*** de $\mathcal{O}$ est l'application

$$\circ : \mathcal{O}(n) \times \mathcal{O}(m_1) \times \cdots \times \mathcal{O}(m_n) \to \mathcal{O}(m_1 + \cdots + m_n)$$

définie pour tout $x \in \mathcal{O}(n)$ et $y_1, \ldots, y_n \in \mathcal{O}$ par

$$x \circ [y_1, \ldots, y_n] := (\cdots((x \circ_n y_n) \circ_{n-1} y_{n-1}) \cdots) \circ_1 y_1.$$

Soient $\mathcal{O}_1$ et $\mathcal{O}_2$ deux opérades. On dit qu'une application

$$\phi : \mathcal{O}_1 \to \mathcal{O}_2$$

est un ***morphisme d'opérades*** si elle respecte l'arité, envoie l'unité de $\mathcal{O}_1$ sur l'unité de $\mathcal{O}_2$ et commute avec les compositions partielles. Par opposition, on appelle ***anti-morphisme d'opérades*** une application

$$\phi : \mathcal{O}_1 \to \mathcal{O}_2$$

qui respecte l'arité, envoie l'unité de $\mathcal{O}_1$ sur l'unité de $\mathcal{O}_2$ et telle que

$$\phi(x \circ_i y) = \phi(x) \circ_{|x|-i+1} \phi(y)$$

pour tout $x, y \in \mathcal{O}_1$ et $i \in [1, |x|]$.

Une ***congruence*** d'opérade est une relation d'équivalence $\equiv$ sur $\mathcal{O}$ qui respecte les arités et telle que pour tout $x, x', y, y' \in \mathcal{O}$,

$$x \equiv x' \text{ et } y \equiv y'$$

implique

$$x \circ_i y \equiv x' \circ_i y'$$

lorsque i est un entier valide. Étant donnée une congruence d'opérade $\equiv$, l'opérade quotient $\mathcal{O}/_{\equiv}$ de $\mathcal{O}$ par $\equiv$ est l'opérade des classes d'équivalence de $\equiv$.

Lorsque tous les ensembles $\mathcal{O}(n)$ sont finis, on appelle ***série de Hilbert*** de $\mathcal{O}$ la série

$$\mathcal{H}_{\mathcal{O}}(t) := \sum_{n \geq 1} |\mathcal{O}(n)| \ t^n.$$

4.1.2 Arbre binaire et opérade magmatique

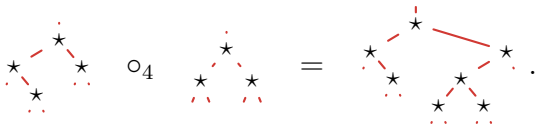
On définit formellement un ***arbre binaire*** comme une feuille | ou un couple $(\mathfrak{t}_1, \mathfrak{t}_2)$ d'arbres binaires. Nous allons utiliser le vocabulaire standard de la théorie des arbres (tel que racine, nœud, fils gauche, fils droit, ...). L'arité $|\mathfrak{t}|$ d'un arbre binaire $\mathfrak{t}$ est son nombre de feuilles et son degré $\deg(\mathfrak{t})$ est son nombre de nœuds. On a pour habitude de dessiner la racine d'un arbre en haut, par exemple

est la représentation graphique de l'arbre binaire $((|, (|, |)), (|, |))$.

L'***opérade magmatique*** **Mag** est l'ensemble gradué de tous les arbres binaires, où $\mathbf{Mag}(n)$ est l'ensemble des arbres binaires d'arité n. Les compositions partielles de **Mag** sont les greffes d'arbres : étant donnés deux arbres $\mathfrak{t}$ et $\mathfrak{s}$, l'arbre

$$\mathfrak{t} \circ_i \mathfrak{s}$$

est l'arbre binaire obtenu en greffant l'arbre $\mathfrak{s}$ sur la i^e feuille de l'arbre $\mathfrak{t}$. Par exemple,

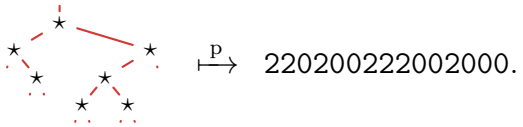


L'unité de **Mag** est la feuille.

Le nombre d'arbres binaires d'arité $n \geq 1$ est le $(n-1)^e$ nombre de Catalan. La série de Hilbert de **Mag** est donc

$$\mathcal{H}_{\mathbf{Mag}}(t) = \sum_{n\geq 1} \mathrm{cat}(n-1)t^n = \sum_{n\geq 1} \binom{2n-2}{n-1} \frac{1}{n} t^n.$$

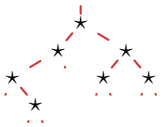
Nous allons maintenant introduire des outils sur les arbres binaires. Étant donné un arbre $\mathfrak{t}$, nous notons $\mathrm{p}(\mathfrak{t})$ son ***parcours préfixe***. Nous représentons le parcours préfixe d'un arbre par un mot sur l'alphabet $\{0, 2\}$, où 0 correspond à une feuille et 2 à un nœud. Par exemple,



Nous définissons la relation d'ordre $\leq$ sur les arbres par

$$\mathfrak{t} \leq \mathfrak{t}'$$

si et seulement si $|\mathfrak{t}| = |\mathfrak{t}'|$ et $\mathrm{p}(\mathfrak{t})$ est plus petit que $\mathrm{p}(\mathfrak{t}')$ pour l'ordre lexicographique induit par $\mathtt{0} < \mathtt{2}$. Nous définissons également le ***rang gauche*** d'un arbre $\mathfrak{t}$ comme son nombre $\mathrm{lr}(\mathfrak{t})$ de nœuds présent sur sa branche gauche principale (celle commençant à sa racine). Par exemple,



$\longmapsto 3.$

Formellement, $\mathrm{lr}(\mathfrak{t})$ est la longueur du plus long préfixe de $\mathrm{p}(\mathfrak{t})$ qui soit uniquement composé de $\mathtt{2}$.

Un arbre $\mathfrak{s}$ est un ***sous-arbre*** de $\mathfrak{t}$ s'il est possible de superposer $\mathfrak{s}$ sur $\mathfrak{t}$. Formellement, $\mathfrak{s}$ est un sous-arbre de $\mathfrak{t}$ s'il existe

$$\mathfrak{r}, \mathfrak{r}_1, \ldots, \mathfrak{r}_n \in \mathbf{Mag}$$

tels que

$$\mathfrak{t} = \mathfrak{r} \circ_i (\mathfrak{s} \circ [\mathfrak{r}_1, \ldots, \mathfrak{r}_n]) .$$

Quand au contraire $\mathfrak{s}$ n'est pas un sous-arbre de $\mathfrak{t}$ alors on dit que $\mathfrak{t}$ ***évite*** $\mathfrak{s}$.

4.1.3 Réécriture dans les arbres binaires

Nous présentons dans cette partie la notion de réécriture [BN98] dans les arbres binaires.

Une ***règle de réécriture*** est un couple $(\mathfrak{s}, \mathfrak{s}')$ d'arbres binaires tel que $|\mathfrak{s}| = |\mathfrak{s}'|$. Un ensemble S de règles de réécriture est une relation binaire sur **Mag** et on aura tendance à le noter $\to$. On note

$$\mathfrak{s} \to \mathfrak{s}'$$

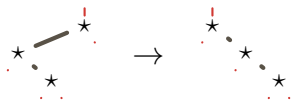
le fait que $(\mathfrak{s}, \mathfrak{s}') \in \to$. Si $\to$ est un ensemble de règles de réécriture alors nous notons $\Rightarrow$ la ***relation de réécriture induite*** par $\to$. Plus formellement,

$$\mathfrak{t} \circ_i (\mathfrak{s} \circ [\mathfrak{r}_1, \ldots, \mathfrak{r}_n]) \Rightarrow \mathfrak{t} \circ_i (\mathfrak{s}' \circ [\mathfrak{r}_1, \ldots, \mathfrak{r}_n]) ,$$

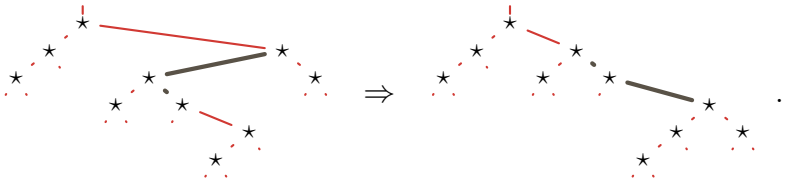
si $\mathfrak{s} \to \mathfrak{s}'$ avec $n = |\mathfrak{s}|$ et $\mathfrak{t}$, $\mathfrak{r}_1$, …, $\mathfrak{r}_n \in \mathbf{Mag}$. En d'autres termes, on a

$$\mathfrak{t} \Rightarrow \mathfrak{t}'$$

s'il est possible d'obtenir $\mathfrak{t}'$ à partir de $\mathfrak{t}$ en remplaçant un sous-arbre $\mathfrak{s}$ de $\mathfrak{t}$ par $\mathfrak{s}'$ quand $\mathfrak{s} \to \mathfrak{s}'$. Par exemple, si $\to$ contient la règle



alors



Quand $\mathfrak{t}_0$, $\mathfrak{t}_1$, …, $\mathfrak{t}_k$ sont des arbres binaires tels que

$$\mathfrak{t}_0 \Rightarrow \mathfrak{t}_1 \Rightarrow \cdots \Rightarrow \mathfrak{t}_k,$$

on dit que $\mathfrak{t}_0$ se ***réécrit*** en $\mathfrak{t}_k$ par $\Rightarrow$ en k ***étapes***. On dit que $\Rightarrow$ est ***terminant*** s'il n'existe pas de chaîne infinie

$$\mathfrak{t}_0 \Rightarrow \mathfrak{t}_1 \Rightarrow \mathfrak{t}_2 \Rightarrow \cdots$$

Nous utiliserons le critère classique ci-dessous pour montrer qu'une relation de réécriture est terminante.

Lemme 4.1.1. *Soit $\to$ un ensemble de règles de réécriture sur* **Mag**. *Si pour tout $\mathfrak{t} \to \mathfrak{t}'$ on a $\mathfrak{t} > \mathfrak{t}'$ alors la relation de réécriture induite par $\to$ est terminante.*

Une ***forme normale*** pour $\Rightarrow$ est un arbre binaire $\mathfrak{t}$ tel que $\mathfrak{t} \overset{*}{\Rightarrow} \mathfrak{t}'$ implique que $\mathfrak{t}' = \mathfrak{t}$. En d'autres termes, une forme normale est un arbre qui ne peut pas être réécrit. On note $\mathfrak{N}_\Rightarrow$ l'ensemble des formes normales. L'ensemble $\mathfrak{N}_\Rightarrow$ est également l'ensemble des arbres binaires qui évitent les membres gauches des règles de $\to$.

Si pour tous les arbres binaires $\mathfrak{t}$, $\mathfrak{s}_1$ et $\mathfrak{s}_2$ tels que $\mathfrak{t} \overset{*}{\Rightarrow} \mathfrak{s}_1$ et $\mathfrak{t} \overset{*}{\Rightarrow} \mathfrak{s}_2$ on a un arbre $\mathfrak{t}'$ tel que $\mathfrak{s}_1 \overset{*}{\Rightarrow} \mathfrak{t}'$ et $\mathfrak{s}_2 \overset{*}{\Rightarrow} \mathfrak{t}'$ alors on dit que $\Rightarrow$ est ***confluent***. Quand $\Rightarrow$ est terminant et confluent on dit que $\Rightarrow$ est ***convergent***.

4.2 Opérades peignes

Nous rappelons que l'***opérade associative*** **As** est le quotient de **Mag** par la plus petite congruence d'opérade $\equiv$ vérifiant

Nous proposons dans cette partie d'étudier une généralisation de l'opérade **As**.

Pour tout entier $\gamma \geq 1$, les arbres ***peigne gauche*** $\mathfrak{p}^{(\gamma)}_{/}$ et ***peigne droit*** $\mathfrak{p}^{(\gamma)}_{\backslash}$ sont les arbres

de degré γ. Pour $\gamma \geq 1$, nous appelons l'***opérade peigne*** de taille γ (que nous notons $\mathbf{CAs}^{(\gamma)}$) l'opérade quotient $\mathbf{Mag}/_{\equiv^{(\gamma)}}$, où $\equiv^{(\gamma)}$ est la plus petite congruence sur **Mag** telle que

$$\mathfrak{p}^{(\gamma)}_{/} \equiv^{(\gamma)} \mathfrak{p}^{(\gamma)}_{\backslash}.$$

Remarquons que $\mathbf{CAs}^{(1)}$ est trivialement isomorphe à **Mag** et que $\mathbf{CAs}^{(2)}$ est l'opérade **As**. Nous posons

$$\mathbf{CAs} := \left\{\mathbf{CAs}^{(\gamma)} : \gamma \geq 1\right\},$$

l'ensemble des opérades peignes.

4.2.1 Le treillis des opérades peignes

Afin d'introduire une structure de treillis sur **CAs**, nous allons dans un premier temps étudier les morphismes d'opérades entre ses éléments.

Lemme 4.2.1. *Soient γ et γ' deux entiers positifs. Il existe un morphisme d'opérades*

$$\varphi : \mathbf{CAs}^{(\gamma')} \to \mathbf{CAs}^{(\gamma)}$$

si et seulement si

$$\mathfrak{p}^{(\gamma')}_{/} \equiv^{(\gamma)} \mathfrak{p}^{(\gamma')}_{\backslash}.$$

Si un tel morphisme existe alors il est nécessairement surjectif.

Démonstration. Supposons que

$$\varphi : \mathbf{CAs}^{(\gamma')} \to \mathbf{CAs}^{(\gamma)}$$

est un morphisme d'opérades. Puisque

$$\left|\mathbf{CAs}^{(\gamma)}(2)\right| = \left|\mathbf{CAs}^{(\gamma')}(2)\right| = 1,$$

l'application φ envoie nécessairement le générateur de l'opérade $\mathbf{CAs}^{(\gamma')}$ sur celui de l'opérade $\mathbf{CAs}^{(\gamma)}$. L'application φ est donc surjective et

$$\varphi\left(\left[\mathfrak{p}_{/}^{(\gamma')}\right]_{\equiv(\gamma')}\right) = \left[\mathfrak{p}_{/}^{(\gamma')}\right]_{\equiv(\gamma)}. \tag{4.2}$$

Par définition

$$\mathfrak{p}_{/}^{(\gamma')} \equiv^{(\gamma')} \mathfrak{p}_{\backslash}^{(\gamma')},$$

nous en déduisons donc d'après (4.2) que

$$\left[\mathfrak{p}_{/}^{(\gamma')}\right]_{\equiv(\gamma)} = \left[\mathfrak{p}_{\backslash}^{(\gamma')}\right]_{\equiv(\gamma)}.$$

Ce qui conclut la première partie de cette démonstration.

Réciproquement, si

$$\mathfrak{p}_{/}^{(\gamma')} \equiv^{(\gamma)} \mathfrak{p}_{\backslash}^{(\gamma')}$$

alors la relation $\equiv^{(\gamma)}$ est moins fine que la relation $\equiv^{(\gamma')}$. L'application

$$\varphi : \mathbf{CAs}^{(\gamma')}(2) \to \mathbf{CAs}^{(\gamma)}(2)$$

définie par

$$\varphi\left([\star]_{\equiv(\gamma')}\right) := [\star]_{\equiv(\gamma)}$$

s'étend de façon unique en un morphisme (surjectif) d'opérades. Ce qui conclut la démonstration. □

Le lemme suivant énonce une propriété nécessaire sur l'appartenance à une même classe d'équivalence de $\equiv^{(\gamma)}$.

Lemme 4.2.2. *Soient $\mathfrak{t}, \mathfrak{t}'$ deux arbres et $\gamma \geq 2$ un entier. Si $\mathfrak{t} \equiv^{(\gamma)} \mathfrak{t}'$ alors*

$$\mathrm{lr}(\mathfrak{t}) \pmod{\gamma - 1} = \mathrm{lr}(\mathfrak{t}') \pmod{\gamma - 1}. \tag{4.3}$$

Démonstration. Notons $\Rightarrow$ la relation sur **Mag** induite par la règle

$$\mathfrak{p}_{/}^{(\gamma)} \to \mathfrak{p}_{\backslash}^{(\gamma)}.$$

Soit $\mathfrak{t}$ un arbre binaire. Il se décompose ainsi

$$\mathfrak{t} = \mathfrak{p}_{/}^{(\mathrm{lr}(\mathfrak{t}))} \circ \left[\mathfrak{l}, \mathfrak{t}_1, \ldots, \mathfrak{t}_{\mathrm{lr}(\mathfrak{t})}\right],$$

où les $\mathfrak{t}_i$ sont des arbres binaires. Si $\mathfrak{t} \Rightarrow \mathfrak{t}'$ alors nous sommes dans l'un de ces deux cas :

1. La réécriture se fait dans un des arbres $\mathfrak{t}_i$, ce qui signifie qu'il existe un arbre $\mathfrak{t}'_i$ tel que

$$\mathfrak{t}' = \mathfrak{p}_{/}^{(\mathrm{lr}(\mathfrak{t}))} \circ \left[\mathfrak{l}, \mathfrak{t}_1, \ldots, \mathfrak{t}'_i, \ldots, \mathfrak{t}_{\mathrm{lr}(\mathfrak{t})}\right].$$

   Donc $\mathrm{lr}(\mathfrak{t}') = \mathrm{lr}(\mathfrak{t})$.

2. La réécriture implique un nœud de la principale branche gauche de $\mathfrak{t}$. Ce qui signifie qu'il existe un entier i tel que

$$\mathfrak{t}' = \mathfrak{p}_{/}^{(\mathrm{lr}(\mathfrak{t})-\gamma-1)} \circ \left[\mathfrak{l}, \mathfrak{t}_1, \ldots, \mathfrak{p}_{\backslash}^{(\gamma-1)} \circ \left[\mathfrak{t}_i, \ldots, \mathfrak{t}_{i+\gamma-1}\right], \ldots, \mathfrak{t}_{\mathrm{lr}(\mathfrak{t})}\right].$$

   Donc $\mathrm{lr}(\mathfrak{t}') = \mathrm{lr}(\mathfrak{t}) - \gamma - 1 = \mathrm{lr}(\mathfrak{t}) \pmod{\gamma - 1}$.

Les deux cas impliquent (4.3). Nous concluons la démonstration par le fait que $\equiv^{(\gamma)}$ est la clôture réflexive, symétrique et transitive de $\Rightarrow$. □

Grâce à ces deux lemmes nous pouvons caractériser tous les morphismes d'opérades agissant sur les éléments de **CAs**.

Proposition 4.2.3. *Soient $\gamma \geq 2$ et $\gamma' \geq 1$ deux entiers. Il existe un morphisme (surjectif) d'opérades*

$$\varphi : \mathbf{CAs}^{(\gamma')} \to \mathbf{CAs}^{(\gamma)}$$

si et seulement si $\gamma - 1 \mid \gamma' - 1$.

Démonstration. D'après le lemme 4.2.1, il nous suffit de montrer que

$$\mathfrak{p}_{/}^{(\gamma')} \equiv^{(\gamma)} \mathfrak{p}_{\backslash}^{(\gamma')} \tag{4.4}$$

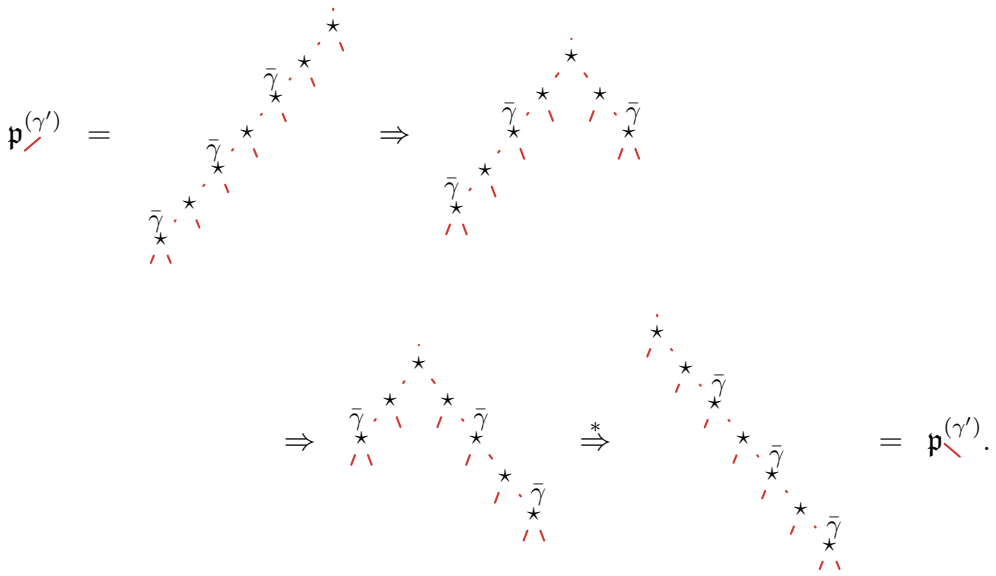
si et seulement si $\gamma - 1 \mid \gamma' - 1$. Si (4.4) alors d'après le lemme 4.2.2

$$\mathrm{lr}\left(\mathfrak{p}_{/}^{(\gamma')}\right) = \mathrm{lr}\left(\mathfrak{p}_{\backslash}^{(\gamma')}\right) \pmod{\gamma - 1}.$$

Donc $\gamma - 1 \mid \gamma' - 1$.

Réciproquement, si $\gamma - 1 \mid \gamma' - 1$ alors en $\frac{\gamma'-1}{\gamma-1}$ pas de réécriture nous

obtenons



Donc $\mathfrak{p}^{(\gamma')} \equiv^{(\gamma)} \mathfrak{p}^{(\gamma')}$. Ce qui conclut la démonstration. □

Cette caractérisation des morphismes d'opérades peignes met en évidence une structure de treillis sur les éléments de **CAs**. Afin de la formaliser, nous définissons $\wedge_{\mathrm{d}}$ et $\vee_{\mathrm{d}}$ comme étant

$$\mathbf{CAs}^{(\gamma)} \wedge_{\mathrm{d}} \mathbf{CAs}^{(\gamma')} := \mathbf{CAs}^{(\mathrm{pgcd}(\bar{\gamma},\bar{\gamma'}))}$$

et

$$\mathbf{CAs}^{(\gamma)} \vee_{\mathrm{d}} \mathbf{CAs}^{(\gamma')} := \mathbf{CAs}^{(\mathrm{ppcm}(\bar{\gamma},\bar{\gamma'}))}$$

pour tous les entiers γ et γ'. On note $\preceq_{\mathrm{d}}$ la relation définie par

$$\mathbf{CAs}^{(\gamma)} \preceq_{\mathrm{d}} \mathbf{CAs}^{(\gamma')}$$

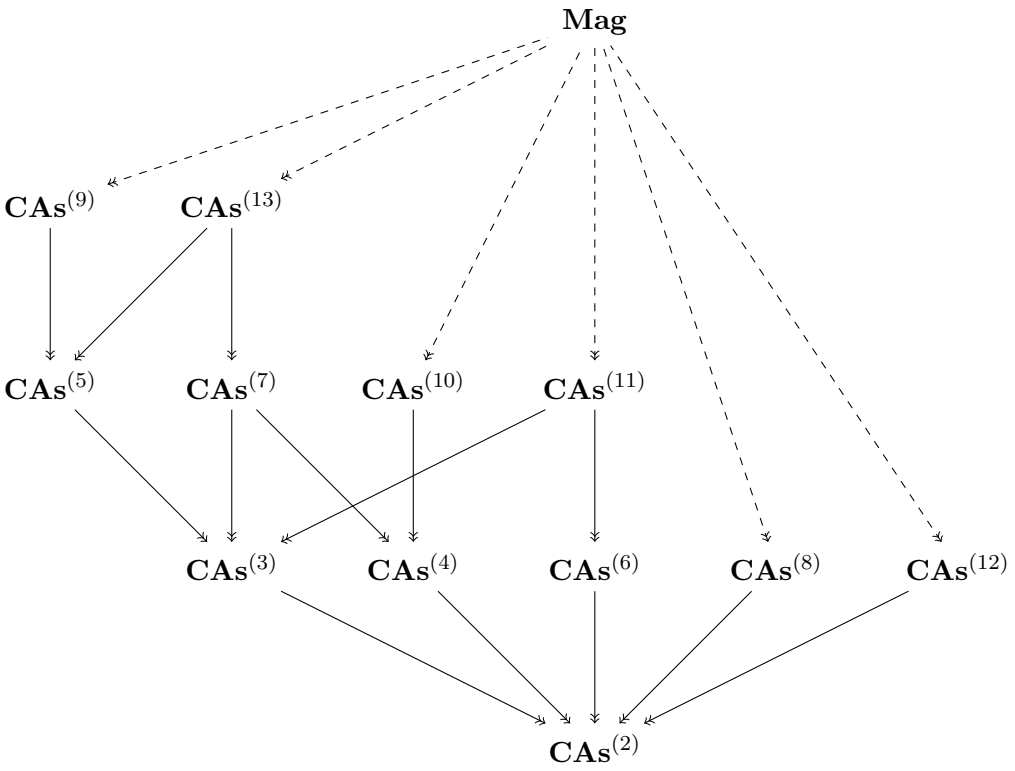
si et seulement s'il existe un morphisme de $\mathbf{CAs}^{(\gamma)}$ à $\mathbf{CAs}^{(\gamma')}$. La proposition 4.2.3 implique le théorème suivant.

Théorème 4.2.4. *Le quadruplet*

$$(\mathbf{CAs}, \preceq_{\mathrm{d}}, \wedge_{\mathrm{d}}, \vee_{\mathrm{d}})$$

est un treillis et il est isomorphe au treillis de la division $(\mathbb{N}, |, pgcd, ppcm)$.

Nous représentons une partie de ce treillis sur les opérades peignes dans le graphique suivant



(4.5)

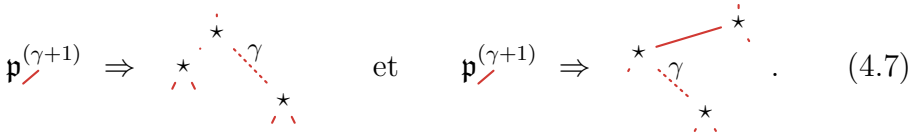
Son élément minimal est $\mathbf{As} = \mathbf{CAs}^{(2)}$ et son élément maximal est $\mathbf{Mag} \cong \mathbf{CAs}^{(1)}$. Algébriquement, cela signifie que toutes les opérades peignes se projettent sur **As** et qu'elles sont toutes des quotients de **Mag**.

4.2.2 Complétions des opérades peignes

Nous cherchons des présentations finies et convergentes des opérades peignes. Par définition, l'opérade $\mathbf{CAs}^{(\gamma)}$ est le quotient de **Mag** par la congruence engendrée par la règle de réécriture

$$\mathfrak{p}^{(\gamma)} \quad \to \quad \mathfrak{p}^{(\gamma)} \,. \tag{4.6}$$

Cependant, la relation de réécriture $\Rightarrow$ induite par $\to$ n'est pas confluente pour $\gamma \geq 3$. En effet, on peut remarquer que



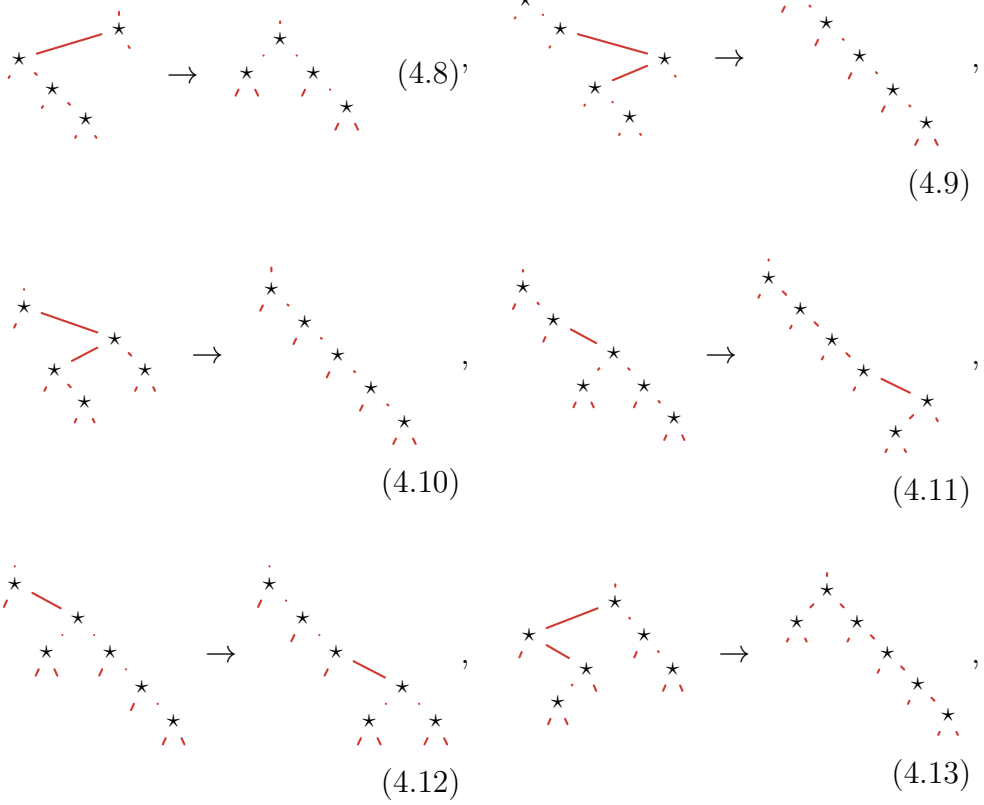
(4.7)

Or ces deux arbres à droite de $\Rightarrow$ sont en formes normales.

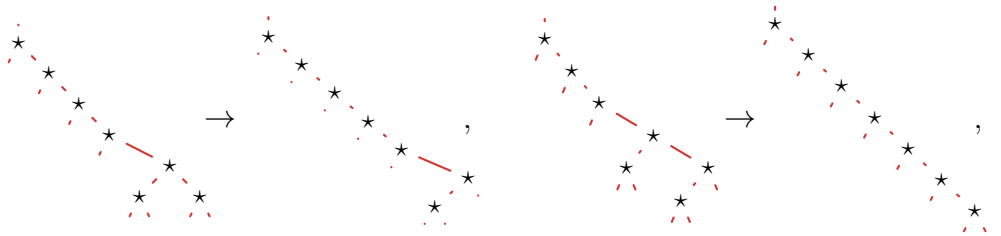
Afin de la rendre confluente, nous allons appliquer le classique algorithme de Buchberger [DK10, Section 3.7]. Nous allons d'abord nous restreindre au cas $\gamma = 3$.

L'opérade peigne de taille 3

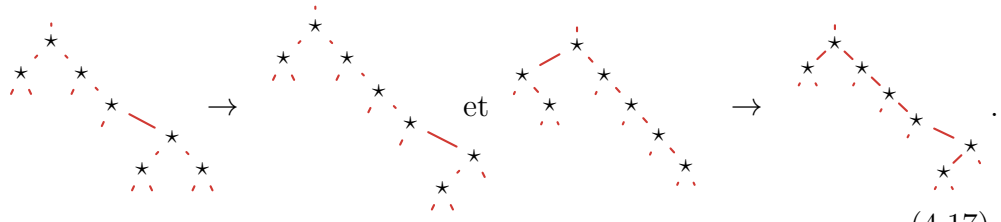
Lorsqu'on applique l'algorithme de Buchberger sur les arbres binaires dont le degré est compris entre 4 et 7, nous obtenons les nouvelles règles de réécriture



(4.8)' (4.9) (4.10) (4.11) (4.12) (4.13)



(4.14) (4.15)



(4.16) (4.17)

En réécrivant ces règles sous la forme de parcours préfixes, nous obtenons

$$22020200 \to 220020200,\ 20202202000 \to 20202020200,$$
$$20220200200 \to 20202020200,\ 2020220020200 \to 2020202022000,$$
$$2022002020200 \to 2020202200200,\ 2202200020200 \to 2200202020200,$$
$$202020202200200 \to 202020202022000,\ 202020220022000 \to 202020202020200,$$
$$220020202200200 \to 220020202022000 \text{ et } 220200202020200 \to 220020202022000.$$

Le théorème suivant exhibe une présentation finie et convergente de l'opérade peigne de taille 3.

Théorème 4.2.5. *L'ensemble* $\to$ *des règles de réécriture* (4.6), (4.8)—(4.17) *forme une présentation finie et convergente de* $\mathbf{CAs}^{(3)}$.

Démonstration. Montrons que la relation de réécriture $\Rightarrow$ induite par $\to$ est convergente. Premièrement, pour tout $\mathfrak{t} \to \mathfrak{t}'$ on a $\mathfrak{t} > \mathfrak{t}'$. Donc d'après le lemme 4.1.1, $\Rightarrow$ est terminante.

Le plus grand degré des arbres apparaissant dans les règles de $\to$ est de 7. Si cette présentation n'est pas une confluente alors il existe des arbres pouvant se réécrire grâce à $\Rightarrow$ en deux formes normales différentes. Ces arbres doivent donc contenir deux membres gauches des règles $\to$. Les plus petits de ces arbres (en terme de degré) sont donc de degré au plus $2 \times 7 = 14$. Or cela n'est pas le cas car nous avons vérifié à l'ordinateur que $\Rightarrow$ est une présentation confluente de $\mathbf{CAs}^{(3)}(n)$ pour $n \leq 14$. Ceci conclut la démonstration. □

La règle de réécriture du théorème 4.2.5 a, arité par arité, les cardinaux suivant

$$0, 0, 0, 1, 1, 2, 3, 4, 0, 0, \dots \ . \tag{4.18}$$

Elle a également des conséquences énumératives comme celle de la proposition suivante.

Proposition 4.2.6. *La série de Hilbert* $\mathcal{H}_{\mathbf{CAs}^{(3)}}(t)$ *de* $\mathbf{CAs}^{(3)}$ *est*

$$\frac{t}{(1-t)^2}\left(1 - t + t^2 + t^3 + 2t^4 + 2t^5 - 7t^7 - 2t^8 + t^9 + 2t^{10} + t^{11}\right). \tag{4.19}$$

Démonstration. Étant donné un entier $n \geq 1$, le cardinal de $\mathbf{CAs}^{(3)}(n)$ est le nombre d'arbres qui évitent les membres gauches de la règle $\rightarrow$.

Dans [Gir18b] (voir aussi [Row10, KP15]) Giraudo propose une méthode pour calculer la série génératrice des arbres évitant un ensemble donné d'arbres. En appliquant cette méthode à notre cas, nous obtenons la série (4.19). □

Pour $n \leq 10$, les dimensions de $\mathbf{CAs}^{(3)}(n)$ sont données par la suite

$$1, 1, 2, 4, 8, 14, 20, 19, 16, 14, 14, 15, 16, 17 \tag{4.20}$$

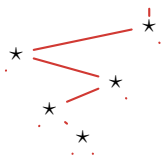
et pour tout $n \geq 11$, le développement de Taylor de la série (4.19) montre que

$$\left|\mathbf{CAs}^{(3)}(n)\right| = n + 3. \tag{4.21}$$

Nous pouvons également calculer les formes normales pour la relation de réécriture induite par la présentation fournie dans le théorème 4.2.5. Nous définissons récursivement pour $d \geq 0$ l'arbre binaire $\mathfrak{z}_d$

$$\mathfrak{z}_d := \begin{cases} | & \text{si } d = 0, \\ \mathfrak{z}_{d-1} \circ_{\lfloor \frac{d-1}{2} \rfloor + 1} \star & \text{sinon.} \end{cases}$$

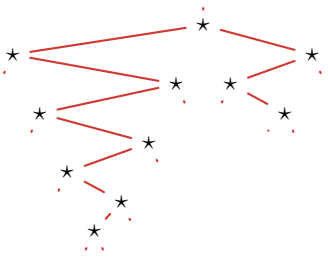
Par exemple,

$\mathfrak{z}_4 =$  et $\mathfrak{z}_5 =$  .

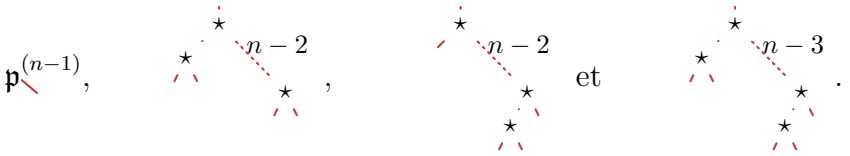
Les formes normales se répartissent en deux familles. La première est l'ensemble des $n-1$ arbres de la forme

$$\mathfrak{z}_d \circ_{d+1} \mathfrak{z}_{n-1-d}, \tag{4.22}$$

pour $d \in [1, n-1]$. Par exemple, pour $n = 12$,

$z_8 \circ_9 z_3 =$ 

est un arbre de la première famille. La deuxième famille contient les quatre arbres



(4.23)

Nous pouvons en effet vérifier que ces $n+3$ arbres évitent les membres gauches des règles de réécriture données dans le théoreme 4.2.5. Nous avons de plus la propriété absorbante suivante.

Proposition 4.2.7. *Si un des arbres* $\mathfrak{t}$ *ou* $\mathfrak{t}'$ *appartient à la famille* (4.23) *et que* $i \in [1, |\mathfrak{t}|]$ *alors la forme normale de* $\mathfrak{t} \circ_i \mathfrak{t}'$ *est un arbre de la famille* (4.23).

Démonstration. Aucun des arbres de la forme (4.22) ne contient en sous-arbre un membre gauche ou droit d'une des règles de réécriture (4.6). Cet ensemble de règle engendre la relation de congruence $\equiv^{(3)}$, nous en déduisons que ces arbres sont seuls dans leurs classes d'équivalence. De plus, les arbres de la forme (4.23) contiennent le membre droit de la règle (4.6) donc la composition d'un de ces arbres avec un autre arbre ne peut appartenir à la famille (4.22). □

La proposition 4.2.7 nous dit que la famille d'arbres (4.23) est absorbante pour les compositions partielles.

L'exploration informatique nous permet de conjecturer la table des compositions des formes normales de $\mathbf{CAs}^{(3)}$. Cependant, nous ne trouvons pas de description simple de celle-ci. Par exemple, la composition partielle $\mathfrak{t} \circ_i \mathfrak{t}'$ où $\mathfrak{t}$ est un arbre de la forme (4.22) et $\mathfrak{t}'$ un arbre de la forme (4.23), peut être décrite par 36 cas. Ces cas dépendent des valeurs des paramètres associés à $\mathfrak{t}$, $\mathfrak{t}'$ et i.

Les autres opérades peignes

Nous avons lancé le même algorithme de complétion pour les opérades $\mathbf{CAs}^{(\gamma)}$ lorsque $\gamma \in [1, 9]$. Nous exposons les résultats obtenus dans l'annexe C. La table 4.37 nous montre le nombre de règles de réécriture qu'il faut ajouter, arité par arité, afin d'obtenir une présentation confluente.

Nous avons conjecturé de ces explorations informatiques que l'algorithme ne donnerait pas de présentation finie convergente de $\mathbf{CAs}^{(\gamma)}$, quand $\gamma \geq 4$. Nous avons observé que pour $\mathbf{CAs}^{(4)}$ de nouvelles règles de réécriture continuent d'apparaître à l'arité 42. De plus, le nombre total de règles à ajouter jusqu'à cette arité est de 3149. Le tableau (4.37) est incomplet car notre programme était trop lent pour pousser plus loin les explorations pour $\mathbf{CAs}^{(5)}$, $\mathbf{CAs}^{(6)}$, $\mathbf{CAs}^{(7)}$ et $\mathbf{CAs}^{(8)}$.

Cependant, l'algorithme de complétion dépend de l'ordre que l'on choisit sur les arbres. Dans le but de savoir si l'algorithme de complétion trouverait une présentation finie et convergente pour un ordre différent, nous avons lancé l'algorithme avec *retour sur trace* (*backtracking* en anglais) suivant : pour chaque nouvelle règle

$$(\mathfrak{t}_1, \mathfrak{t}_2)$$

que l'algorithme de complétion propose, nous tentons récursivement de trouver une complétion finie en ajoutant la règle

$$\mathfrak{t}_1 \to \mathfrak{t}_2$$

ou bien la règle

$$\mathfrak{t}_2 \to \mathfrak{t}_1.$$

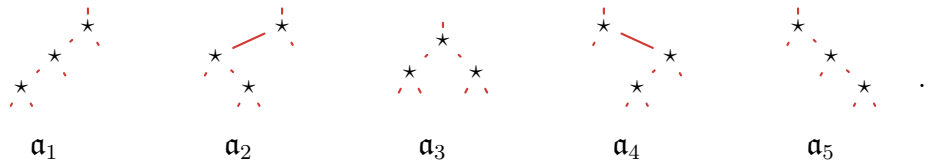
Si à un moment donné la relation de réécriture $\Rightarrow$ induite par $\to$ boucle (c'est-à-dire si $\overset{*}{\Rightarrow}$ n'est pas antisymétrique) alors nous la rejetons. Nous n'avons pas trouvé de présentation finie et convergente pour $\mathbf{CAs}^{(4)}$, $\mathbf{CAs}^{(5)}$ et $\mathbf{CAs}^{(6)}$ jusqu'à l'arité 12.

Nous conjecturons qu'il n'existe pas de présentation finie et convergente de $\mathbf{CAs}^{(\gamma)}$ quand $\gamma \geq 4$ et où les membres gauches et droits des règles de réécriture appartiennent à $\mathbf{Mag}$.

Grâce aux complétions partielles que nous avons présentées dans la table (4.37) de l'annexe C, nous avons pu calculer les premiers coefficients des séries de Hilbert des opérades $\mathbf{CAs}^{(\gamma)}$, pour $1 \leq \gamma \leq 9$. Nous présentons ces résultats dans l'annexe D.

4.3 Quotients cubiques

Nous allons dans cette partie explorer tous les quotients de **Mag** obtenus en confondant deux arbres de degré 3. Nous notons $\mathfrak{a}_i$ le i^e arbre de degré 3 pour l'ordre lexicographique. Ainsi,



Nous notons $\mathbf{Mag}^{\{i,j\}}$ l'opérade quotient $\mathbf{Mag}/_{\equiv}$, où $\equiv$ est la congruence d'opérade engendrée par $\mathfrak{a}_i \equiv \mathfrak{a}_j$. Nous avons déjà étudié dans la partie 4.2.2 l'opérade $\mathbf{Mag}^{\{1,5\}} = \mathbf{CAs}^{(3)}$. Certains de ces quotients sont anti-isomorphes.

4.3.1 Classes anti-isomorphes

L'application $\psi : \mathbf{Mag} \to \mathbf{Mag}$ qui envoie un arbre $\mathfrak{t}$ sur l'arbre obtenu en échangeant récursivement les fils gauche et droit de chaque nœud de $\mathfrak{t}$ est un anti-isomorphisme de **Mag**. Pour cette raison, les $\binom{5}{2} = 10$ quotients $\mathbf{Mag}^{\{i,j\}}$ se répartissent en six classes d'équivalence

$$\left\{\mathbf{Mag}^{\{1,2\}}, \mathbf{Mag}^{\{4,5\}}\right\}, \left\{\mathbf{Mag}^{\{1,3\}}, \mathbf{Mag}^{\{3,5\}}\right\}, \left\{\mathbf{Mag}^{\{1,4\}}, \mathbf{Mag}^{\{2,5\}}\right\}, \left\{\mathbf{CAs}^{(3)}\right\}, \left\{\mathbf{Mag}^{\{2,3\}}, \mathbf{Mag}^{\{3,4\}}\right\}, \left\{\mathbf{Mag}^{\{2,4\}}\right\} \quad (4.24)$$

d'opérades anti-isomorphes.

Étant donnée une opérade $\mathcal{O}$ dotée des compositions partielles $\circ_i$, nous définissons les compositions partielles $\bar{\circ}_i$ par

$$x \,\bar{\circ}_i\, y := x \circ_{|x|-i+1} y$$

pour tout $x, y \in \mathcal{O}$ et $i \in [1, |x|]$. Le lecteur peut facilement vérifier les assertions du lemme suivant.

Lemme 4.3.1. *Si $\mathcal{O}_1$ et $\mathcal{O}_2$ sont deux opérades anti-isomorphes et que ϕ est un anti-isomorphisme de $\mathcal{O}_1$ dans $\mathcal{O}_2$ alors*

1. $\mathcal{H}_{\mathcal{O}_1}(t) = \mathcal{H}_{\mathcal{O}_2}(t)$;
2. *si $\to^{(1)}$ est une présentation convergente de $\mathcal{O}_1$ alors l'ensemble des règles de réécriture $\to^{(2)}$ défini par*

$$\psi(x) \to^{(2)} \psi(y),$$

pour tout $x \to^{(1)} y$, est une présentation convergente de $\mathcal{O}_2$;

3. *si* $(\mathcal{O}, \circ_i)$ *est une réalisation combinatoire de* $\mathcal{O}_1$ *alors* $(\mathcal{O}, \bar{\circ}_i)$ *est une réalisation combinatoire de* $\mathcal{O}_2$.

4.3.2 Réalisations combinatoires

Quatre des six classes d'équivalence (4.24) des quotients $\mathbf{Mag}^{\{i,j\}}$ peuvent être réalisées par des opérades sur les compositions d'entiers. Nous allons les étudier une par une.

Les opérades $\mathbf{Mag}^{\{1,2\}}$ et $\mathbf{Mag}^{\{4,5\}}$

Le lecteur peut vérifier grâce à l'algorithme de Buchberger que la règle de réécriture $\mathfrak{a}_2 \to \mathfrak{a}_1$ est une présentation convergente de $\mathbf{Mag}^{\{1,2\}}$. Les opérades $\mathbf{Mag}^{\{1,2\}}$ et $\mathbf{Mag}^{\{4,5\}}$ sont anti-isomorphes, donc d'après le lemme 4.3.1 la règle $\mathfrak{a}_4 \to \mathfrak{a}_5$ est une présentation convergente de $\mathbf{Mag}^{\{4,5\}}$. En utilisant la même méthode que celle de la proposition 4.2.6, nous obtenons le résultat suivant.

Théorème 4.3.2. *Les séries de Hilbert de* $\mathbf{Mag}^{\{1,2\}}$ *et* $\mathbf{Mag}^{\{4,5\}}$ *sont*

$$\mathcal{H}_{\mathbf{Mag}^{\{1,2\}}}(t) = \mathcal{H}_{\mathbf{Mag}^{\{4,5\}}}(t) = t\frac{1-t}{1-2t}. \tag{4.25}$$

Nous obtenons grâce à un développement de Taylor de la série (4.25) la formule close suivante.

Proposition 4.3.3. *Pour tout* $n \geq 2$,

$$\left|\mathbf{Mag}^{\{1,2\}}(n)\right| = \left|\mathbf{Mag}^{\{4,5\}}(n)\right| = 2^{n-2}. \tag{4.26}$$

Nombreux sont les objets combinatoires énumérés par les puissances de 2. Nous avons choisi les compositions d'entiers comme support pour notre réalisation de $\mathbf{Mag}^{\{1,2\}}$. On rappelle qu'une ***composition d'entier*** est une suite finie d'entiers. Si

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)$$

est une composition d'entier alors nous notons $s_{i,j}(\boldsymbol{\lambda})$ le nombre

$$1 + \sum_{i \leq k \leq j} \boldsymbol{\lambda}_k.$$

L'arité de $\boldsymbol{\lambda}$ est le nombre $s_{1,p}(\boldsymbol{\lambda})$. La composition vide ϵ est l'unique objet d'arité 1. Nous notons $\mathcal{C}$ l'ensemble des compositions d'entiers.

Étant donné un entier $i \geq 1$, nous définissons l'opération binaire

$$\circ_i^{(1,2)} : \mathcal{C}(n) \times \mathcal{C}(m) \to \mathcal{C}(n+m-1)$$

pour toutes les compositions d'entiers

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \text{ et } \boldsymbol{\mu} := (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q),$$

dont les arités respectives sont $n \geq i$ et $m \geq 1$, par

$$\boldsymbol{\lambda} \circ_i^{(1,2)} \boldsymbol{\mu} := \begin{cases} (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q) & \text{si } i = n, \\ (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k, \boldsymbol{\lambda}_{k+1} + m - 1, \boldsymbol{\lambda}_{k+2}, \ldots, \boldsymbol{\lambda}_p) & \text{sinon,} \end{cases}$$

où $k \geq 0$ est l'entier tel que $s_{1,k}(\boldsymbol{\lambda}) \leq i < s_{1,k+1}(\boldsymbol{\lambda})$.

Théorème 4.3.4. *L'opérade*

$$\left(\mathcal{C}, \circ_i^{(1,2)}\right)$$

est une réalisation combinatoire de $\mathbf{Mag}^{\{1,2\}}$.

Démonstration. Il nous suffit de montrer que les opérades $\left(\mathcal{C}, \circ_i^{(1,2)}\right)$ et $\mathbf{Mag}^{\{1,2\}}$ sont isomorphes. L'ensemble des formes normales d'arité n pour la relation de réécriture $\Rightarrow$ induite par la règle $\mathfrak{a}_2 \to \mathfrak{a}_1$ est

$$\left\{\mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)} : (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \in \mathcal{C}(n)\right\}, \tag{4.27}$$

où

$$\mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)} := \mathfrak{p}_{\diagdown}^{(p)} \circ \left[\mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_1 - 1)}, \ldots, \mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_p - 1)}, |\right].$$

L'application $\phi : \mathbf{Mag}^{\{1,2\}}(n) \to \mathcal{C}(n)$ définie par

$$\phi\left(\mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)}\right) := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \tag{4.28}$$

est donc une bijection. Il nous reste à montrer que ϕ est un morphisme d'opérades. Soit

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \text{ et } \boldsymbol{\mu} := (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q)$$

deux compositions d'entiers dont les arités respectives sont n et m. Si

$$\mathfrak{t} := \mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)} \in \mathbf{Mag}^{\{1,2\}}(n) \text{ et } \mathfrak{t}' := \mathfrak{p}_{\diagup}^{(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q)} \in \mathbf{Mag}^{\{1,2\}}(m)$$

alors

$$\mathfrak{t} \circ_n \mathfrak{t}' = \mathfrak{p}_{\diagup}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q)}$$

donc

$$\phi\left(\mathfrak{t} \circ_n \mathfrak{t}'\right) = \phi\left(\mathfrak{t}\right) \circ_n^{(1,2)} \phi\left(\mathfrak{t}'\right).$$

Si $i \in [1, n-1]$ et k est un entier tel que $s_{1,k}(\boldsymbol{\lambda}) \leq i < s_{1,k+1}(\boldsymbol{\lambda})$ alors l'arbre $\mathfrak{t} \circ_i \mathfrak{t}'$ est l'arbre

$$\mathfrak{p}_{\diagdown}^{(p)} \circ \left[\mathfrak{p}_{/}^{(\boldsymbol{\lambda}_1-1)}, \ldots, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_k-1)}, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_{k+1}-1)} \circ_{i+1-s_{1,k}(\boldsymbol{\lambda})} \mathfrak{t}', \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_{k+2}-1)}, \ldots, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_p-1)}, \big|\right]. \tag{4.29}$$

L'arbre (4.29) se réécrit par $\Rightarrow$ en

$$\mathfrak{p}_{\diagdown}^{(p)} \circ \left[\mathfrak{p}_{/}^{(\boldsymbol{\lambda}_1-1)}, \ldots, \left(\mathfrak{p}_{/}^{(\boldsymbol{\lambda}_{k+1}-1)} \circ_{i+1-s_{1,k}(\boldsymbol{\lambda})} \mathfrak{p}_{/}^{(\boldsymbol{\mu}_1-1)}\right) \circ_{i+1-s_{1,k}(\boldsymbol{\lambda})+\boldsymbol{\mu}_1} \mathfrak{p}_{/}^{(\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_q)}, \ldots, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_p-1)}, \big|\right].$$

Ce dernier se réécrit en $\boldsymbol{\mu}_1 - 1$ pas par $\Rightarrow$ en

$$\mathfrak{p}_{\diagdown}^{(p)} \circ \left[\mathfrak{p}_{/}^{(\boldsymbol{\lambda}_1-1)}, \ldots, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_k-1)}, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_{k+1}-1+\boldsymbol{\mu}_1)} \circ_{i+1-s_{1,k}(\boldsymbol{\lambda})+\boldsymbol{\mu}_1} \mathfrak{p}_{/}^{(\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_q)}, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_{k+2}-1)}, \ldots, \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_p-1)}, \big|\right]. \tag{4.30}$$

En itérant $q-1$ fois les étapes de réécriture de (4.29) à (4.30), nous obtenons

$$\mathfrak{t} \circ_i \mathfrak{t}' \Rightarrow \mathfrak{p}_{/}^{(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k,\boldsymbol{\lambda}_{k+1}+m-1,\boldsymbol{\lambda}_{k+2},\ldots,\boldsymbol{\lambda}_p)}.$$

Nous avons donc montré que

$$\phi(\mathfrak{t} \circ_i \mathfrak{t}') \;=\; \phi(\mathfrak{t}) \circ_i^{(1,2)} \phi(\mathfrak{t}').$$

Ce qui démontre que l'application ϕ est bien un morphisme d'opérades et permet ainsi de conclure la démonstration. □

Grâce au lemme 4.3.1, nous savons que $\left(\mathcal{C}, \bar{\circ}_i^{(1,2)}\right)$ est une réalisation combinatoire de $\mathbf{Mag}^{\{4,5\}}$.

Les opérades $\mathbf{Mag}^{\{1,3\}}$ et $\mathbf{Mag}^{\{3,5\}}$

Le lecteur peut vérifier que la règle de réécriture $\mathfrak{a}_3 \to \mathfrak{a}_1$ est une présentation convergente de l'opérade $\mathbf{Mag}^{\{1,3\}}$. D'après le lemme 4.3.1, la règle $\mathfrak{a}_3 \to \mathfrak{a}_5$ est une présentation convergente de l'opérade $\mathbf{Mag}^{\{3,5\}}$. En utilisant la même méthode que celle de la proposition 4.2.6, nous obtenons que les séries de Hilbert de $\mathbf{Mag}^{\{1,3\}}$ et $\mathbf{Mag}^{\{3,5\}}$ sont égales à (4.25). Les coefficients

$$\left|\mathbf{Mag}^{\{1,3\}}(n)\right| \;\text{ et }\; \left|\mathbf{Mag}^{\{3,5\}}(n)\right|$$

sont donc égaux à (4.3.3) lorsque $n \geq 2$.

Comme pour notre étude de l'opérade $\mathbf{Mag}^{\{1,2\}}$ dans la partie 4.3.2, nous avons choisi une réalisation basée sur les compositions d'entiers. Étant donné un entier $i \geq 1$, nous définissons l'opération binaire

$$\circ_i^{(1,3)} : \mathcal{C}(n) \times \mathcal{C}(m) \to \mathcal{C}(n+m-1)$$

pour toutes les compositions d'entiers

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \text{ et } \boldsymbol{\mu} := (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q),$$

dont les arités respectives sont $n \geq i$ et $m \geq 1$, par

$$\boldsymbol{\lambda} \circ_i^{(1,3)} \boldsymbol{\mu} := \begin{cases} (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{i-1}, \boldsymbol{\mu}_1 + s_{i,p}(\boldsymbol{\lambda}), \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_q) & \text{si } i \leq p+1, \\ (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k + m - 1, \boldsymbol{\lambda}_{k+1}, \ldots, \boldsymbol{\lambda}_p) & \text{sinon,} \end{cases}$$

où k est l'entier tel que $k + 1 + s_{k+1,p}(\boldsymbol{\lambda}) \leq i < k + s_{k,p}(\boldsymbol{\lambda})$.

Théorème 4.3.5. *L'opérade*

$$\left(\mathcal{C}, \circ_i^{(1,3)}\right)$$

est une réalisation combinatoire de l'opérade $\mathbf{Mag}^{\{1,3\}}$.

Démonstration. La démonstration étant similaire à celle du théorème 4.3.4, nous n'en donnerons que les grandes lignes.

Il est suffisant de montrer que les opérades $\left(\mathcal{C}, \circ_i^{(1,3)}\right)$ et $\mathbf{Mag}^{\{1,3\}}$ sont isomorphes. L'ensemble des formes normales d'arité n pour la relation $\Rightarrow$ induite par la règle $\mathfrak{a}_3 \to \mathfrak{a}_1$ est

$$\left\{ \mathfrak{c}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)} : (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \in \mathcal{C}(n) \right\},$$

où

$$\mathfrak{c}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)} := \mathfrak{p}^{(\boldsymbol{\lambda}_1)} \circ_2 \left(\mathfrak{p}^{(\boldsymbol{\lambda}_2)} \circ_2 \left(\cdots \left(\mathfrak{p}^{(\boldsymbol{\lambda}_{p-1})} \circ_2 \mathfrak{p}^{(\boldsymbol{\lambda}_p)} \right) \cdots \right) \right).$$

Nous pouvons conclure la démonstration en montrons que l'application

$$\phi : \mathbf{Mag}^{\{1,3\}}(n) \to \mathcal{C}(n)$$

définie par

$$\phi\left(\mathfrak{c}^{(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)} \right) := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)$$

est un isomorphisme de l'opérade $\mathbf{Mag}^{\{1,3\}}$ dans l'opérade $\left(\mathcal{C}, \circ_i^{(1,3)}\right)$. □

D'après le lemme 4.3.1, nous savons que l'opérade $\left(\mathcal{C}, \bar{\circ}_i^{(1,3)}\right)$ est une réalisation combinatoire de l'opérade $\mathbf{Mag}^{\{3,5\}}$.

Les opérades $\mathbf{Mag}^{\{1,4\}}$ et $\mathbf{Mag}^{\{2,5\}}$

Le lecteur peut vérifier que la règle $\mathfrak{a}_4 \to \mathfrak{a}_1$ est une présentation convergente de l'opérade $\mathbf{Mag}^{\{1,4\}}$. D'après le lemme 4.3.1, la règle $\mathfrak{a}_2 \to \mathfrak{a}_5$ est une présentation convergente de l'opérade $\mathbf{Mag}^{\{2,5\}}$. En utilisant la même méthode que celle de la proposition 4.2.6, nous obtenons que les séries de Hilbert de $\mathbf{Mag}^{\{1,4\}}$ et $\mathbf{Mag}^{\{2,5\}}$ sont égales à (4.25). Les coefficients

$$\left|\mathbf{Mag}^{\{1,4\}}(n)\right| \text{ et } \left|\mathbf{Mag}^{\{2,5\}}(n)\right|$$

sont donc égaux à (4.3.3) lorsque $n \geq 2$.

Comme pour notre étude des opérades $\mathbf{Mag}^{\{1,2\}}$ et $\mathbf{Mag}^{\{1,3\}}$ de la partie 4.3.2, nous avons choisi une réalisation basée sur les compositions d'entiers. Étant donné un entier $i \geq 1$, nous définissons l'opération binaire

$$\circ_i^{(2,5)} : \mathcal{C}(n) \times \mathcal{C}(m) \to \mathcal{C}(n+m-1)$$

pour toutes les compositions d'entiers

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \text{ et } \boldsymbol{\mu} := (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q),$$

dont les arités respectives sont $n \geq i$ et $m \geq 1$, par

$$\boldsymbol{\lambda} \circ_i^{(2,5)} \boldsymbol{\mu} := \begin{cases} (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{q-1}, \boldsymbol{\mu}_q + \boldsymbol{\lambda}_{k+1}, \ldots, \boldsymbol{\lambda}_p) & \text{si } i = s_{1,k}(\boldsymbol{\lambda}), \\ (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k, i - s_{1,k}(\boldsymbol{\lambda}), \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q, s_{1,k+1}(\boldsymbol{\lambda}) - i, \boldsymbol{\lambda}_{k+2}, \ldots, \boldsymbol{\lambda}_p) & \\ & \text{sinon,} \end{cases}$$

où k est l'entier tel que $s_{1,k}(\boldsymbol{\lambda}) \leq i < s_{1,k+1}(\boldsymbol{\lambda})$.

Théorème 4.3.6. *L'opérade*

$$\left(\mathcal{C}, \circ_i^{(2,5)}\right)$$

est une réalisation combinatoire de l'opérade $\mathbf{Mag}^{\{2,5\}}$.

Démonstration. La démonstration étant semblable à celle du théorème 4.3.4, nous n'en donnerons que les grandes lignes.

Il suffit de montrer que ces deux opérades sont isomorphes. L'ensemble des formes normales d'arité n pour la règle induite par $\mathfrak{a}_2 \to \mathfrak{a}_5$ est (4.27). Il reste donc à montrer que l'application (4.28) est un isomorphisme de l'opérade $\mathbf{Mag}^{\{2,5\}}$ dans l'opérade $\left(\mathcal{C}, \circ_i^{(2,5)}\right)$. □

D'après le lemme 4.3.1, nous savons que l'opérade $\left(\mathcal{C}, \bar{\circ}_i^{(2,5)}\right)$ est une réalisation combinatoire de l'opérade $\mathbf{Mag}^{\{1,4\}}$.

L'opérade $\mathbf{Mag}^{\{2,4\}}$

Le lecteur peut vérifier que les règles $\mathfrak{a}_2 \to \mathfrak{a}_4$ et $\mathfrak{a}_4 \to' \mathfrak{a}_2$ sont toutes les deux des présentations convergentes de l'opérade $\mathbf{Mag}^{\{2,4\}}$. Nous obtenons de même que la série de Hilbert de $\mathbf{Mag}^{\{2,4\}}$ est égale à (4.25). Ainsi les coefficients

$$\left|\mathbf{Mag}^{\{2,4\}}(n)\right|$$

sont donc égaux à (4.3.3) lorsque $n \geq 2$.

Étant donné un entier $i \geq 1$, nous définissons l'opération binaire

$$\circ_i^{(2,4)} : \mathcal{C}(n) \times \mathcal{C}(m) \to \mathcal{C}(n+m-1)$$

pour toutes les compositions d'entiers

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p) \text{ et } \boldsymbol{\mu} := (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_q),$$

dont les arités respectives sont $n \geq i$ et $m \geq 1$, par

$$\boldsymbol{\lambda} \circ_i^{(2,4)} \boldsymbol{\mu} := \begin{cases} (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{q-1}, \boldsymbol{\mu}_q + \boldsymbol{\lambda}_{k+1}, \boldsymbol{\lambda}_{k+2}, \ldots, \boldsymbol{\lambda}_p) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{si } i = s_{1,k}(\boldsymbol{\lambda}), \\ (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k, i - s_{1,k}(\boldsymbol{\lambda}), \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{q-1}, \boldsymbol{\mu}_q + s_{1,k+1}(\boldsymbol{\lambda}) - i, \\ \qquad\qquad\qquad\qquad\qquad\qquad \boldsymbol{\lambda}_{k+2}, \ldots, \boldsymbol{\lambda}_p) \quad \text{sinon}, \end{cases}$$

où k est l'entier tel que $s_{1,k}(\boldsymbol{\lambda}) \leq i < s_{1,k+1}(\boldsymbol{\lambda})$.

Théorème 4.3.7. *Les opérades*

$$\left(\mathcal{C}, \circ_i^{(2,4)}\right) \text{ et } \left(\mathcal{C}, \bar{\circ}_i^{(2,4)}\right)$$

sont des réalisations combinatoires de l'opérade $\mathbf{Mag}^{\{2,4\}}$.

Démonstration. La démonstration est semblable à celle du théorème 4.3.6. □

Opérades non-isomorphes

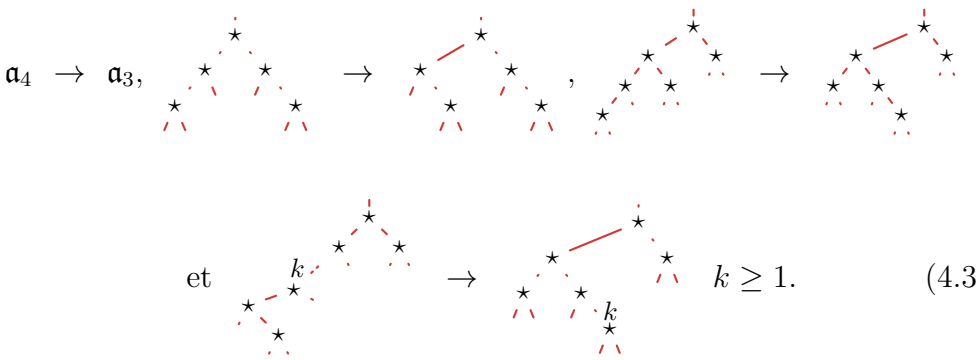
Comme nous venons de le voir, les opérades des quatre classes d'équivalence

$$\left\{\mathbf{Mag}^{\{1,2\}}, \mathbf{Mag}^{\{4,5\}}\right\}, \left\{\mathbf{Mag}^{\{1,3\}}, \mathbf{Mag}^{\{3,5\}}\right\}, \left\{\mathbf{Mag}^{\{1,4\}}, \mathbf{Mag}^{\{2,5\}}\right\} \text{ et } \left\{\mathbf{Mag}^{\{2,4\}}\right\}$$

ont les mêmes séries de Hilbert. Même si elles peuvent être réalisées par le même ensemble de compositions d'entier, elles sont deux à deux non isomorphes et non anti-isomorphes. En effet, tout (anti-)isomorphisme entre deux de ces opérades envoie nécessairement le générateur de l'une sur le générateur de l'autre. De plus, les relations sur ces générateurs sont intrinsèquement différentes de l'une à l'autre. Elles ne peuvent donc pas être (anti-)isomorphes.

4.3.3 Quotients aux présentations possiblement infinies

Nous n'avons pas trouvé de présentation finie et convergente des opérades $\mathbf{Mag}^{\{2,3\}}$ et $\mathbf{Mag}^{\{3,4\}}$. Cependant, grâce à l'exploration informatique nous avons conjecturé que les règles de réécriture



(4.31)

forme une présentation convergente de $\mathbf{Mag}^{\{3,4\}}$. Nous l'avons vérifiée jusqu'à l'arité 40. Si elle est exacte alors les séries de Hilbert des opérades $\mathbf{Mag}^{\{2,3\}}$ et $\mathbf{Mag}^{\{3,4\}}$ sont

$$\mathcal{H}_{\mathbf{Mag}^{\{2,3\}}}(t) = \mathcal{H}_{\mathbf{Mag}^{\{3,4\}}}(t) = \frac{t}{(1-t)^3}\left(1 - 2t + 2t^2 + t^4 - t^6\right). \tag{4.32}$$

Par un développement de Taylor, nous obtenons la suite

$$1, 1, 2, 4, 8, 14, 21, 29, 38, 48 \tag{4.33}$$

pour les premières dimensions de $\mathbf{Mag}^{\{2,3\}}$ et de $\mathbf{Mag}^{\{3,4\}}$ et nous obtenons

$$\left|\mathbf{Mag}^{\{2,3\}}(n)\right| = \left|\mathbf{Mag}^{\{3,4\}}(n)\right| = \frac{n(n+1)}{2} - 7, \tag{4.34}$$

pour $n \geq 5$.

Perspectives

Nous suggérons dans l'ordre des perspectives pour les trois principaux chapitres de ce manuscrit.

Nous proposons deux perspectives pour le chapitre 2, portant sur l'exploration informatique de la théorie des codes. Premièrement, nous nous demandons quels sont les plus petits codes (au sens de la longueur du plus long mot) non commutativement préfixes. On ne sait toujours pas quelle est la longueur du plus long mot de ces codes. Cependant, nous avons montré qu'elle est comprise (bornes incluses) entre 7 et 12. La recherche exhaustive jusqu'à 6 a été effectuée en quelques minutes sur un ordinateur de bureau quadricœur. La recherche exhaustive jusqu'à 7, nous paraît envisageable sur des machines plus conséquentes. En ce qui concerne la borne supérieure, nous pourrions nous restreindre à la recherche de codes non commutativement préfixes dont les mots contiennent au plus deux b.

La deuxième perspective est de savoir s'il existe un code modulaire baïonnette complet contenant un code non commutativement préfixe. Une réponse négative démontrerait la *conjecture du triangle*.

Nous soumettons quatre perspectives pour le chapitre 3, portant sur la combinatoire des prographes. Premièrement, nous avons énuméré les prographes selon la nature de leurs générateurs, leurs nombres de générateurs, leurs nombres d'entrées et leurs nombres de sorties. Nous proposons d'y ajouter leurs hauteurs. La *hauteur* d'un prographe peut être informellement définie comme le plus grand nombre de générateurs traversés par un chemin d'une entrée à une sortie. L'intérêt d'un tel paramètre se manifeste quand, par exemple, les prographes modélisent des circuits électroniques. En effet, dans ce cas la hauteur correspond à la complexité du circuit.

Deuxièmement, nous pensons que les prographes peuvent encoder et modéliser des structures qui ne peuvent pas être capturées par la combinatoire classique des arbres. Afin de mesurer cette expressivité, nous nous demandons à quelles classes (rationnelle, algébrique, holonome, D-algébrique,... [Sta97]) les séries génératrices (3.23) des prographes appartiennent. Nos recherches informatiques, basées sur le paquet GFun du logiciel Maple, nous laissent

penser que ces séries sont pour la plupart non holonomes.

Troisièmement, dans leur article [GHJMM03], les auteurs ont trouvé une formule close pour décrire la croissance asymptotique des coefficients (3.26). Nous proposons comme piste de réflexion de généraliser cette formule aux prographes engendrés par un seul type de générateur.

Et pour finir, notre construction des prographes peut se généraliser aux prographes à générateurs typés. Informellement, chaque entrée et sortie d'un générateur typé a un type donné. Si p_1 et p_2 sont des prographes à générateurs typés alors le prographe $p_1 \circ p_2$ n'est défini que si $\uparrow(p_1) = \downarrow(p_2)$ et que la première entrée de p_2 est du même type que la première sortie de p_1 et ainsi de suite... Nous nous demandons lesquels de nos résultats énumératifs sur les prographes peuvent se généraliser aux prographes à générateurs typés.

Nous exposons deux perspectives pour le chapitre 4 qui porte sur la réécriture dans des quotients de l'opérade **Mag**. Il existe une technique classique en réécriture sur les mots qui consiste à ajouter un nouveau générateur afin d'obtenir une présentation convergente. Nous nous demandons si cette technique appliquée aux opérades peut nous fournir des présentations convergentes de l'opérade $\mathbf{Mag}^{\{2,3\}}$ et des opérades $\mathbf{CAs}^{(\gamma)}$, quand $\gamma \geq 4$.

Nous avons étudié les 10 quotients cubiques de l'opérade **Mag**. Notre dernière perspective propose d'étudier les $\binom{14}{2} = 91$ quotients de **Mag** qui confondent deux arbres de degré 4. Nous avons observé ces quotients à l'ordinateur. Certains ont des séries de Hilbert qui nous semblent, en utilisant le paquet GFUN, être non holonomes.

Annexes

Annexe A

Les 38 codes baïonnettes non commutativement préfixes pour $n = 15$ ont en commun les mots

$$01\ 07\ 0D\ 0E\ 82\ 84\ 86\ B1\ B2. \tag{4.35}$$

Nous les recensons dans la table ci-dessous. Il faut leurs ajouter les éléments de l'ensemble (4.35). Notons que le code $\mathcal{Y}_1$ est le code de Shor.

Nom	Code non commutativement préfixe						
$\mathcal{Y}_1$	00	30	32	34	36	80	B0
$\mathcal{Y}_2$	00	30	32	34	36	80	B3
$\mathcal{Y}_3$	00	30	34	36	3A	80	B0
$\mathcal{Y}_4$	00	30	34	36	3A	80	B3
$\mathcal{Y}_5$	00	31	33	35	37	80	B0
$\mathcal{Y}_6$	00	31	33	35	37	80	B3
$\mathcal{Y}_7$	00	31	35	37	3B	80	B0
$\mathcal{Y}_8$	00	31	35	37	3B	80	B3
$\mathcal{Y}_9$	00	32	34	36	38	80	B0
$\mathcal{Y}_{10}$	00	32	34	36	38	80	B3
$\mathcal{Y}_{11}$	00	33	35	37	39	80	B0
$\mathcal{Y}_{12}$	00	33	35	37	39	80	B3
$\mathcal{Y}_{13}$	00	34	36	38	3A	80	B0
$\mathcal{Y}_{14}$	00	34	36	38	3A	80	B3
$\mathcal{Y}_{15}$	00	35	37	39	3B	80	B0
$\mathcal{Y}_{16}$	00	35	37	39	3B	80	B3
⋮			⋮	⋮			⋮

(4.36)

$\mathcal{Y}_{17}$	10	32	34	36	40	90	C0
$\mathcal{Y}_{18}$	10	34	36	3A	40	90	C0
$\mathcal{Y}_{19}$	20	32	34	36	50	A0	D0
$\mathcal{Y}_{20}$	20	32	34	36	58	A0	D0
$\mathcal{Y}_{21}$	20	34	36	3A	50	A0	D0
$\mathcal{Y}_{22}$	20	34	36	3A	58	A0	D0
$\mathcal{Y}_{23}$	30	32	34	36	60	B0	E0
$\mathcal{Y}_{24}$	30	32	34	36	60	B3	E0
$\mathcal{Y}_{25}$	30	34	36	3A	60	B0	E0
$\mathcal{Y}_{26}$	30	34	36	3A	60	B3	E0
$\mathcal{Y}_{27}$	31	33	35	37	60	B0	E0
$\mathcal{Y}_{28}$	31	33	35	37	60	B3	E0
$\mathcal{Y}_{29}$	31	35	37	3B	60	B0	E0
$\mathcal{Y}_{30}$	31	35	37	3B	60	B3	E0
$\mathcal{Y}_{31}$	32	34	36	38	60	B0	E0
$\mathcal{Y}_{32}$	32	34	36	38	60	B3	E0
$\mathcal{Y}_{33}$	33	35	37	39	60	B0	E0
$\mathcal{Y}_{34}$	33	35	37	39	60	B3	E0
$\mathcal{Y}_{35}$	34	36	38	3A	60	B0	E0
$\mathcal{Y}_{36}$	34	36	38	3A	60	B3	E0
$\mathcal{Y}_{37}$	35	37	39	3B	60	B0	E0
$\mathcal{Y}_{38}$	35	37	39	3B	60	B3	E0

Annexe B

Nous avons découvert les 25 codes ci-dessous pour $n = 16$.

Nom	Code non commutativement préfixe																
$\mathcal{Z}_1$	00	01	02	0B	0C	3B	3C	50	51	52	80	82	84	86	D0	D1	D2
$\mathcal{Z}_2$	00	01	02	0B	0C	3B	3C	50	51	52	81	83	85	87	D0	D1	D2
$\mathcal{Z}_3$	00	01	02	0B	0C	3B	3C	50	51	5A	80	82	84	86	D0	D1	D2
$\mathcal{Z}_4$	00	01	02	0B	0C	3B	3C	50	51	5A	81	83	85	87	D0	D1	D2
$\mathcal{Z}_5$	00	01	0A	0B	0C	3B	3C	50	51	52	80	82	84	86	D0	D1	D2
$\mathcal{Z}_6$	00	01	0A	0B	0C	3B	3C	50	51	52	81	83	85	87	D0	D1	D2
$\mathcal{Z}_7$	00	01	0A	0B	0C	3B	3C	50	51	5A	80	82	84	86	D0	D1	D2
$\mathcal{Z}_8$	00	01	0A	0B	0C	3B	3C	50	51	5A	81	83	85	87	D0	D1	D2
$\mathcal{Z}_9$	00	02	0B	0C	11	3B	3C	50	52	61	83	85	87	91	D0	D2	E1
$\mathcal{Z}_{10}$	00	02	0B	0C	11	3B	3C	50	5A	61	83	85	87	91	D0	D2	E1
$\mathcal{Z}_{11}$	01	02	03	0C	0D	3B	3C	50	51	52	80	82	84	86	D0	D1	D2
$\mathcal{Z}_{12}$	01	02	03	0C	0D	3B	3C	50	51	52	81	83	85	87	D0	D1	D2
$\mathcal{Z}_{13}$	01	02	03	0C	0D	3B	3C	50	51	5A	80	82	84	86	D0	D1	D2
$\mathcal{Z}_{14}$	01	02	03	0C	0D	3B	3C	50	51	5A	81	83	85	87	D0	D1	D2
$\mathcal{Z}_{15}$	01	02	0B	0C	0D	3B	3C	50	51	52	80	82	84	86	D0	D1	D2
$\mathcal{Z}_{16}$	01	02	0B	0C	0D	3B	3C	50	51	52	81	83	85	87	D0	D1	D2
$\mathcal{Z}_{17}$	01	02	0B	0C	0D	3B	3C	50	51	5A	80	82	84	86	D0	D1	D2
$\mathcal{Z}_{18}$	01	02	0B	0C	0D	3B	3C	50	51	5A	81	83	85	87	D0	D1	D2
$\mathcal{Z}_{19}$	01	02	0B	0C	10	3B	3C	51	52	60	82	84	86	90	D1	D2	E0
$\mathcal{Z}_{20}$	01	02	0B	0C	10	3B	3C	51	5A	60	82	84	86	90	D1	D2	E0
$\mathcal{Z}_{21}$	01	02	0B	0C	1D	3B	3C	51	52	60	82	84	86	90	D1	D2	E0
$\mathcal{Z}_{22}$	01	02	0B	0C	20	3B	3C	51	52	70	82	84	86	A0	D1	D2	F0
$\mathcal{Z}_{23}$	01	02	0B	0C	20	3B	3C	51	5A	70	82	84	86	A0	D1	D2	F0
$\mathcal{Z}_{24}$	01	02	0B	0C	2D	3B	3C	51	52	70	82	84	86	A0	D1	D2	F0
$\mathcal{Z}_{25}$	01	02	0B	0C	2D	3B	3C	51	5A	70	82	84	86	A0	D1	D2	F0

Annexe C

Nous exposons dans ce tableau tourné à 90 degrés les suites des cardinalités, arité par arité, des complétions de $\equiv^{(\gamma)}$.

Les premières cardinalités des complétions de $\mathbf{CAs}^{(\gamma)}$

γ																										
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	2	3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	1	0	3	4	5	18	22	11	12	15	19	25	36	44	52	68	79	93	105	106	109	107
5	0	0	0	0	0	0	1	1	0	0	4	5	8	18	31	36	48	73	111	172	272	455	783					
6	0	0	0	0	0	0	0	1	1	0	0	0	5	6	11	23	30	48	73	117	204	348	589	1004				
7	0	0	0	0	0	0	0	0	1	1	0	0	0	0	6	7	16	24	32	49	88	150	261	475	854			
8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	7	8	21	29	34	53	93	172	311	565		
9	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	8	9	28	30	36	57	101	185	348	648

(4.37)

Annexe D

Nous exposons dans ce tableau tourné à 90 degrés les premiers coefficients des séries de Hilbert des opérades $\mathbf{CAs}^{(\gamma)}$, pour $1 \leq \gamma \leq 9$.

γ	Les premières dimensions de $\mathbf{CAs}^{(\gamma)}$																			
1	1	1	2	5	14	42	132	429	1430	4862	16796	58786	208012	742900	2674440	9694845	35357670	129644790
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	2	4	8	14	20	19	16	14	14	15	16	17	18	19	20	21
4	1	1	2	5	13	35	96	264	724	1973	5335	14390	38872	105141	284929	774254	2111088	5778420
5	1	1	2	5	14	41	124	384	1210	3861	12440	40392	131997	433782	1432696	4752857	15829261	52905635
6	1	1	2	5	14	42	131	420	1375	4576	15431	52598	180895	626862	2186504	7670138	27041833	95764569
7	1	1	2	5	14	42	132	428	1420	4796	16432	56966	199444	704140	2503914	8959699	32236657	116551168
8	1	1	2	5	14	42	132	429	1429	4851	16718	58331	205632	731272	2620176	9449688	34276116	124958386
9	1	1	2	5	14	42	132	429	1430	4861	16784	58695	207452	739840	2658936	9620232	35011566	128082515

(4.38)

Bibliographie

[Awo10] Steve Awodey : Category theory. Oxford University Press, 2010.

[BG16a] Jean-Paul Bultel et Samuele Giraudo : Combinatorial hopf algebras from pros. Journal of Algebraic Combinatorics, 44(2): 455–493, 2016.

[BG16b] Jean-Paul Bultel et Samuele Giraudo : Combinatorial hopf algebras from pros. Journal of Algebraic Combinatorics, 44(2): 455–493, Sep 2016.

[BMP00] Mireille Bousquet-Mélou et Marko Petkovšek : Linear recurrences with constant coefficients : the multivariate case. Discrete Mathematics, 225(1):51 – 75, 2000. FPSAC'98.

[BN98] Franz Baader et Tobias Nipkow : Term rewriting and all that. Cambridge University Press, 1998.

[BN99] Franz Baader et Tobias Nipkow : Term rewriting and all that. Cambridge university press, 1999.

[Bón15] Miklós Bóna : Handbook of enumerative combinatorics, volume 87. CRC Press, 2015.

[Bor17] Nicolas Borie : Three-dimensional catalan numbers and product-coproduct prographs. Sém. Lothar. Combin B, 78, 2017.

[Bou70] Nicolas Bourbaki : Théorie des ensembles. Chapitre IV : Structures, 1970.

[BPR10] Jean Berstel, Dominique Perrin et Christophe Reutenauer : Codes and automata, volume 129. Cambridge University Press, 2010.

[BV68] John Michael Boardman et Rainer Vogt : Homotopy-everything h-spaces. Bulletin of the American mathematical society, 74(6):1117–1122, 1968.

[BV06] John Michael Boardman et Rainer M Vogt : Homotopy invariant algebraic structures on topological spaces, volume 347. Springer, 2006.

[CCG18] Cyrille Chenavier, Christophe Cordero et Samuele Giraudo : Generalizations of the associative operad and convergent rewrite systems. Higher-Dimensional Rewriting and Applications, 2018.

[CCG19] Cyrille Chenavier, Christophe Cordero et Samuele Giraudo : Quotients of the magmatic operad : lattice structures and convergent rewrite systems. Experimental Mathematics, pages 1–18, 2019.

[Cor18] Christophe Cordero : Enumerative combinatorics of prographs. In Formal Power Series and Algebraic Combinatorics, 2018.

[Cor19] Christophe Cordero : A note with computer exploration on the triangle conjecture. In International Conference on Language and Automata Theory and Applications, pages 409–420. Springer, 2019.

[CS97] John Conway et Neil Sloane : Low–dimensional lattices. VII. Coordination sequences. 1997.

[DFR85] Clelia De Felice et Antonio Restivo : Some results on finite maximal codes. RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications, 19(4): 383–403, 1985.

[DK10] Vladimir Dotsenko et Anton Khoroshkin : Gröbner bases for operads. Duke Math. J., 153(2):363–396, 2010.

[Gas05] Olivier Gascuel : Mathematics of evolution and phylogeny. OUP Oxford, 2005.

[GHJMM03] Olivier Gascuel, Michael Hendy, Alain Jean-Marie et Robert McLachlan : The combinatorics of tandem duplication trees. Systematic Biology, 52(1):110–118, 2003.

[Gir18a] Samuele Giraudo : Nonsymmetric Operads in Combinatorics. Springer, 2018.

[Gir18b] Samuele Giraudo : Tree series and pattern avoidance in syntax trees. Prepublication, 2018.

[Han82] Georges Hansel : Baionnettes et cardinaux. Discrete Mathematics, 39(3):331–335, 1982.

[Inc] OEIS Foundation Inc : The On-Line Encyclopedia of Integer Sequences. http ://oeis.org.

[Knu68] Donald Knuth : The art of computer programming 1 : Fundamental algorithms 2 : Seminumerical algorithms 3 : Sorting and searching. MA : Addison-Wesley, 30, 1968.

[KP15] Anton Khoroshkin et Dmitri Piontkovski : On generating series of finitely presented operads. J. Algebra, 426:377–429, 2015.

[Kra49] Leon Gordon Kraft : A device for quantizing, grouping, and coding amplitude-modulated pulses. Thèse de doctorat, Massachusetts Institute of Technology, 1949.

[Laf09] Yves Lafont : Diagram rewriting and operads. Lecture Notes from the Thematic school : Operads CIRM, Luminy (Marseille), 2025, 2009.

[Lam96] Nguyen Huong Lam : A property of finite maximal codes. Acta Mathematica Vietnamica, 21(279-288):5, 1996.

[Lei04] Tom Leinster : Higher Operads, Higher Categories. Higher Operads, Higher Categories. Cambridge University Press, 2004.

[LLMN19] Éric Laugerotte, Jean-Gabriel Luque, Ludovic Mignot et Florent Nicart : Multilinear representations of free pros. Linear and Multilinear Algebra, pages 1–45, 2019.

[Lot97] Lothaire : Combinatorics on words, volume 17. Cambridge university press, 1997.

[ML65] Saunders Mac Lane : Categorical algebra. Bulletin of the American Mathematical Society, 71(1):40–106, 1965.

[ML13] Saunders Mac Lane : Categories for the working mathematician, volume 5. Springer Science & Business Media, 2013.

[MR01] Sabrina Mantaci et Antonio Restivo : Codes and equations on trees. Theoretical Computer Science, 255(1-2):483–509, 2001.

[PS77] Dominique Perrin et Marcel-Paul Schützenberger : Codes et sous-monoïdes possédant des mots neutres. pages 270–281, 1977.

[PS81] Dominique Perrin et Marcel-Paul Schützenberger : A conjecture on sets of differences of integer pairs. Journal of Combinatorial Theory, Series B, 30(1):91–93, 1981.

[Res77] Antonio Restivo : On codes having no finite completions. Discrete Mathematics, 17(3):309–316, 1977.

[Reu85] Christophe Reutenauer : Noncommutative factorization of variable-length codes. Journal of Pure and Applied Algebra, 36:167–186, 1985.

[Row10] Eric Rowland : Pattern avoidance in binary trees. J. Comb. Theory A, 117(6):741–758, 2010.

[RSS89] Antonio Restivo, Sergio Salemi et Tecla Sportelli : Completing codes. RAIRO-Theoretical Informatics and Applications, 23(2):135–147, 1989.

[San00] Arthur Sands : Replacement of factors by subgroups in the factorization of abelian groups. Bulletin of the London Mathematical Society, 32(3):297–304, 2000.

[Sch65] Marcel-Paul Schützenberger : Codes à longueur variable, cours à l'ecole d'été de l'otan sur les méthodes combinatoires en théorie du codage. Royan, France, 1965.

[Sho85] Peter Shor : A counterexample to the triangle conjecture. Journal of Combinatorial Theory, Series A, 38(1):110–112, 1985.

[SP53] August Albert Sardinas et George Patterson : A necessary and sufficient condition for unique decomposition of coded messages. Proceedings Of The Institute Of Radio Engineers, 41(3):425–425, 1953.

[Sta97] Richard Stanley : Enumerative Combinatorics, volume 2. Cambridge university press, 1997.