UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MÉTHODES RÉCURSIVES D'ÉNUMÉRATION ET APPLICATION AUX POLYCUBES-ARBRES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN MATHÉMATIQUES

PAR

LOTFI BOUALLAGUI

JUIN 2018

REMERCIEMENTS

Je tiens à exprimer mes sincères remerciements et ma plus haute considération et reconnaissance à mon directeur de recherche Mr Alexandre Blondin Massé, professeur au Département d'informatique de l'Université du Québec à Montréal, pour son enthousiasme, sa disponibilité et ses conseils avisés.

Je remercie également Mme Halima Elbiaze, professeure au même département, pour son implication et ses encouragements.

J'adresse ma profonde gratitude à ma femme pour son soutien et sa patience.

Ma reconnaissance ultime à Notre Seigneur...

DÉDICACE

À mes parents, à Amine et Sarah.

Don't get the wrong idea - combinatorics is not just putting balls into boxes. Counting finite sets can be a highbrow undertaking, with sophisticated techniques.

Ne vous méprenez pas - la combinatoire ne consiste pas simplement à mettre des balles dans des boîtes. Compter les ensembles finis peut être une entreprise de haut niveau, avec des techniques sophistiquées.

Gian-Carlo ROTA

TABLE DES MATIÈRES

LISTE DES FIGURES xi

LISTE DES TABLEAUX xiii

RÉSUMÉ xv

INTRODUCTION 1

CHAPITRE I
STRUCTURES COMBINATOIRES FRÉQUENTES 5

1.1 Les Polyominos 5

1.1.1 Résultats asymptotiques 6

1.1.2 Sous-classes particulières 7

1.2 Les Polycubes 8

1.3 Les Arbres 9

1.4 Les Chemins 11

CHAPITRE II
MÉTHODE SYMBOLIQUE 13

2.1 Fonctions génératrices 14

2.1.1 Classe combinatoire 14

2.1.2 Types de fonctions génératrices 15

2.1.3 Calcul sur les séries formelles 18

2.2 Constructions admissibles 21

2.2.1 Admissibilité 21

2.2.2 Constructions de base 21

2.3 Spécifications combinatoires 26

2.4 Exemples d'application de la méthode symbolique 27

CHAPITRE III

GRAMMAIRES D'OBJETS 31
3.1 Généralités et définitions 31
3.2 Formes normales 38
3.2.1 Forme normale réduite 38
3.2.2 Forme normale 1-2 41
3.2.3 forme normale complète 41
3.3 Non-ambiguïté 42
3.4 Notations équationnelle et schématique 44
3.4.1 Notation équationnelle 44
3.4.2 Représentation schématique 44
3.5 Application à l'énumération 45
3.5.1 Théorème d'énumération 45
3.5.2 Valuations d'objets linéaires 47
CHAPITRE IV
MÉTHODE *ECO* 51
4.1 Énumération par la méthode *ECO* 52
4.1.1 Principe de la méthode 52
4.1.2 Arbre de génération 53
4.1.3 Règles de succession 54
4.1.4 Exemples 54
4.2 Génération exhaustive par la méthode *ECO* 64
4.2.1 Présentation de l'algorithme 64
4.2.2 Analyse de la complexité 65
CHAPITRE V
GÉNÉRATION EXHAUSTIVE DES POLYCUBES-ARBRES 67
5.1 Définitions 68
5.2 État de l'art 68

5.3 Codage des polycubes-arbres 69
5.3.1 Racine d'un polycube-arbre 70
5.3.2 Étiquetage d'un polycube-arbre fixe 71
5.3.3 Ordre sur les sommets de l'arbre 71
5.3.4 Codage *DFSE* des polycubes-arbres fixes 72
5.3.5 Forme canonique *DFCS* des polycubes-arbres libres 74
5.4 Arbre de génération 75
5.4.1 Structure de l'arbre de génération 76
5.4.2 Opérateur d'expansion locale 76
5.5 Résultats obtenus 80
Conclusion 87
BIBLIOGRAPHIE 89

LISTE DES FIGURES

Figure		Page
1.1	Réseaux réguliers en dimension 2	5
1.2	Un polyomino	6
1.3	Polyominos de différentes classes	8
1.4	$3d$-Polycubes	9
1.5	Un arbre planaire non-étiqueté	11
1.6	Exemples de chemins dans $\mathbb{N}^2$	12
4.1	Un arbre planaire et ses successeurs par ϑ	56
4.2	Arbre de génération des arbres planaires associé à ϑ	57
4.3	Partition de la classe des polyominos convexes	60
4.4	L'opérateur ϑ pour la classe C_b	61
4.5	L'opérateur ϑ pour la classe C_a	61
4.6	L'opérateur ϑ pour la classe C_r	62
4.7	L'opérateur ϑ pour la classe C_g	62
4.8	Arbre de génération des polyominos convexes associé à l'opérateur ϑ	63
5.1	Polycubes-arbres	68
5.2	Définition de racines pour les arbres non-enracinés	70
5.4	Code $DFSE$ de polycube-arbre fixe	73
5.5	Forme canonique d'un polyomino-arbre libre	75
5.6	Illustration de l'opérateur ϑ	79
5.7	$3d$-Pleinement-feuillu de taille 15	85

LISTE DES TABLEAUX

Tableau Page

5.1 Polyominos-arbres libres 82

5.2 $3d$-Polycubes-arbres libres 83

5.3 $4d$-Polycubes-arbres libres 84

RÉSUMÉ

Ce mémoire se situe dans le cadre de la combinatoire énumérative et s'intéresse particulièrement aux approches basées sur les constructions récursives. Une application est effectuée pour la génération exhaustive des *polycubes-arbres*, une classe particulière de polycubes multidimensionnels dont le graphe dual, graphe d'adjacence des cellules dans le réseau hypercubique, est connexe et acyclique. La génération est restreinte aux dimensions 2, 3 et 4 et à la forme libre de ces objets.

On explore les développements les plus récents pour trois méthodes qui partagent la même approche récursive, qui sont la méthode symbolique, les grammaires d'objets et la méthode *ECO*. La méthode symbolique fournit un dictionnaire pour la traduction systématique de la majorité des constructions discrètes en opérations algébriques sur les fonctions génératrices. C'est un outil simple et formel qui réduit l'étude d'une classe d'objets à la tâche plus facile de lui trouver une spécification en termes de constructions combinatoires de base. Les grammaires d'objets sont aussi des constructions récursives des objets, et sous certaines conditions, elles se traduisent en équations sur les fonctions génératrices. Quant à la méthode *ECO*, elle construit les objets par expansion locale d'objets plus petits, ce qui se traduit aussi par des équations sur la fonction génératrice de la classe énumérée.

Mots clés: polyomino-arbre, polycube-arbre, énumération, fonction génératrice, grammaire d'objets, méthode ECO

INTRODUCTION

Ce mémoire présente, dans le cadre de la combinatoire énumérative, certains aspects récents d'une approche algébrique et analytique, qui se fonde sur la description récursive des objets analysés.

L'analyse combinatoire s'intéresse à l'étude de classes d'objets discrets, qui servent le plus souvent comme modèles pour des objets réels. Les motivations pour l'étude de ces objets proviennent de nombreuses disciplines : l'étude des polyominos et polycubes multidimensionnels est un ancien problème de la physique statistique, en particulier de la théorie de la percolation (Broadbent et Hammersley, 1957). En chimie et en biologie moléculaire, les molécules sont souvent modélisées par des polycubes (Denise, 2001), la combinatoire des mots a des applications dans plusieurs domaines comme la recherche de motifs en génomique ou la reconnaissance des formes en géométrie discrète (Provençal, 2008), de même l'arbre des suffixes est une structure de données principale pour l'algorithmique du texte.

Une compréhension profonde de la structure d'une classe quelconque d'objets discrets doit commencer par l'énumération de ses éléments, de manière exhaustive, partielle ou aléatoire, exacte ou approximative, en utilisant des algorithmes ou en définissant des relations entre eux.

Étant donnée une classe $\mathcal{O}$ d'objets discrets et un paramètre p, appelé taille, sur cette classe, on considère l'ensemble O_n des objets de $\mathcal{O}$ de taille n où n est un entier positif. Le paramètre p est dit *discriminateur* si pour tout n, O_n est fini (Pergola *et al.*, 2002), on note alors $a_n = |O_n|$. Dans certains cas a_n est déterminé par une expression explicite, directement calculable, ou même par une formule de récurrence avec une

procédure simple pour sa résolution progressive. Dans d'autres cas, si on n'a pas une formule simple pour a_n, on peut en avoir une pour la série formelle $f(x) = \sum_{n\geq 0} a_n x^n$, appelée *fonction génératrice ordinaire* de la classe $\mathcal{C}$ pour le paramètre p. On pourrait alors déduire les coefficients a_n par des techniques de calcul standards (Goulden et Jackson, 1983; Graham *et al.*, 1989), ou encore obtenir des valeurs asymptotiques de ces coefficients par l'analyse des singularités de la fonction génératrice (Flajolet et Sedgewick, 2009).

La méthodologie d'énumération considérée dans ce mémoire repose sur l'idée de définir et exploiter une description récursive de la classe d'objets énumérée. Elle est illustrée par trois techniques principales : la méthode symbolique, les grammaires d'objets et la méthode *ECO*.

La méthode symbolique (Flajolet et Sedgewick, 2009) remonte à la théorie des espèces de structures, introduite par Joyal vers 1980, qui décrit les objets récursivement par des opérations entre eux, ce qui se traduit par des opérations entre les fonctions génératrices correspondantes. Cette dernière a fait l'objet de nombreux développements, en particulier par (Bergeron *et al.*, 1998), arrivant à la méthode symbolique.

La technique de grammaire d'objets, introduite par (Dutour et Fedou, 1998), tire son origine de la méthodologie de Schützenberger (Schützenberger, 1963). qu'il appelait méthodologie DSV (Dyck-Schützenberger-Viennot). Cette dernière se décompose en trois étapes : d'abord, construire une bijection entre les objets et les mots d'un langage algébrique. Puis, si la grammaire associée est non-ambiguë, alors ses productions se traduisent par un système d'équations dont l'une des solutions est la fonction génératrice de la classe énumérée. D'où l'idée d'étendre le concept classique de grammaire des mots à un concept plus général de grammaire d'objets.

Enfin, La méthode *ECO* (Enumerating Combinatorial Objects), introduite pendant les années 90 par Renzo Pinzani et son groupe de recherche (Barcucci *et al.*, 1999),

construit chaque objet par expansion locale d'un objet plus petit, avec une certaine régularité pour être facilement décrite à l'aide d'une règle de succession, qui se traduit par une équation fonctionnelle satisfaite par la fonction génératrice.

Un algorithme a été conçu, en s'inspirant de cette méthode, pour la génération exhaustive des *polycubes-arbres libres* multidimensionnels. Il a permis de générer ces objets jusqu'aux tailles 16, 12 et 11, pour les dimensions 2, 3 et 4 respectivement. Un filtrage des objets générés, relativement au nombre de feuilles, a permis d'énumérer la sous-classe des polycubes-arbres libres *pleinement feuillus* (Blondin-Massé *et al.*, 2017), jusqu'aux tailles 22, 21 et 21 pour les mêmes dimensions.

Le mémoire est organisé de la manière suivante :

- Le premier chapitre présente quelques structures combinatoires fondamentales et utiles pour la suite du mémoire. On s'intéresse aux polyominos et polycubes dans les réseaux hypercubiques, aux structures arborescentes et aux chemins dans le réseau carré.
- Dans le deuxième chapitre, on introduit l'approche symbolique de l'énumération combinatoire. Cette méthode se situe dans le cadre de la combinatoire analytique qui vise à décrire les problèmes combinatoires dans le langage des fonctions génératrices, et d'en déduire des résultats exacts et asymptotiques par les techniques usuelles du calcul algébrique et de l'analyse complexe. Cette approche a connu de nombreuses contributions dans la littérature, mais elle a été développée et organisée dans un formalisme unifié notamment par Philippe Flajolet (Flajolet et Sedgewick, 2009). La méthode symbolique consiste en une association entre des opérations ensemblistes sur les classes combinatoires et des opérations algébriques sur les fonctions génératrices correspondantes. Le processus de traduction est complètement formel et s'applique aux constructions de base les plus importantes. Ce formalisme permet de résoudre de nombreux problèmes de nature

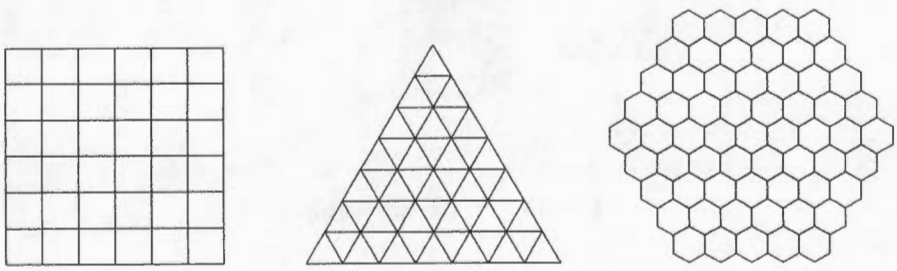
combinatoire, comme l'énumération exacte de certaines classes ou l'obtention de résultats asymptotiques, et il possède des applications importantes pour l'analyse des algorithmes et pour l'étude de propriétés probabilistes de grandes structures combinatoires. On présente cette méthode symbolique par quelques constructions de base pour les structures non-étiquetées, et on donne des exemples d'application.

- Le troisième chapitre est consacré à l'introduction des grammaires d'objets. C'est une généralisation des grammaires algébriques basée sur la description récursive des objets combinatoires. On présente quelques méthodes de résolution ainsi que des exemples d'application.
- Au quatrième chapitre on aborde la méthode *ECO* et certaines de ses applications. Cette méthode dérive elle aussi de la méthodologie générale de construction récursive des objets, et se caractérise par sa simplicité et par l'efficacité des algorithmes qui en découlent. Après avoir explicité la méthode, on décrit le lien avec les fonctions génératrices, puis un exemple d'application pour la génération exhaustive des polyominos convexes est détaillé.
- Finalement, le cinquième chapitre présente un travail expérimental qui consiste à l'implémentation d'un algorithme inspiré de la méthode *ECO*, pour la génération exhaustive des polycubes-arbres libres dans les réseaux hypercubiques. Un résumé est donné des résultats obtenus pour les dimensions 2, 3 et 4.

CHAPITRE I

STRUCTURES COMBINATOIRES FRÉQUENTES

Parmi les structures combinatoires les plus présentes et les plus étudiées dans la littérature, on trouve les polyominos, les polycubes, les chemins, les polygones ainsi que les arbres et les structures arborescentes. Les chemins et les polyominos peuvent être définis sur différents *réseaux réguliers* du plan, tels que les réseaux carré, triangulaire et hexagonal, illustrés dans la figure suivante :



(a) Carré (b) Triangulaire (c) Hexagonal

Figure 1.1: Réseaux réguliers en dimension 2

Dans la suite on ne considère que les réseaux carré, cubique et hypercubique $\mathbb{Z}^d$.

1.1 Les Polyominos

Le terme *polyomino* a été introduit par Solomon W. Golomb en 1953 dans un ouvrage intitulé Polyominoes (Golomb, 1996), puis rendu populaire par Martin Gardner une

année après. Un polyomino est constitué de plusieurs carrés unitaires connectés par leurs côtés :

Définition 1.1. Dans le plan $\mathbb{Z}^2$, une cellule est un carré unitaire, et un polyomino est défini, à translation près, comme une union finie et connexe de cellules. L'adjacence de deux cellules étant définie par l'existence d'un côté commun.

Un polyomino peut être caractérisé par plusieurs paramètres, notamment l'aire, le périmètre, la largeur et la hauteur. L'*aire*, appelée aussi taille, est le nombre de cellules, le *périmètre* est la longueur du contour, la *largeur* est le nombre de colonnes et la *hauteur* est le nombre de lignes.

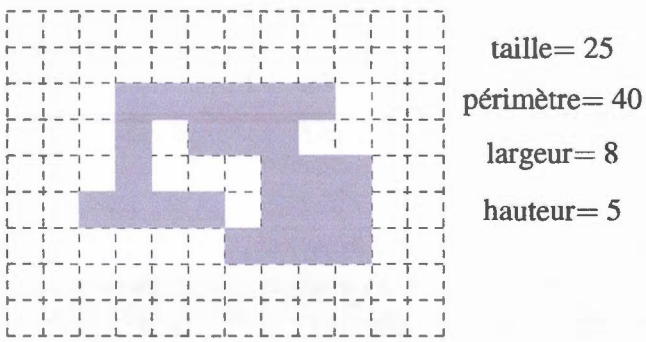


Figure 1.2: Un polyomino

1.1.1 Résultats asymptotiques

Le problème général d'énumération de polyominos, selon la taille ou le périmètre, est difficile et il est encore ouvert. En revanche, quelques résultats asymptotiques ont été obtenus. Par exemple, le record actuel pour le nombre a_n de polyominos d'aire n est détenu par (Jensen, 2003) avec

$$a_{56} = 69\,150\,714\,562\,532\,896\,936\,574\,425\,480\,218$$

De même, (Klarner et Rivest, 1973) ont montré que $\lambda_2 = \lim_n a_n^{\frac{1}{n}}$ existe, appelée constante de Klarner. Puis, (Madras, 1999) a prouvé la convergence du quotient a_{n+1}/a_n vers λ_2, qui est alors le taux d'accroissement limite du nombre de polyominos. À l'heure actuelle, les meilleures bornes, inférieure (Barequet *et al.*, 2016) et supérieure (Klarner et Rivest, 1973), connues sur λ_2 sont les suivantes : $4.0025 < \lambda_2 < 4.5685$. Une estimation numérique non prouvée, $\lambda_2 \approx 4.062570$ (Jensen et Guttmann, 2000), est basée sur l'analyse des 46 premiers termes de la fonction génératrice.

Les polyominos peuvent être *fixes* ou *libres*. Ils sont fixes lorsqu'ils forment des classes d'équivalence à translation près, et sont libres lorsqu'ils forment des classes d'équivalence à translation, rotation et réflexion près. Soit b_n le nombre de polyominos libres de taille n, dans le réseau carré, alors asymptotiquement $a_n \approx 8b_n$, et il est admis (Guttmann, 2009), sans être prouvé, que $b_n \sim A\lambda_2^n n^\theta$ lorsque $n \to \infty$, où A est une constante positive et θ est une constante appelée exposant critique. Par suite d'arguments physiques il est conjecturé que $\theta = -1$ (Parisi et Sourlas, 1981).

1.1.2 Sous-classes particulières

En raison de la difficulté du problème dans sa forme générale, la communauté scientifique s'est orientée vers l'énumération de sous-classes particulières de polyominos pour comprendre plus en profondeur la structure de ces objets. La plupart de ces sous-classes sont définies par des contraintes de convexité ou d'existence d'une direction privilégiée, et les solutions peuvent être de plusieurs formes, dont une formule exacte, une fonction génératrice, une récurrence ou un algorithme.

Un polyomino est dit *verticalement* (resp. *horizontalement*) *convexe* si chacune de ses colonnes (resp. lignes) est connexe. Un polyomino à la fois verticalement et horizontalement convexe est dit *convexe*. Il est dit *dirigé* s'il contient une cellule distinguée, appelée *source*, à partir de laquelle chacune de ses cellules peut être atteinte par un che-

min formé de pas $Nord(0,1)$ et $Est(1,0)$. Des exemples de polyominos de différents types sont montrés ci-après :

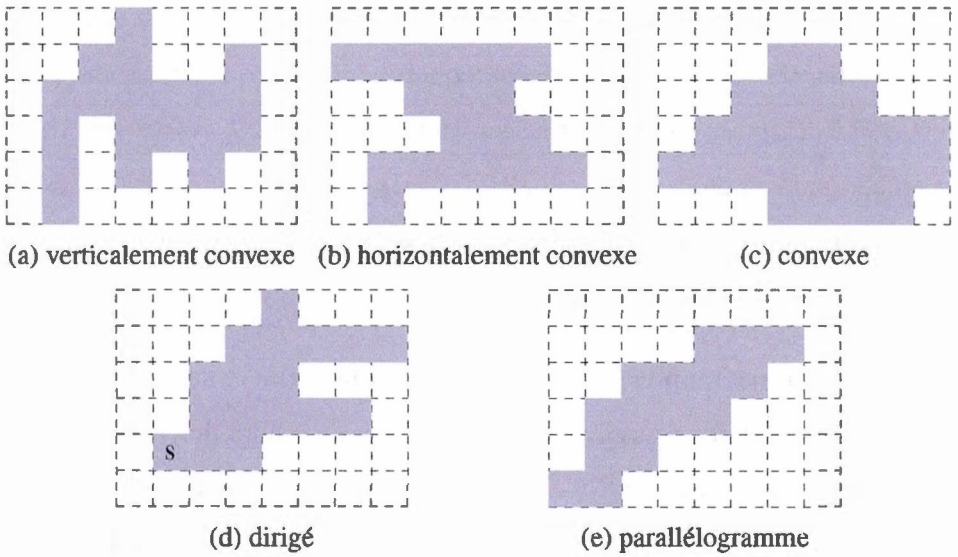


Figure 1.3: Polyominos de différentes classes

1.2 Les Polycubes

En dimension $d > 2$, l'analogue d'un polyomino du réseau carré est appelé polycube sur le réseau hypercubique $\mathbb{Z}^d$ (Figure1.4 avec $d = 3$).

Définition 1.2. Un *polycube* d-dimensionnel de taille n est un ensemble connexe de n hypercubes unitaires, appelés cellules, du réseau hypercubique $\mathbb{Z}^d$, où la connectivité se fait à travers les faces de dimension $d-1$ de ses cellules.

Les polycubes sont dits *fixes* lorsqu'ils forment des classes d'équivalence à translation près. Ils sont dits *libres* lorsqu'ils forment des classes d'équivalence à translation, rotation et réflexion près. Dans les deux cas, tous les polycubes d'une même classe d'équivalence comptent pour un seul.

Figure 1.4: $3d$-Polycubes

Comme pour les polyominos, l'énumération des polycubes d-dimensionnels est un problème ouvert, tant pour le nombre $A_d(n)$ de polycubes fixes de taille n, que pour le taux d'accroissement limite λ_d. De ce fait, les travaux de recherche sont plus orientés vers la construction d'algorithmes efficaces pour calculer le plus de termes $A_d(n)$ possible. Les meilleurs résultats connus à ce jour sont dûs à (Aleksandrovicz et Barequet, 2006) pour le calcul des premiers termes jusqu'à $A_3(19)$, $A_4(16)$, $A_5(15)$, $A_6(15)$ et $A_7(14)$, et à (Luther et Mertens, 2011) qui ont calculé $A_8(12)$ et $A_9(12)$. D'autre part, (Madras, 1999) a montré que le taux d'accroissement limite du nombre de polycubes $\lambda_d = \lim_n \frac{A_d(n+1)}{A_d(n)}$ existe pour tout $d \geq 2$. (Barequet *et al.*, 2010) ont prouvé le résultat asymptotique suivant : $\lambda_d = 2ed - o(d)$ où e est la base du logarithme naturel, et ont conjecturé que $\lambda_d \to (2d-3)e$ lorsque $d \to \infty$.

En dimension 3, comme c'est le cas en dimension 2, il est admis, sans preuve à l'appui, que le nombre de polycubes fixes de taille n vérifie l'équivalence asymptotique $A_3(n) \sim A\lambda_3^n n^\theta$ lorsque $n \to \infty$, où A et θ sont des constantes réelles. L'analyse des données disponibles confirme la prédiction de (Guttmann et Gaunt, 1978) que $\theta = -1.5$ et que $\lambda_3 \approx 8.3479$.

1.3 Les Arbres

La structure d'arbre est utilisée dans plusieurs disciplines et elle est à la base de nombreuses applications fondamentales en informatique. Par exemple, elle intervient dans les algorithmes de recherche ou de tri, dans l'organisation des fichiers dans les sys-

tèmes d'exploitation, ou encore dans la représentation des programmes traités par un compilateur, en particulier, les arbres binaires sont utilisés en compilation pour l'évaluation efficace des expressions arithmétiques. Les arbres jouent aussi un rôle important dans l'analyse des algorithmes (Kemp, 1984), d'une part parce qu'ils sont directement impliqués dans plusieurs algorithmes de base, et d'autre part parce qu'ils représentent implicitement la structure récursive de certains programmes. Pour la suite, on adopte les définitions suivantes pour les arbres (Bergeron *et al.*, 1998; Flajolet et Sedgewick, 2009), et on se limite au cas fini.

Définition 1.3. Un *arbre* est un graphe non-orienté, connexe et sans cycle. Il peut éventuellement être vide. Lorsqu'un arbre est non vide et que l'un de ses sommets est distingué, on dit qu'il est enraciné, et le sommet distingué est appelé *racine* de l'arbre. Un arbre enraciné est naturellement orienté, en dirigeant chaque arête dans le sens d'éloignement de la racine, et dans ce cas il est appelé *arborescence*. Si de plus les fils de chaque sommet, lorsqu'il en existe, sont ordonnés entre eux, alors l'arborescence est appelée arbre *planaire*.

La *taille* d'un arbre est le nombre de ses sommets. Dans une arborescence, les sommets sans fils sont appelés *feuilles* et les autres des *nœuds internes*. La *hauteur* est la longueur du chemin le plus long reliant la racine à une feuille, la *profondeur* d'un sommet est la longueur du chemin le reliant à la racine, et la *longueur du chemin interne* est la somme des profondeurs de tous les sommets (Figure 1.5).

On remarque qu'un arbre planaire peut être défini récursivement par la donnée de la racine r, avec ou bien $\emptyset$ ou bien un ensemble fini ordonné d'arbres planaires $A_1, \cdots, A_k$ où k est un entier positif. Donc, un arbre planaire A, qui n'est pas réduit à un seul sommet, peut être représenté par une liste $A = (r.A_1, \cdots .A_k)$. Les racines des arbres A_i sont les fils de r, et k est son degré.

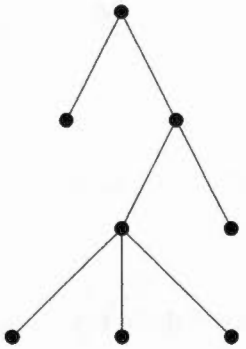


Figure 1.5: Un arbre planaire non-étiqueté

Le nombre d'arbres étiquetés de taille n est égal à n^{n-2}, et le nombre d'arborescences étiquetées de taille n est égal à n^{n-1}, d'après la formule de *Cayley* (Flajolet et Sedgewick, 2009) p127.

Pour les arbres et les arborescences non-étiquetés, on ne connait pas, à l'heure actuelle, de formule explicite pour leur dénombrement, mais il y a des équations fonctionnelles sur leurs fonctions génératrices ordinaires, voir Exemple 2.10 ci-après et (Flajolet et Sedgewick, 2009), et il y a aussi une formule de récurrence donnée par (Finch, 2003), ce qui a permis de calculer quelques premiers termes de leurs nombres respectifs en fonction de la taille (*OEIS* A000055 et A000081).

Enfin, le nombre d'arbres planaires non-étiquetés, appelés aussi arbres de *Catalan*, de taille $n+1$ est égal au n-ième nombre de *Catalan* (Flajolet et Sedgewick, 2009) p65 :

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

1.4 Les Chemins

On s'intéresse aux chemins dans le plan discret $\mathbb{Z}^2$.

Pour $x = (x_1, x_2) \in \mathbb{Z}^2$, soit la norme $\|x\|_\infty = max\{|x_1|, |x_2|\}$

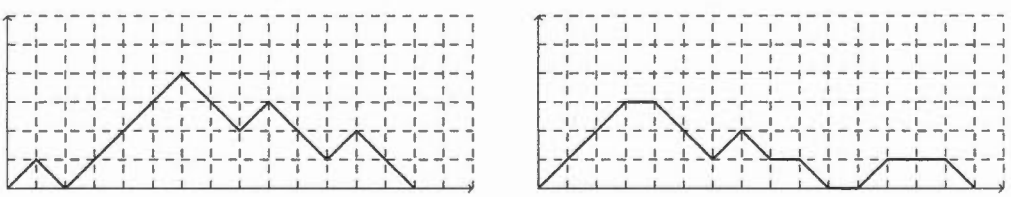
Définition 1.4. Un *chemin* de longueur n dans le plan discret $\mathbb{Z}^2$, est une suite de points $s_0, s_1, \cdots, s_n$ tels que $\|s_{i+1} - s_i\|_\infty \leq 1$ pour $i = 0, \cdots, n-1$.

Les points s_i sont les sommets du chemin, et un couple de sommets consécutifs (s_i, s_{i+1}) est appelé un *pas*.

Deux familles particulières de chemins du réseau carré discret $\mathbb{N}^2$ sont les chemins de *Motzkin* et les chemins de *Dyck* (Figure1.6) :

- Les chemins de *Motzkin* ont pour extrêmités $(0,0)$ et $(n,0)$, et sont formés de trois types de pas : $E(1,0)$, $NE(1,1)$ et $SE(1,-1)$.
- Les chemins de *Dyck* ont pour extrêmités $(0,0)$ et $(2n,0)$, et sont formés uniquement de deux types de pas : $NE(1,1)$ et $SE(1,-1)$.

(Flajolet, 1980) fournit une interprétation combinatoire des fractions continues en termes de chemins de *Motzkin*. Ils interviennent également dans des problèmes informatiques telles que l'évaluation des algorithmes sur les fichiers par (Flajolet *et al.*, 1980). ou l'énumération et la génération aléatoire de nombreux objets combinatoires (Viennot, 1985; Barcucci *et al.*, 1991). Pour une étude détaillée sur la combinatoire des chemins, on pourra se référer à (Goulden et Jackson, 1983).



(a) Chemin de Dyck (b) Chemin de *Motzkin*

Figure 1.6: Exemples de chemins dans $\mathbb{N}^2$

CHAPITRE II

MÉTHODE SYMBOLIQUE

La méthode symbolique pour l'énumération de structures combinatoires découle de l'élaboration d'un formalisme algébrique unifié décrivant le lien entre les structures et les fonctions génératrices. Ce mécanisme est simple, général et complètement formel. Il permet la traduction systématique de constructions combinatoires en opérations algébriques sur les fonctions génératrices, grâce à un dictionnaire établi pour des constructions élémentaires, comme l'union disjointe, le produit cartésien, l'ensemble des parties, les suites et les cycles. Ainsi le problème d'énumération se réduit à la recherche d'une spécification de la classe considérée par des constructions de base, ce qui se traduit en une récurrence ou une équation fonctionnelle, permettant dans plusieurs cas de trouver la solution exacte, et dans d'autres cas plus compliqués, on pourrait déduire des informations asymptotiques difficiles à obtenir autrement.

Dans la suite on définit les fonctions génératrices. Puis, l'essentiel de la méthode symbolique est explicité pour le cas des structures non-étiquetées, lesquelles sont associées aux fonctions génératrices ordinaires. Un mécanisme équivalent existe pour les structures étiquetées et les fonctions génératrices exponentielles. Cette méthode peut être adaptée aux structures paramétrées, grâce aux fonctions génératrices à plusieurs variables.

2.1 Fonctions génératrices

Le problème d'énumération consiste à déterminer le nombre de configurations combinatoires décrites par un nombre fini de règles, pour toutes les tailles possibles. Les solutions sont alors des suites infinies de termes, qu'on peut parfaitement représenter par des séries formelles, appelées alors *fonctions génératrices*. Les séries formelles sont une généralisation des polynômes aux séries infinies de la forme $\sum_{n\geq 0} f_n z^n$, où z est une indéterminée formelle. et les coefficients f_n appartiennent à un anneau $\mathbb{A}$, généralement $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$. On écrit $f(z) = \sum_{n\geq 0} f_n z^n$ et on note $[z^n]\, f(z) = f_n$ le coefficient de z^n dans $f(z)$.

Les notions de convergence ne sont pas considérées pour les séries formelles. Elles sont des objets formels et purement algébriques qui agissent simplement comme supports pour les suites qu'elles représentent. Pourtant, les fonctions génératrices s'avèrent être un outil puissant de la combinatoire énumérative (Roberts et Tesman, 2005). Elles contiennent de manière très compacte l'information sur l'infinité de termes des suites correspondantes, et peuvent être obtenues de plusieurs manières, ce qui permet dans plusieurs cas d'obtenir la suite par des formules exactes ou par des récurrences. D'autre part, elles peuvent être interprétées comme objets analytiques, permettant ainsi d'extraire des informations sur la suite, même si elle est inconnue. En particulier, l'analyse des singularités fournit souvent des propriétés asymptotiques des structures combinatoires de grande taille (Flajolet et Sedgewick, 2009).

2.1.1 Classe combinatoire

L'énumération est concernée par des objets discrets, tels que les arbres, les mots et les permutations, selon des paramètres caractéristiques.

Définition 2.1. Une classe combinatoire est un ensemble $\mathscr{A}$ fini ou dénombrable tel

que :

(i) $\mathscr{A}$ est muni d'une fonction taille $|\cdot| : \mathscr{A} \to \mathbb{N}$

(ii) le nombre d'éléments de taille donnée est fini.

Les éléments d'une classe combinatoire sont appelés des objets, et la taille d'un objet est le nombre d'éléments de taille 1, appelés *atomes*, qui le constituent. Soit A_n l'ensemble des objets de taille n, il est fini par définition. On note $a_n = card(A_n)$.

Exemple 2.1. Deux exemples de classes combinatoires :

1. La classe $\mathscr{W}$ des mots binaires sur l'alphabet $\{0,1\}$ est telle que le nombre de mots binaires de longueur n est $w_n = 2^n$.
2. La classe $\mathscr{P}$ des permutations vérifie $p_n = n!$

2.1.2 Types de fonctions génératrices

Il existe plusieurs types de fonctions génératrices : ordinaires, exponentielles, de Dirichlet et autres. Chaque type est mieux adapté que les autres selon le type de la structure énumérée et la nature du problème à résoudre. Dans ce chapitre on s'intéresse aux structures ordinaires non-étiquetées, pour lesquelles on utilisera les *fonctions génératrices ordinaires*.

Définition 2.2. Soit $\{a_n\}_{n\geq 0}$ la suite de décompte d'une classe combinatoire $\mathscr{A}$. La fonction génératrice ordinaire (*fgo*) associée à la classe $\mathscr{A}$ est la série formelle

$$a(z) = \sum_{n\geq 0} a_n z^n.$$

Par contre, pour certaines classes, il se peut que les atomes ne soient pas équivalents, et ainsi les objets se trouvent naturellement étiquetés. Dans ce cas, il est profitable de

considérer les *fonctions génératrices exponentielles*, qui ont l'avantage de traduire de manière plus simple les constructions étiquetées, et en particulier le produit étiqueté.

Définition 2.3. La fonction génératrice exponentielle (*fge*) associée à la classe $\mathscr{A}$ est la série formelle

$$a(z) = \sum_{n\geq 0} a_n \frac{z^n}{n!}.$$

La puissance de la méthode symbolique est qu'on peut l'utiliser non seulement pour dénombrer des objets combinatoires, mais aussi pour quantifier leurs propriétés. Elle permet de prendre en compte les paramètres des configurations simplement en ajoutant des variables supplémentaires, et ainsi étendre son mécanisme formel à la traduction d'objets paramétrés en fonctions génératrices à plusieurs variables. Il en résulte un outil simple et d'une grande utilité pour l'analyse de plusieurs sortes de paramètres, et d'en déduire des informations précises sur leurs propriétés asymptotiques.

Définition 2.4. Un paramètre multidimentionnel sur une classe $\mathscr{A}$ est une fonction

$$\chi = (\chi_1, \cdots, \chi_d) : \mathscr{A} \longrightarrow \mathbb{N}^d \text{ où } d \text{ est un entier positif.}$$

Dans le cas $d = 1$ on dit que χ est un paramètre scalaire, sinon il est appelé multiparamètre. La suite de décompte de $\mathscr{A}$ est alors notée

$$a_{n,k_1,\cdots,k_d} = card\{\alpha \in \mathscr{A} : |\alpha| = n \text{ et } \chi_i(\alpha) = k_i \ , \ i = 1, \cdots, d\}.$$

Définition 2.5. Soit une classe $\mathscr{A}$ sur laquelle est défini un paramètre $\chi = (\chi_1, \cdots, \chi_d)$. La fonction génératrice associée à la paire $(\mathscr{A}, \chi)$ est définie par la série formelle

- *ordinaire* $\quad a(z, u_1, \cdots . u_d) = \sum_{n,k_1,\cdots,k_d} a_{n,k_1,\cdots,k_d} u_1^{k_1} \cdots u_d^{k_d} z^n.$
- *exponentielle* $\quad a(z, u_1, \cdots, u_d) = \sum_{n,k_1,\cdots,k_d} a_{n,k_1,\cdots,k_d} u_1^{k_1} \cdots u_d^{k_d} \frac{z^n}{n!}.$

Exemple 2.2. Exemples de fonctions génératrices de classes combinatoires

1. La fgo de la classe $\mathscr{W}$ des mots binaires est

$$w(z) = \sum_{n\geq 0} 2^n z^n = \frac{1}{1-2z}.$$

2. La fge de la classe $\mathscr{P}$ des permutations est

$$p(z) = \sum_{n\geq 0} n! \frac{z^n}{n!} = \frac{1}{1-z}.$$

Remarquons que la fgo $\sum_{n\geq 0} n! z^n$ est de rayon de convergence nul, et par suite elle ne peut pas s'écrire sous forme close à l'aide de fonctions usuelles.

3. Sur la classe $\mathscr{W}$ précédente, des mots binaires, considérons le paramètre scalaire χ qui donne pour chaque mot le nombre d'occurrences d'une lettre désignée. Le nombre de mots binaires de longueur n et ayant k occurrences de la lettre désignée, est donné par le coefficient binomial $w_{n,k} = \binom{n}{k}$. Alors la fgo associée à $(\mathscr{W}, \chi)$ est comme suit :

$$\begin{aligned} w(z,u) &= \sum_{n,k\geq 0} \binom{n}{k} u^k z^n = \sum_{n\geq 0} \left(\sum_{k=0}^{n} \binom{n}{k} u^k \right) z^n = \sum_{n\geq 0} (1+u)^n z^n \\ &= \frac{1}{1-z(1+u)}. \end{aligned}$$

On peut déduire la fgo associée aux coefficients binomiaux pour une valeur fixée de k :

$$w^{(k)}(z) = \sum_{n\geq 0} \binom{n}{k} z^n = \frac{z^k}{(1-z)^{k+1}}.$$

2.1.3 Calcul sur les séries formelles

Le calcul sur les séries formelles étend les opérations algébriques usuelles sur les polynômes.

Opérations usuelles :

L'ensemble des séries formelles à coefficients dans un anneau $\mathbb{A}$, muni de l'addition et du produit de Cauchy, est un anneau qu'on note $\mathbb{A}[\![z]\!]$, il est commutatif si $\mathbb{A}$ l'est :
Soient $a(z) = \sum_{n\geq 0} a_n z^n$ et $b(z) = \sum_{n\geq 0} b_n z^n$, on définit :

$$a(z) + b(z) = \sum_{n\geq 0} (a_n + b_n) z^n.$$

$$a(z) \cdot b(z) = \sum_{n\geq 0} c_n z^n \text{ où } c_n = \sum_{k=0}^{n} a_k b_{n-k}.$$

Proposition 2.1. *$a(z)$ est inversible dans $\mathbb{A}[\![z]\!]$ si et seulement si a_0 est inversible dans $\mathbb{A}$. L'inverse, lorsqu'il existe, est unique.*

Preuve. $a(z)$ est inversible si et seulement s'il existe $b(z)$ dans $\mathbb{A}[\![z]\!]$ tel que $a(z)b(z) = 1$, donc si et seulement si $a_0 b_0 = 1$ et $\sum_{k=0}^{n} a_k b_{n-k} = 0$ pour tout $n \geq 1$ $(*)$
Si $a(z)$ est inversible, alors on a $a_0 b_0 = 1$ et donc a_0 est inversible.
Réciproquement, si a_0 est inversible alors le système $(*)$ admet une solution unique $b(z)$, qui est l'inverse de $a(z)$. □

Exemple 2.3. L'inverse de $1 - z$ est $(1-z)^{-1} = \sum_{n\geq 0} z^n$. En effet, d'après ce qui précède on a $a_0 = 1$, $a_1 = -1$ et $a_n = 0$ pour tout $n > 1$, ce qui donne $b_n = 1$ pour tout $n \geq 0$.

Il y a aussi la composition des séries formelles $b(a(z)) = \sum_{n\geq 0} b_n a(z)^n$ qui est définie si et seulement si $a_0 = 0$ ou $b(z)$ est un polynôme.

Exemple 2.4. Si $a_0 = 0$ alors la somme $Q(a(z)) = \sum_{n\geq 0} a(z)^n$ est définie, et vérifie $Q(a(z))(1-a(z)) = 1$. On écrit alors $Q(a(z)) = (1-a(z))^{-1}$ et on l'appelle *quasi-inverse* de $a(z)$. On écrit formellement $\sum_{n\geq 0} a^n = \frac{1}{1-a}$.

On dit que $a(z)$ et $b(z)$ sont réciproques l'une de l'autre lorsque $a(b(z)) = b(a(z)) = z$.

Proposition 2.2. *Si $a_0 = 0$ et $a(z)$ admet une réciproque alors $a_1 \neq 0$.*

Preuve. Soit $a(z) = a_r z^r + \cdots$ avec $r \geq 1$ et $a_r \neq 0$, et soit $b(z) = b_s z^s + \cdots$ avec $s \geq 0$ et $b_s \neq 0$ l'inverse de $a(z)$. Alors $a(b(z)) = z = a_r b_s^r z^{rs} + \cdots$ donc $rs = 1$ et par suite $r = s = 1$. □

On définit aussi la dérivation des séries formelles $a'(z) = \sum_{n\geq 1} n a_n z^{n-1}$ qui satisfait les règles de calcul usuelles, et on étend de la même manière d'autres opérations comme l'exponentielle et le logarithme.

Règles de calcul :

Les opérations sur les séries formelles impliquent des opérations sur les coefficients. En particulier, dans le cas de séries convergentes, les opérations sur les fonctions correspondantes se traduisent par certaines opérations sur les coefficients de leurs développements en série.

Soit $a(z) = \sum_{n\geq 0} a_n z^n$; ci-après quelques règles régissant ces relations :

Règle 1. *pour tout entier $k > 0$ on a* $\sum_{n\geq 0} a_{n+k} z^n = \frac{a(z) - a_0 - \cdots - a_{k-1} z^{k-1}}{z^k}$.

Exemple 2.5. La suite de Fibonacci $\{f_n\}$ vérifie la récurrence :

$$f_{n+2} = f_{n+1} + f_n.\ f_0 = 0\ et f_1 = 1$$

alors la fgo correspondante $f(z) = \sum_{n\geq 0} f_n z^n$ vérifie d'après la règle 1, la relation $\frac{f(z)-z}{z^2} = \frac{f(z)}{z} + f(z)$ ce qui donne $f(z) = \frac{z}{1-z-z^2}$.

Règle 2. *On note $D = \frac{d}{dz}$ l'opérateur de dérivation. Soit P un polynôme, alors*

$$P(zD)\,a(z) = \sum_{n\geq 0} P(n) a_n z^n.$$

Exemple 2.6. Trouver la somme des carrés des n premiers entiers positifs.

On remarque que

$$\sum_{k=1}^{n} k^2 = (zD)^2 \left\{ \sum_{k=0}^{n} z^k \right\}_{|z=1} \quad \text{et que} \quad \sum_{k=0}^{n} z^k = \frac{z^{n+1}-1}{z-1}.$$

On retrouve alors la formule $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$.

Règle 3. *Soit k un entier positif, alors*

$$a^k(z) = \sum_{n\geq 0} t_n z^n \;\; où \;\; t_n = \sum_{\substack{(n_1,\cdots,n_k)\in \mathbb{N}^k, \\ n_1 \mid \cdots \mid n_k \mid n}} a_{n_1} \cdots a_{n_k}.$$

Règle 4.

$$\frac{a(z)}{1-z} = \sum_{n\geq 0} s_n z^n \; où \; s_n = \sum_{j=0}^{n} a_j.$$

Exemple 2.7. On veut calculer la fgo des nombres harmoniques $h_n = 1 + \cdots + \frac{1}{n}, n \geq 1$.

Soit $h(z) = \sum_{n\geq 1} h_n z^n$. Par la règle 4 on a $h(z) = \frac{a(z)}{1-z}$ où $a(z) = \sum_{n\geq 1} \frac{1}{n} z^n$

On remarque que $a'(z) = \frac{1}{1-z}$ donc $a(z) = \log\left(\frac{1}{1-z}\right)$, et par suite

$$h(z) = -\frac{\log(1-z)}{1-z}.$$

Pour plus de détails concernant les calculs algébriques sur les séries formelles, et en particulier sur les fonctions génératrices, ainsi que sur les techniques pour les construire, comme le développement en série de McLaurin, la décomposition en éléments simples, la résolution de récurrences ou encore la résolution d'équations fonctionnelles, on peut

consulter (Wilf, 1994; Roberts et Tesman, 2005; Stanley, 2010: Joyal, 1981).

2.2 Constructions admissibles

L'approche symbolique est basée essentiellement sur des constructions particulières, dites admissibles, qui se traduisent directement en fonctions génératrices.

2.2.1 Admissibilité

On considère une construction m-aire Φ qui associe à toute collection de classes combinatoires $\mathscr{B}^{(1)}, \cdots, \mathscr{B}^{(m)}$, une nouvelle classe $\mathscr{A} = \Phi\left(\mathscr{B}^{(1)}, \cdots, \mathscr{B}^{(m)}\right)$.

Soient $(a_n)_{n\geq 0}, \left(b_n^{(1)}\right)_{n\geq 0}, \cdots, \left(b_n^{(m)}\right)_{n\geq 0}$ les suites de décompte associées respectivement aux classes $\mathscr{A}, \mathscr{B}^{(1)}, \cdots, \mathscr{B}^{(m)}$.

Définition 2.6. La construction Φ est dite admissible si et seulement si $(a_n)_{n\geq 0}$ dépend uniquement de $\left(b_n^{(1)}\right)_{n\geq 0}, \cdots \left(b_n^{(m)}\right)_{n\geq 0}$. Autrement dit, aucune information complémentaire, autre que la donnée des suites de décompte, n'est requise pour le calcul de la suite de décompte de la classe construite par Φ.

Pour une telle construction admissible, on peut alors associer un opérateur ϕ, agissant sur les fgo correspondantes, tel que $a(z) = \phi\left(b^{(1)}(z) \cdots, b^{(m)}(z)\right)$.

2.2.2 Constructions de base

On suppose l'existence de deux objets particuliers, un objet de taille 0 noté ε et appelé objet *neutre*, et un objet de taille 1 appelé *atome* et représenté par un symbole quelconque, comme $\bullet$, $\circ$ ou une lettre. On définit alors la classe neutre $\mathscr{E} = \{\varepsilon\}$ et la classe atomique $\mathscr{Z}$ contenant le seul objet atome. Les fgo des deux classes $\mathscr{E}$ et $\mathscr{Z}$ sont respectivement $E(z) = 1$ et $Z(z) = z$.

Produit cartésien ($\times$) :

C'est le produit cartésien usuel muni de la notion naturelle de taille :

$$\mathscr{B}\times\mathscr{C}=\{(\beta,\gamma)/\ \beta\in\mathscr{B},\ \gamma\in\mathscr{C}\}\ avec\ |(\beta.\gamma)|=|\beta|+|\gamma|.$$

Soit $\mathscr{A}=\mathscr{B}\times\mathscr{C}$ alors

$$a(z)=\sum_{\alpha\in\mathscr{A}}z^{|\alpha|}=\sum_{(\beta,\gamma)\in\mathscr{B}\times\mathscr{C}}z^{|\beta|+|\gamma|}=\left(\sum_{\beta\in\mathscr{B}}z^{|\beta|}\right)\times\left(\sum_{\gamma\in\mathscr{C}}z^{|\gamma|}\right)=b(z)c(z)\,.$$

Ce qui montre que le produit cartésien est admissible et que

$$\mathscr{A}=\mathscr{B}\times\mathscr{C}\Longrightarrow a(z)=b(z)c(z).$$

Remarque. Toute classe combinatoire $\mathscr{A}$ vérifie l'isomorphisme $\mathscr{A}\cong\mathscr{E}\times\mathscr{A}\cong\mathscr{A}\times\mathscr{E}$.

Somme combinatoire ($+$) :

La somme combinatoire de deux classes $\mathscr{B}$ et $\mathscr{C}$, appelée aussi union disjointe, est l'union de deux copies disjointes de $\mathscr{B}$ et $\mathscr{C}$. Pour ce faire, on se donne deux objets neutres distincts ε et δ, pour marquer différemment les éléments de $\mathscr{B}$ et ceux de $\mathscr{C}$: $\mathscr{B}\cong\{\varepsilon\}\times\mathscr{B}$ et $\mathscr{C}\cong\{\delta\}\times\mathscr{C}$.

On définit alors la somme combinatoire par l'union usuelle de deux ensembles disjoints :

$$\mathscr{B}+\mathscr{C}=(\{\varepsilon\}\times\mathscr{B})\bigcup(\{\delta\}\times\mathscr{C})\ \text{ avec } |(\varepsilon,\beta)|=|\beta|\,et\,|(\delta,\gamma)|=|\gamma|,\beta\in\mathscr{B},\gamma\in\mathscr{C}.$$

Cette définition est équivalente à l'union usuelle lorsque les deux classes sont disjointes. Il découle de la définition que $si\ \mathscr{A}=\mathscr{B}+\mathscr{C}\ \ alors\ \ a_n=b_n+c_n$, $n\geq 0$, et donc

$a(z) = b(z) + c(z)$. On en déduit que la somme combinatoire est admissible :

$$\mathscr{A} = \mathscr{B} + \mathscr{C} \Longrightarrow a(z) = b(z) + c(z).$$

Remarque. L'union usuelle des ensembles n'est pas admissible car elle dépend de leur intersection. C'est à dire qu'il faut connaître le nombre d'objets communs à ces ensembles, en plus de la donnée de leurs suites de décompte, pour pouvoir calculer la suite de décompte de leur union.

Suites finies (Sequence *Seq*) :

Soit $\mathscr{B}$ une classe combinatoire sans objet vide ($B_0 = \emptyset$). La classe des suites finies de $\mathscr{B}$, notée $Seq(\mathscr{B})$, est définie par la somme infinie suivante :

$$\begin{aligned} Seq(\mathscr{B}) &= \sum_{n\geq 0} \mathscr{B}^n \text{ avec } \mathscr{B}^0 = \{\varepsilon\} \\ &= \{\varepsilon\} \bigcup_{n\geq 1} \{(\beta_1, \cdots, \beta_n), \beta_i \in \mathscr{B} \text{ pour } i = 1, \cdots, n\} \\ &\quad \text{avec } |(\beta_1, \cdots, \beta_n)| = |\beta_1| + \cdots + |\beta_n|. \end{aligned}$$

L'admissibilité découle de celle de la somme et du produit, et on a

$$\mathscr{A} = Seq(\mathscr{B}) \text{ avec } B_0 = \emptyset \Longrightarrow a(z) = \sum_{n\geq 0} b(z)^n = \frac{1}{1 - b(z)}.$$

$a(z)$ est le quasi-inverse de $b(z)$ (voir Exemple 2.4) qui est bien défini puisque $b_0 = 0$.

Sous-ensembles finis (Powerset *Pset*) :

Soit $\mathscr{B}$ une classe combinatoire sans objet vide ($B_0 = \emptyset$). $Pset(\mathscr{B})$ est la classe des sous-ensembles finis de $\mathscr{B}$, telle que la taille de chacun d'eux est égale à la somme des

tailles de ses éléments.

Cette construction est admissible (Flajolet et Sedgewick, 2009) et on a

$$\mathscr{A} = Pset(\mathscr{B}) \Longrightarrow a(z) = \exp\left(\sum_{k\geq 1}(-1)^{k-1}\frac{b(z^k)}{k}\right)$$
$$= \exp\left(\frac{b(z)}{1} - \frac{b(z^2)}{2} + \frac{b(z^3)}{3} - \cdots\right).$$

En effet, dans le cas où la classe $\mathscr{B}$ est finie on peut écrire : $Pset(\mathscr{B}) \cong \prod_{\beta\in\mathscr{B}}(\{\varepsilon\} + \{\beta\})$.

Soit $\mathscr{A} = Pset(\mathscr{B})$ alors

$$a(z) = \prod_{\beta\in\mathscr{B}}\left(1+z^{|\beta|}\right) = \prod_{n=0}^{max\{|\beta|,\beta\in\mathscr{B}\}}(1+z^n)^{b_n}$$
$$= \exp\left(\sum_{n=0}^{max\{|\beta|,\beta\in\mathscr{B}\}} b_n \log(1+z^n)\right) = \exp\left(\sum_{n=0}^{max\{|\beta|,\beta\in\mathscr{B}\}} b_n\left(\sum_{k\ 1}^{\infty}\frac{(-1)^{k-1}}{k}z^{nk}\right)\right)$$
$$= \exp\left(\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}\left(\sum_{n=0}^{max\{|\beta|,\beta\in\mathscr{B}\}} b_n\left(z^k\right)^n\right)\right) = \exp\left(\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}b(z^k)\right).$$

Cette expression s'étend au cas où $\mathscr{B}$ est infini :

soient $\mathscr{B}^{(m)} = \sum_{k=1}^{m} B_k$ et $\mathscr{A}^{(m)} = Pset\left(\mathscr{B}^{(m)}\right)$ pour tout entier $m \geq 1$.

$\mathscr{B}^{(m)}$ étant fini alors $a^{(m)}(z) = \exp\left(\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}b^{(m)}(z^k)\right)$.

D'autre part, en remarquant que pour tout $n \geq 0$, A_n dépend uniquement des B_k avec $k \leq n$, on peut écrire pour tout $m \geq 1$:

$a(z) = a^{(m)}(z) + z^{m+1}f(z)$ et $b(z) = b^{(m)}(z) + z^{m+1}g(z)$ où $f(z)$ et $g(z)$ sont deux séries formelles.

En faisant tendre $m \to \infty$ on déduit que $a^{(m)}(z)$ et $b^{(m)}(z)$ convergent respectivement vers $a(z)$ et $b(z)$, ce qui donne $a(z) = \exp\left(\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}b(z^k)\right)$.

Multi-ensembles finis (Multiset *Mset*) :

Soit $\mathscr{B}$ une classe combinatoire sans objet vide ($B_0 = \emptyset$). Sachant qu'un multi-ensemble permet des répétitions arbitraires de ses éléments, la classe $Mset(\mathscr{B})$ est la collection de tous les multi-ensembles finis d'éléments de $\mathscr{B}$. Cette construction est admissible (Flajolet et Sedgewick, 2009) et on a :

$$\mathscr{A} = Mset(\mathscr{B}) \Longrightarrow a(z) = \exp\left(\sum_{k\geq 1} \frac{b(z^k)}{k}\right) = \exp\left(\frac{b(z)}{1} + \frac{b(z^2)}{2} + \frac{b(z^3)}{3} - \cdots\right).$$

En effet, lorsque $\mathscr{B}$ est finie, avec $B_0 = \emptyset$, on choisit un ordre pour ses éléments, et avec cet ordre chaque multi-ensemble fini d'éléments de $\mathscr{B}$ lui correspond une unique suite finie et ordonnée, avec répétitions possibles, de ces éléments, et vice-versa. On en déduit l'isomorphisme $Mset(\mathscr{B}) \cong \prod_{\beta\in\mathscr{B}} Seq(\{\beta\})$.
Soit $\mathscr{A} = Mset(\mathscr{B})$ alors on peut écrire :

$$\begin{aligned} a(z) &= \prod_{\beta\in\mathscr{B}} \left(1 - z^{|\beta|}\right)^{-1} = \prod_{n\geq 1} (1-z^n)^{-b_n} \\ &= \exp\left(\sum_{n\geq 1} -b_n \log(1-z^n)\right) = \exp\left(\sum_{n\geq 1} b_n \left(\sum_{k\geq 1} \frac{z^{nk}}{k}\right)\right) \\ &= \exp\left(\sum_{k\geq 1} \frac{b(z^k)}{k}\right). \end{aligned}$$

Cette expression reste valide lorsque $\mathscr{B}$ est infinie par le même raisonnement utilisé pour la construction *Pset*.

Les deux constructions *Pset* et *Mset* sont des cas particuliers de la théorie de Pólya (deBruijn, 1964). De même, (Bergeron *et al.*, 1998) présentent des généralisations dans le cadre de la théorie des espèces de Joyal.

Le théorème suivant résume les résultats d'admissibilité des constructions de base pour

les classes d'objets non-étiquetés et les *fgo* associées :

Théorème 2.3 ((Flajolet et Sedgewick, 2009) p.27). *Les constructions somme combinatoire, produit cartésien, suites finies, sous-ensembles finis et multi-ensembles finis sont admissibles. Les opérateurs associés sont comme suit :*

$$\mathscr{A} = \mathscr{B} + \mathscr{C} \Longrightarrow a(z) = b(z) + c(z),$$

$$\mathscr{A} = \mathscr{B} \times \mathscr{C} \Longrightarrow a(z) = b(z)c(z),$$

$$\mathscr{A} = Seq(\mathscr{B}) \text{ et } B_0 = \emptyset \Longrightarrow a(z) = \sum_{n\geq 0} b(z)^n = \frac{1}{1-b(z)}.$$

$$\mathscr{A} = Pset(\mathscr{B}) \text{ et } B_0 = \emptyset \Longrightarrow a(z) = \exp\left(\sum_{k\geq 1}(-1)^{k-1}\frac{b(z^k)}{k}\right),$$

$$\mathscr{A} = Mset(\mathscr{B}) \text{ et } B_0 = \emptyset \Longrightarrow a(z) = \exp\left(\sum_{k\geq 1}\frac{b(z^k)}{k}\right).$$

Il existe d'autres constructions admissibles simples, comme les cycles et celles obtenues en imposant des restrictions sur les constructions précédentes (Flajolet et Sedgewick, 2009), ce qui permet de spécifier un large éventail de classes combinatoires.

2.3 Spécifications combinatoires

Une large variété de classes combinatoires peuvent être décrites par des compositions de constructions de base, permettant ainsi d'en déduire les *fgo* correspondantes. Par suite, l'énumération d'une classe se réduit à lui trouver une *spécification* en termes de constructions admissibles.

Définition 2.7. Une spécification d'un tuple de classes $\left(\mathscr{A}^{(1)}, \cdots . \mathscr{A}^{(k)}\right)$ est une collection de k équations : $\mathscr{A}^{(j)} = \Phi_j\left(\mathscr{A}^{(1)}, \cdots, \mathscr{A}^{(k)}\right)$. $j = 1, \cdots, k$ où Φ_j est une composition de constructions de base, utilisant éventuellement les classes neutre $\mathscr{E}$ et atomique $\mathscr{Z}$.

Remarque. On dit aussi que c'est une spécification de l'une quelconque des classes $\mathscr{A}^{(j)}$. Une classe qui possède une spécification est dite spécifiable ou constructible.

Pour une classe constructible, on peut produire systématiquement des équations fonctionnelles pour sa fgo par le théorème 2.3 sur les constructions de base :

Théorème 2.4 ((Flajolet et Sedgewick, 2009) p.33). *La fgo d'une classe constructible vérifie un système d'équations fonctionnelles formées par les opérateurs suivants :*

$$1\ ,\ z\ ,\ +\ .\ \times\ ,\ Q\ ,\ Exp \text{ et } \overline{Exp} \text{ où}$$

$$Q[f] = \frac{1}{1-f} \text{ quasi-inverse pour } Seq,$$

$$Exp[f] = \exp\left(\sum_{k\geq 1} \frac{f(z^k)}{k}\right) \text{ exp de Pólya pour } Mset,$$

$$\overline{Exp}[f] = \exp\left(\sum_{k\geq 1} (-1)^{k-1} \frac{f(z^k)}{k}\right) \text{ exp de Pólya modifié pour } Pset.$$

Remarque. Si une classe possède une spécification itérative alors sa fonction génératrice se calcule explicitement à partir des opérateurs de base. Alors que si la spécification est récursive, la fonction génératrice est déterminée implicitement comme solution d'équations fonctionnelles.

2.4 Exemples d'application de la méthode symbolique

Exemple 2.8. Soit $\mathscr{W}$ la classe des mots binaires sur l'alphabet $A = \{a, b\}$.
Les lettres a, b sont les atomes de taille 1 et $A \cong \mathscr{Z} + \mathscr{Z}$.
Un mot binaire fini est une suite finie de lettres, donc sa taille est égale à sa longueur et $\mathscr{W} = Seq(\mathscr{Z} + \mathscr{Z})$ donc $\mathscr{W}$ est constructible et sa fgo est $w(z) = \frac{1}{1-2z} = \sum_{n\geq 0} 2^n z^n$.
On retrouve le nombre de mots binaires de taille n qui est $w_n = 2^n$.

Exemple 2.9. On considère la classe $\mathscr{G}$ des arbres planaires (Définition 1.3).

Un arbre planaire est une arborescence pour laquelle les fils de chaque nœud interne sont ordonnés entre eux. Donc $\mathscr{G}$ peut être spécifiée récursivement par $\mathscr{G} = \mathscr{Z} \times Seq(\mathscr{G})$ qui se traduit par l'équation fonctionnelle suivante pour sa fgo : $g(z) = \frac{z}{1-g(z)}$. La résolution de cette équation quadratique donne $g(z) = \frac{1}{2}\left(1 - \sqrt{1-4z}\right)$.
Puis le développement en série entière donne

$$g(z) = \sum_{n\geq 1} \frac{1}{n}\binom{2n-2}{n-1} z^n = z + z^2 + 2z^3 + 5z^4 + 14z^5 + 42z^6 + \cdots .$$

On reconnaît ici les nombres de *Catalan* (OEIS A000108) :

$$g_n = C_{n-1} \text{ pour } n \geq 1 \quad \text{où} \quad C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Considérons maintenant sur la classe $\mathscr{G}$ le paramètre χ qui donne le nombre de feuilles pour chaque arbre planaire, et notons $\mathscr{G}^0 = (\mathscr{G}, \chi)$. La fgo associée à $\mathscr{G}^0$ s'écrit : $g(z,u) = \sum_{n,l\geq 0} g_{n,l} u^l z^n$ où $g_{n,l}$ est le nombre d'arbres planaires de taille n et ayant l feuilles.

À partir de la spécification précédente $\mathscr{G} = \mathscr{Z} \times Seq(\mathscr{G})$, on déduit la spécification suivante de $\mathscr{G}^0$, en distinguant les feuilles des autres sommets :

$$\mathscr{G}^0 = \mathscr{Z}u + \mathscr{Z} \times Seq_{\geq 1}(\mathscr{G}^0).$$

ce qui se traduit par l'équation fonctionnelle : $g(z,u) = zu + \frac{zg(z,u)}{1-g(z,u)}$ $\quad (*)$
En résolvant l'équation quadratique associée à $(*)$, on obtient :

$$g(z.u) = \frac{1}{2}\left(1 + (u-1)z - \sqrt{1 - 2(u+1)z + (u-1)^2 z^2}\right).$$

Le développement en série de $g(z,u)$ par rapport aux deux variables z et u, donne l'expression des termes $g_{n,l}$. Mais on peut aussi utiliser la formule de Lagrange-Bürmann

rappelée ci-après, on peut consulter () pour une démonstration, qui permet de déterminer l'expression des termes $g_{n,l}$ à partir de l'équation $(*)$ de manière plus directe :

Théorème 2.5 (Formule de Lagrange-Bürmann). *Soit $\phi(z) \in \mathbb{C}[[z]]$ telle que $\phi_0 \neq 0$. Alors l'équation $y(z) = z\phi(y(z))$ admet une solution unique $y(z) \in \mathbb{C}[[z]]$ donnée par :*

$$[z^n]\, y^k(z) = \frac{k}{n}\left[y^{n-k}\right]\phi^n(y) \text{ pour tous entiers positifs } n \text{ et } k.$$

Dans notre exemple $\phi(y) = u + \frac{y}{1-y}$ ce qui donne pour $n \geq 2$ et $0 < l < n$:

$$\begin{aligned} g_{n,l} &= \left[u^l\right]\left([z^n]\, g(z,u)\right) = \left[u^l\right]\left(\frac{1}{n}\left[y^{n-1}\right]\left(u+\frac{y}{1-y}\right)^n\right) \\ &= \frac{1}{n}\binom{n}{l}\left[y^{n-1}\right]\left(\frac{y}{1-y}\right)^{n-l} = \frac{1}{n}\binom{n}{l}\binom{n-2}{l-1} = \frac{1}{n-1}\binom{n-1}{l}\binom{n-1}{l-1}. \end{aligned}$$

On remarque que les termes $g_{n,l}$ sont les nombres de *Narayana* $N(n-1,l)$ (OEIS *A*001263).

Exemple 2.10. Dans cet exemple on considère la classe $\mathscr{H}$ des arborescences non-étiquetées. Une arborescence (Définition 1.3) est un arbre enraciné sans structure d'ordre sur les fils d'un même nœud interne. C'est donc une racine à laquelle est attaché un multi-ensemble d'arborescences.
Donc $\mathscr{H}$ est constructible par la spécification récursive : $\mathscr{H} = \mathscr{Z} \times Mset(\mathscr{H})$ qui se traduit par l'équation suivante :

$$h(z) = zExp[h](z) = z\exp\left(h(z) + \frac{h(z^2)}{2} + \cdots\right).$$

Il n'y a pas de solution explicite pour cette équation, mais les termes h_n peuvent être calculés récursivement (OEIS *A*000081) :

$$h(z) = z + z^2 + 2z^3 + 4z^4 + 9z^5 + 20z^6 + 48z^7 + 115z^8 + \cdots.$$

CHAPITRE III

GRAMMAIRES D'OBJETS

Dans ce chapitre, on introduit une méthode d'énumération combinatoire basée sur les grammaires d'objets. Ce sont des descriptions récursives des objets, inspirées de la théorie des langages, et souvent décrites par des schémas. Il s'agit d'une approche similaire à la théorie des espèces de Joyal et à la méthode symbolique de Flajolet, permettant une résolution systématique de nombreux problèmes d'énumération.

3.1 Généralités et définitions

On se donne une famille d'ensembles d'objets $\mathscr{E}$.

Définition 3.1. On appelle opération d'objets dans $\mathscr{E}$ toute application

$$\phi : E_1 \times \cdots \times E_k \longrightarrow E \quad \text{où} \quad k \in \mathbb{N}^* \text{ et } E, E_i \in \mathscr{E} \text{ pour } i = 1, \cdots .k.$$

Une opération d'objets décrit la construction d'un objet à partir d'autres.

Définition 3.2. Un arbre d'opérations d'objets A sur un ensemble $E \in \mathscr{E}$ est un arbre planaire étiqueté, défini récursivement comme suit :

- ou bien A est réduit à une feuille (Définition 1.3) étiquetée par un élément de E,
- ou bien la racine de A est étiquetée par une opération d'objets

$\phi : E_1 \times \cdots \times E_k \to E$ et possède k fils ordonnés, tels que le sous-arbre issu du i-ème fils est un arbre d'opérations d'objets sur E_i pour $i = 1, \cdots, k$.

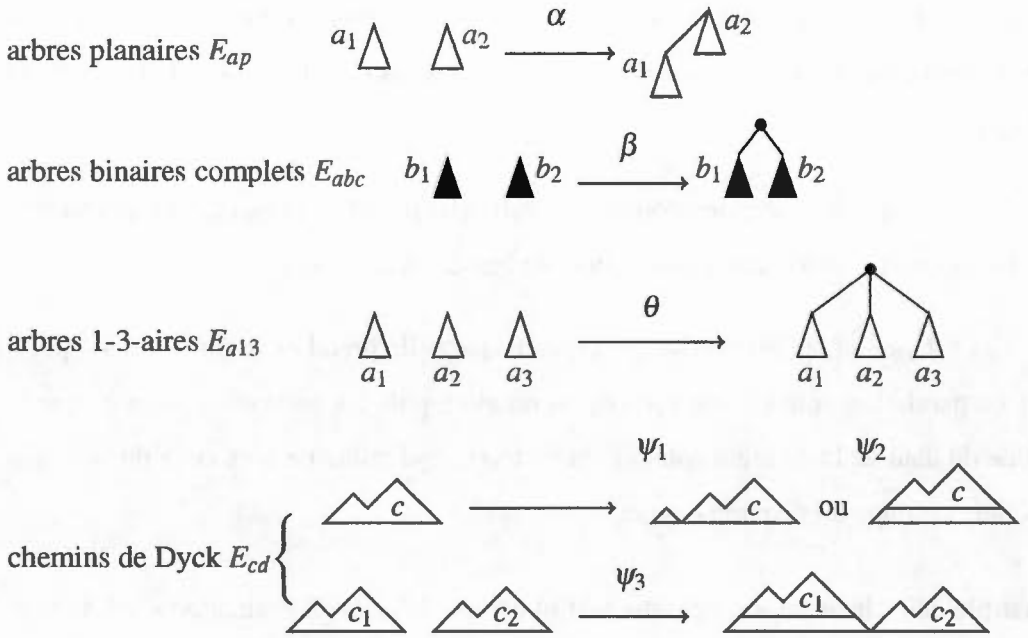
Dans un arbre d'opérations d'objets les nœuds internes sont étiquetés par ces opérations, alors que les feuilles sont étiquetées par des objets.

Définition 3.3. Soit A un arbre d'opérations d'objets sur un ensemble E.

1. On appelle *expression d'objets* de A, et on note $expr(A)$, la concaténation des étiquettes de A dans l'ordre préfixe.
2. On appelle *évaluation* de A, et on note $eval(A)$, l'objet de E défini récursivement :
   - si A est réduit à une feuille étiquetée o, alors $eval(A) = o$.
     On a aussi $expr(A) = o$.
   - sinon, soit ϕ l'étiquette de la racine de A et $A_1, \cdots, A_k$ ses sous-arbres, alors $eval(A) = \phi\,(eval(A_1), \cdots, eval(A_k))$.
     Dans ce cas on a $expr(A) = \phi expr(A_1) \cdots expr(A_k)$.

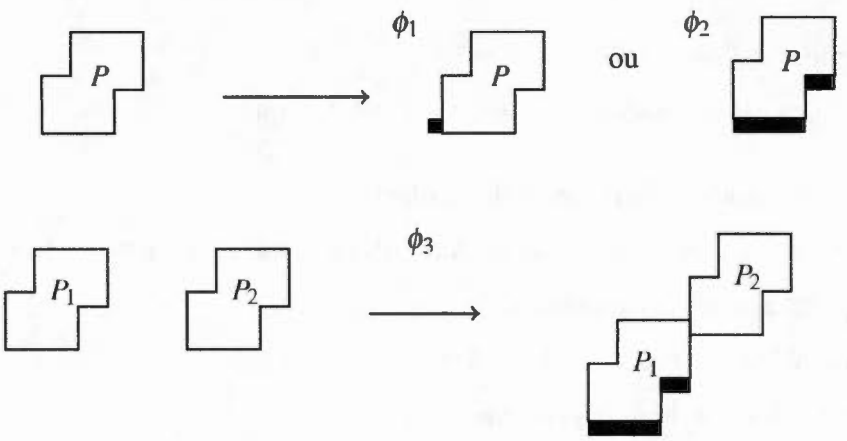
Remarque. L'expression d'un arbre d'opérations d'objets est un mot sur des symboles d'objets et des opérations d'objets, alors que son évaluation est un objet.

Exemple 3.1. Ci-après quelques exemples d'opérations d'objets définies sur certaines classes d'arbres et sur les chemins de Dyck :



Remarque. En général, une opération d'objets peut être définie sur des classes d'objets différentes, et ainsi associer des objets de natures différentes. Mais le plus souvent on se restreint à une seule classe d'objets.

Exemple 3.2. Une opération d'objets sur les polyominos parallélogrammes E_{pp} :
Un polyomino parallélogramme (pp) est défini par un couple de chemins composés uniquement par des pas *Nord* et *Est*, et ne se coupant qu'à leurs extrémités.

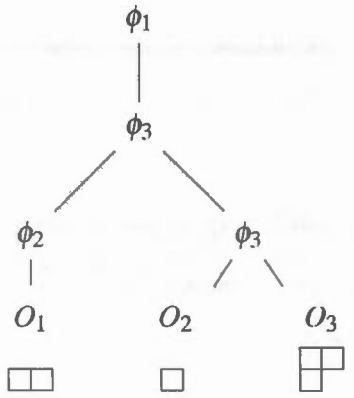


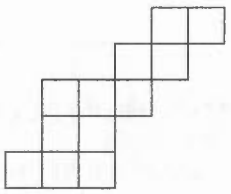
$\phi_1 : E_{pp} \to E_{pp}$, est une extension vers l'ouest. Elle prend en argument un polyomino parallélogramme et lui colle une cellule à l'ouest de la cellule du bas de la première colonne.

$\phi_2 : E_{pp} \to E_{pp}$, est une extension vers le sud. Elle prend en argument un polyomino parallélogramme, et lui ajoute une cellule en bas de chaque colonne.

$\phi_3 : E_{pp} \times E_{pp} \to E_{pp}$, est un collage asymétrique. Elle prend en argument deux polyominos parallélogrammes, applique ϕ_2 au premier, puis les colle de façon à ce que la cellule du haut de la dernière colonne du premier objet coïncide avec celle du bas de la première colonne du deuxième objet.

Exemple 3.3. Un arbre A d'opérations d'objets sur E_{pp} et son évaluation $eval(A)$:





Le pp résultat de $eval(A)$

Définition 3.4. (Dutour et Fedou, 1998)
Une grammaire d'objets est un quadruplet $G =< \mathscr{E}. \mathscr{T}, \mathscr{P}, S >$ où

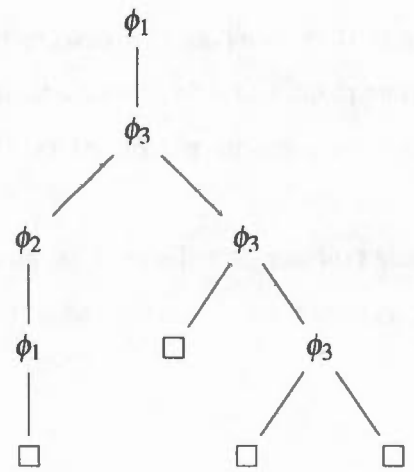
- $\mathscr{E} = \{E_i\}$ est une famille finie d'ensembles d'objets.
- $\mathscr{T} = \{T_{E_i}\}$ est une famille finie de parties finies des ensembles d'objets E_i. Les objets de T_{E_i} sont appelés les terminaux.
- $\mathscr{P}$ est un ensemble d'opérations d'objets dans $\mathscr{E}$.
- S est un élément désigné de $\mathscr{E}$, appelé axiome.

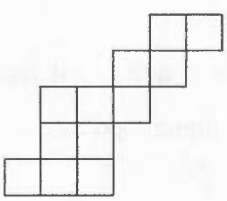
Remarque. Le nombre d'ensembles d'objets E_i de $\mathscr{E}$ est appelé la dimension de la grammaire. Parfois on omet de préciser l'axiome S de la grammaire.

Exemple 3.4. Deux grammaires d'objets de dimension 1 :
$G_1 = \langle \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2\}, E_{pp} \rangle$ et $G_2 = \langle \{E_{pp}\}. \{\{\square\}\}. \{\phi_1, \phi_2, \phi_3\}, E_{pp} \rangle$.

Définition 3.5. Soit une grammaire d'objets $G = \langle \mathscr{E}. \mathscr{T}, \mathscr{P} \rangle$, et soit $E \in \mathscr{E}$.
On appelle *arbre de dérivation* de G sur E, tout arbre d'opérations d'objets sur E dont les nœuds internes sont étiquetés par des opérations d'objets de $\mathscr{P}$, et les feuilles étiquetées par des objets terminaux.

Exemple 3.5. Le polyomino parallélogramme (pp) de l'exemple 3.3 est engendré dans la grammaire G_2 précédente par l'arbre de dérivation suivant :





Le pp engendré par l'arbre de gauche

On note $\mathscr{A}_G$ l'ensemble des arbres de dérivation de G, et E_G l'ensemble des expressions d'objets associées. Un objet est dit engendré dans G par E lorsqu'il est le résultat de l'évaluation d'un arbre de dérivation de G sur E. L'ensemble des objets engendrés dans G par E est un sous-ensemble de E, noté $\mathscr{O}_G(E)$. En particulier si S est l'axiome, on dit que $\mathscr{O}_G(S)$ est l'ensemble d'objets engendré par G.

Remarque. En général $\mathscr{O}_G(E) \subseteq E$ et en particulier $\mathscr{O}_G(S) \subseteq S$.

Définition 3.6. Lorsque $\mathscr{O}_G(E) = E$ pour tout $E \in \mathscr{E}$, on dit que la grammaire est *complète*.

Exemple 3.6. La grammaire G_2 de l'exemple 3.4 est complète, alors que G_1 ne l'est pas. G_2 est complète, en effet :
Soit O un polyomino parallélogramme (*pp*) non réduit à une seule cellule. Si sa première colonne, celle de gauche, contient une seule cellule alors il peut être construit en supprimant cette cellule puis en appliquant ϕ_1 au *pp* qui en résulte. Sinon, sa première colonne contient au moins deux cellules. Dans ce cas on définit pour chaque paire de colonnes adjacentes, la *longueur d'adjacence* qui est le nombre de côtés communs (ou encore le nombre de cellules de l'une collées à l'autre). Il y a alors deux cas possibles :

(i) si l'objet O contient au moins une paire de colonnes adjacentes de longueur d'adjacence égale à 1, on considère la première, et alors on sépare O en deux *pp* contenant chacun l'une des deux colonnes de cette paire. Celui de gauche a toutes ses longueurs d'adjacence ≥ 2, et donc il peut être construit par ϕ_2. Alors, O peut être obtenu par ϕ_3.

(ii) sinon, ou bien que O est réduit à une seule colonne de taille ≥ 2, ou bien que toutes ses longueurs d'adjacence sont ≥ 2, et dans les deux cas il peut être obtenu par ϕ_2.

Donc $\mathscr{O}_{G_2}(E_{pp}) = E_{pp}$.

G_1 n'est pas complète, en effet :
En partant d'une cellule et en appliquant uniquement les opérations ϕ_1 et ϕ_2, la première application de ϕ_2 ajoute une cellule en bas de chaque colonne, y compris la dernière. Donc pour construire un objet O ayant deux colonnes dont la première contient plusieurs cellules et la dernière ne contient qu'une seule, il ne faut pas appliquer ϕ_2. Mais l'application répétée de ϕ_1 construit un objet à une seule ligne, et donc on ne peut pas dériver l'objet O par ϕ_1 et ϕ_2 uniquement : $\mathscr{O}_{G_1}(E_{pp}) \subsetneq E_{pp}$.

Si on ne tient pas compte de l'orientation, alors $\mathcal{O}_{G_1}(E_{pp})$ coïncide avec l'ensemble E_Y des *diagrammes de Young*. lesquels sont des collections finies de cellules, organisées en rangées empilées les unes sur les autres dans l'ordre croissant des longueurs, et alignées à gauche. Ces objets combinatoires jouent un rôle important dans l'étude des représentations des groupes et dans la théorie des fonctions symétriques. En particulier, la suite des longueurs des rangées, ou des colonnes, donne une partition du nombre entier égal à la taille du diagramme.

Définition 3.7. On dit que deux grammaires d'objets sont équivalentes lorsqu'elles engendrent le même ensemble d'objets.

En théorie des langages formels, un langage est dit algébrique lorsqu'il est engendré par une grammaire algébrique, autrement dit, il est reconnu par un automate à pile. Pour des détails sur les langages et les grammaires algébriques. on peut se référer à (Autebert, 1987).

Proposition 3.1. *Pour toute grammaire d'objets G, l'ensemble $\mathcal{X}_G$ des expressions associées aux arbres de dérivation dans G, est un langage algébrique.*

Preuve. On construit une grammaire algébrique Γ qui engendre $\mathcal{X}_G$.
Soit $G = \langle \mathcal{E}, X.\mathcal{R}, S\rangle$ telle que $X = \bigcup_{T\in\mathcal{T}} T \bigcup \{\phi / \phi \in \mathcal{P}\}$ et $\mathcal{R}$ contient les règles suivantes :

$$\begin{cases} E \to \phi E_1 \cdots E_k \text{ . pour tout } \phi \in \mathcal{P},\ \phi : E_1 \times \cdots \times E_k \to E. \\ E \to o \text{ , pour tout } o \in \bigcup_{T\in\mathcal{T}} T. \end{cases}$$

Pour les détails de la démonstration, on pourra consulter (Dutour, 1996). □

3.2 Formes normales

Puisque plusieurs grammaires d'objets peuvent être équivalentes, il est utile de choisir pour chaque classe d'équivalence une grammaire ayant une forme simple et adaptée à la résolution des problèmes. On définit alors trois formes dites *normales* : la forme *réduite*, la forme $1-2$ et la forme *complète*. Les deux premières étendent les notions de formes normales réduite et de Chomsky pour les grammaires algébriques, et la troisième forme est propre aux grammaires d'objets. Ci-après on décrit des transformations de grammaires de formes quelconques en forme normale équivalente.

3.2.1 Forme normale réduite

Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P} \rangle$ une grammaire d'objets.
Un ensemble d'objets $E \in \mathscr{E}$ est dit *productif* lorsqu'il engendre au moins un objet dans G : $\mathscr{O}_G(E) \neq \emptyset$.
Soient $E, F \in \mathscr{E}$. On dit que E est *accessible* à partir de F dans G lorsqu'il existe un arbre de dérivation sur F qui a un nœud étiqueté par une opération d'objets, dont E est l'une des composantes de son domaine de définition. Si E est accessible à partir de l'axiome, on dit qu'il est accessible.

Définition 3.8. G est dite sous la forme normale réduite lorsque tout ensemble d'objets $E \in \mathscr{E}$ est productif et accessible dans G.

Proposition 3.2. *Pour toute grammaire d'objets, on peut construire une grammaire d'objets équivalente sous la forme normale réduite.*

Preuve. La preuve découle des deux lemmes suivants, qui donnent des procédures effectives pour construire, dans une première étape, une grammaire d'objets équivalente dont tous les ensembles d'objets sont productifs, puis dans une deuxième étape, une

autre grammaire d'objets équivalente à la précédente, et dont tous les ensembles d'objets sont accessibles. Elle est donc sous la forme normale réduite. □

Lemme 3.3. *Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P}, S \rangle$ une grammaire d'objets.*
On peut décider pour tout $E \in \mathscr{E}$ si E est productif.

Preuve. On fait la construction suivante :
$\mathscr{E}_0 = \{E \in \mathscr{E} : T_E \neq \emptyset, T_E \in \mathscr{T}\}$
$\mathscr{E}_i = \mathscr{E}_{i-1} \cup \{E \in \mathscr{E} : \exists \phi \in \mathscr{P}, \phi : E_1 \times \cdots \times E_k \to E , E_j \in \mathscr{E}_{i-1} , j = 1, \cdots, k , i \geq 1\}$
$\mathscr{E}_{total} = \bigcup_{i \geq 0} \mathscr{E}_i$.
On peut montrer les deux assertions suivantes, de la même manière que pour les grammaires algébriques (Dutour, 1996) :

1. $\mathscr{E}_{total}$ est la famille des ensembles d'objets de $\mathscr{E}$ qui sont productifs.
2. $\mathscr{E}_{total}$ s'obtient en un nombre borné d'opérations.

Ainsi on a construit une grammaire équivalente $G' = \langle \mathscr{E}', \mathscr{T}', \mathscr{P}' \rangle$ où $\mathscr{E}' = \mathscr{E}_{total}$, $\mathscr{T}' = \{T_{Ej} : E_j \in \mathscr{E}'\}$, $\mathscr{P}' = \{\phi \in \mathscr{P} : \phi : E_1 \times \cdots \times E_k \to E, E, E_j \in \mathscr{E}'\}$.
On dit que G' est *préréduite*. □

Lemme 3.4. *Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P}, S \rangle$ une grammaire d'objets.*
Pour tout $E \in \mathscr{E}$, on peut décider si E est accessible.

Preuve. On fait la construction suivante :
$\mathscr{F}_0 = \{S\}$
$\mathscr{F}_i = \mathscr{F}_{i-1} \cup \{E \in \mathscr{E} : \exists E' \in \mathscr{F}_{i-1}$ et $\phi \in \mathscr{P}$ tels que E est l'une des composantes du domaine de ϕ et E' est sa destination$\}$.
$\mathscr{F}_{total} = \bigcup_{i \geq 0} \mathscr{F}_i$.
On peut montrer les deux assertions suivantes de la même manière que pour les grammaires algébriques (Dutour, 1996) :

1. $\mathscr{F}_{total}$ est la famille des ensembles d'objets accessibles à partir de S.

2. On peut calculer $\mathscr{F}_{total}$ en un nombre borné d'opérations.

Ainsi on a construit une grammaire équivalente $G' = \langle \mathscr{E}', \mathscr{T}', \mathscr{P}', S \rangle$ où $\mathscr{E}' = \mathscr{F}_{total}$, $\mathscr{T}' = \{T_{Ej} : E_j \in \mathscr{E}'\}$, $\mathscr{P}' = \{\phi \in \mathscr{P} : \phi : E_1 \times \cdots \times E_k \to E \ , \ E \in \mathscr{E}'\}$. □

Donc on construit une grammaire équivalente sous la forme normale réduite en enchainant. dans l'ordre, les deux constructions précédentes.

Exemple 3.7. Soit la grammaire d'objets $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P}, S \rangle$ telle que :

$\mathscr{E} = \{E_1.E_2, E_3, E_4\}$, $\mathscr{T} = \{\emptyset, \{o_2\}.\emptyset, \{o_4\}\}$, $S = E_1$ et

$$\mathscr{P} = \begin{cases} \phi_1 : E_2 \to E_1 \ , \ \psi_1 : E_2 \times E_3 \to E_1 \ , \ \theta_1 : E_3 \times E_4 \to E_1 \\ \phi_2 : E_3 \to E_2 \ , \ \psi_2 : E_2 \to E_2 \\ \phi_3 : E_3 \to E_3 \ , \ \psi_3 : E_3 \times E_3 \to E_3 \\ \phi_4 : E_2 \to E_4 \ , \ \psi_4 : E_1 \times E_2 \to E_4 \end{cases}$$

On applique la première construction :

$\mathscr{E}_0 = \{E_2, E_4\}$, $\mathscr{E}_1 = \{E_1, E_2, E_4\} = \mathscr{E}_{total}$. On obtient $G_1 = \langle \mathscr{E}_1, \mathscr{T}_1, \mathscr{P}_1, S \rangle$ où

$\mathscr{T}_1 = \{\emptyset, \{o_2\}, \{o_4\}\}$ et $\mathscr{P}_1 = \begin{cases} \phi_1 : E_2 \to E_1 \\ \psi_2 : E_2 \to E_2 \\ \phi_4 : E_2 \to E_4 \ , \ \psi_4 : E_1 \times E_2 \to E_4 \end{cases}$

Puis on applique la deuxième construction sur la grammaire G_1 :

$\mathscr{F}_0 = \{E_1\}$, $\mathscr{F}_1 = \{E_1, E_2\} = \mathscr{F}_{total}$.

On obtient la grammaire $G_2 = \langle \mathscr{E}_2. \mathscr{T}_2, \mathscr{P}_2, S \rangle$ avec

$\mathscr{E}_2 = \{E_1, E_2\}$, $\mathscr{T}_2 = \{\emptyset, \{o_2\}\}$, $S = E_1$ et $\mathscr{P}_2 = \begin{cases} \phi_1 : E_2 \to E_1 \\ \psi_2 : E_2 \to E_2 \end{cases}$

Alors G_2 est la grammaire d'objets sous la forme normale réduite équivalente à G.

3.2.2 Forme normale 1-2

Cette notion étend aux grammaires d'objets celle de forme normale de Chomsky pour les grammaires algébriques.

Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P}, S \rangle$ une grammaire d'objets.

Définition 3.9. On dit que G est en *forme normale 1-2* si toutes les opérations d'objets de $\mathscr{P}$ sont d'arité 1 ou 2.

Proposition 3.5. *Pour toute grammaire d'objets on peut construire une grammaire d'objets équivalente, sous la forme normale* $1-2$.

Preuve. Il suffit de découper les domaines de définition des opérations d'objets d'arité > 2. La construction est similaire à celle de la forme normale de Chomsky pour les grammaires algébriques (Autebert, 1987).

3.2.3 forme normale complète

L'idée est de se ramener à une grammaire d'objets équivalente, pour laquelle tous les objets sont engendrés. Pour écrire G sous la *forme normale complète*, il suffit de remplacer chaque ensemble d'objets E par $\mathscr{O}_G(E)$ dans $\mathscr{E}$ et dans les définitions des opérations d'objets.

Exemple 3.8. On a vu (exemple 3.6) que la grammaire $G_1 = \langle \{E_{pp}, \{\{\square\}\}\}, \{\phi_1, \phi_2\} \rangle$ n'est pas complète, et qu'elle engendre un sous-ensemble propre de E_{pp}, celui des diagrammes de Young E_Y. Par suite, une grammaire complète équivalente à G_1 est $G'_1 = \langle \{E_Y\}, \{\{\square\}\}, \{\phi_1, \phi_2\} \rangle$ avec ϕ_1 et ϕ_2 restreints à E_Y.

3.3 Non-ambiguïté

La propriété de non-ambiguïté des grammaires d'objets étend celle des grammaires algébriques. Elle est essentielle dès que l'on s'intéresse aux problèmes d'énumération.

Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P}, S \rangle$ une grammaire d'objets.

Définition 3.10. On dit que G est *non-ambiguë* si pour tout ensemble $E \in \mathscr{E}$, tout objet de $\mathscr{O}_G(E)$ est engendré par un unique arbre de dérivation sur E.

Si G est sous la forme normale réduite, alors il suffit de vérifier que tout objet engendré par S dans G lui correspond un unique arbre de dérivation sur S. La proposition suivante donne un critère important pour vérifier la non-ambiguïté d'une grammaire d'objets.

Proposition 3.6. *Une grammaire d'objets G est non-ambiguë si elle vérifie les deux propriétés suivantes :*

(i) toutes les opérations d'objets $\phi \in \mathscr{P}$ sont injectives.

(ii) pour tout $E \in \mathscr{E}$, les opérations de $\mathscr{P}$ de destination E ont des images deux à deux disjointes, et disjointes de l'ensemble $T_E \in \mathscr{T}$ des terminaux de E.

Preuve. Soit $E \in \mathscr{E}$ et soit $o \in \mathscr{O}_G(E)$. Supposons que o est engendré par deux arbres de dérivation A_1 et A_2 : $o = ev(A_1) = ev(A_2)$.
On montre par récurrence sur la hauteur h_1 de A_1, que les arbres A_1 et A_2 sont égaux :
- Si $h_1 = 0$, A_1 et A_2 sont réduits à un sommet étiqueté o, et donc ils sont égaux.
- Soit un entier $h \geq 0$, et supposons que $A_1 = A_2$ dès que $h_1 \leq h$. Soit $h_1 = h + 1$, alors la propriété (ii) implique que les racines des deux arbres sont étiquetées par la même opération d'objets, puis par la propriété (i) on déduit que cette opération est injective et que les sous-arbres de la racine de A_1 ont respectivement les mêmes évaluations que les sous-arbres de la racine de A_2, dans leur ordre. Alors l'hypothèse de récurrence

implique que ces sous-arbres sont deux à deux égaux, dans le même ordre, et donc $A_1 = A_2$.

Remarque. Les deux propriétés de cette proposition forment une condition suffisante pour la non-ambiguïté de G, mais qui n'est pas nécessaire. En effet, pour $E \in \mathscr{E}$ et $\phi \in \mathscr{P}$ de destination E, l'image de ϕ n'est pas obligatoirement incluse dans $\mathscr{O}_G(E)$. Donc certains objets de $Im(\phi)$ peuvent ne pas être engendrés par E, et par suite la vérification des deux propriétés n'est pas nécessaire à l'extérieur de $\mathscr{O}_G(E)$ pour la non-ambiguïté de G. On en déduit la proposition suivante qui affirme que cette condition est nécessaire dans le cas de grammaire réduite et complète.

Proposition 3.7. *Si G est réduite et complète, alors les deux propriétés de la proposition 3.6 forment une condition nécessaire et suffisante à la non-ambiguïté de G.*

Exemple 3.9. La grammaire $G_2 = \langle \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2, \phi_3\} \rangle$ est non-ambiguë. En effet pour $P \in E_{pp}$ on a l'un des cas disjoints suivants :

- ou bien P est le terminal $\square$,
- ou bien $P \in Im(\phi_1)$ lorsque sa première colonne contient une seule cellule,
- ou bien $P \in Im(\phi_2)$ lorsqu'on peut lui enlever une cellule en bas de chacune de ses colonnes,
- ou bien $P \in Im(\phi_3)$ si en lui enlevant une cellule en bas de chaque colonne, de gauche à droite, on arrive à deux colonnes adjacentes entre elles par une seule cellule, alors on décompose P en deux p.p., contenant chacun l'une de ces deux colonnes.

Ainsi on a la partition $E_{pp} = \{\square\} \cup Im(\phi_1) \cup Im(\phi_2) \cup Im(\phi_3)$. Et puisque les opérations ϕ_1, ϕ_2 et ϕ_3 sont injectives, alors G est non-ambiguë.

3.4 Notations équationnelle et schématique

Une grammaire d'objets non-ambiguë et complète peut être décrite d'une manière plus explicite par un système d'équations faisant intervenir des ensembles d'objets, des objets et des opérations d'objets. On peut aussi lui associer une représentation schématique qui reflète de manière visuelle l'action des opérations sur les objets.

3.4.1 Notation équationnelle

Une grammaire complète et non-ambiguë $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P}, S \rangle$ s'écrit comme un système d'équations $\Sigma = \{E_i = p_i(E_1 . \cdots, E_n)\ ,\ i = 1, \cdots, n\}$ où

- $E_i \in \mathscr{E}$.
- p_i est une somme de termes qui peuvent être :
  - soit un objet terminal de T_{E_i},
  - soit une expression de la forme $\phi(E_{i_1}, \cdots, E_{i_k})$ avec $\phi : E_{i_1} \times \cdots \times E_{i_k} \to E_i$

$$p_i(E_1, \cdots, E_n) = \sum_{e_i \in T_{E_i}} e_i + \sum_{dest(\phi) = E_i} \phi(E_{i_1}, \cdots, E_{i_k})$$

où le signe $+$ désigne la somme combinatoire (union disjointe) et $\phi(E_{i_1}, \cdots, E_{i_k})$ est l'image de ϕ.

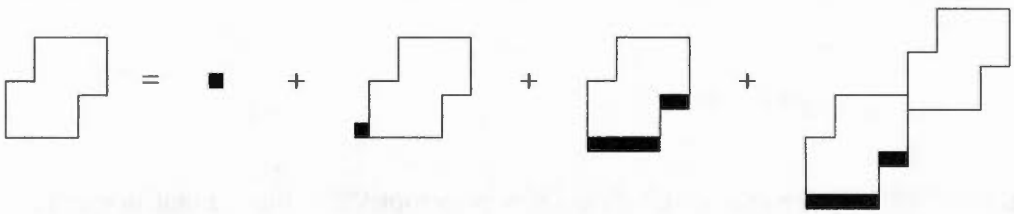
Exemple 3.10. La grammaire $G_2 = \langle \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2, \phi_3\} \rangle$ est complète et non-ambiguë (exemples 3.6 et 3.9), elle s'écrit donc très simplement en notation équationnelle : $E_{pp} = \square + \phi_1(E_{pp}) + \phi_2(E_{pp}) + \phi_3(E_{pp}, E_{pp})$.

3.4.2 Représentation schématique

Une représentation visuelle et intuitive des grammaires d'objets consiste à remplacer dans la notation équationnelle, les objets, ensembles d'objets et opérations d'objets,

par des symboles graphiques. Elle est souvent préférée car elle fournit une description visuelle simple de l'action des opérations d'objets.

Exemple 3.11. Représentation schématique de la grammaire G_2 pour les polyominos parallélogrammes (*pp*) :



3.5 Application à l'énumération

Les grammaires d'objets fournissent un cadre riche dans lequel la résolution de nombreux problèmes d'énumération d'objets combinatoires est grandement simplifiée. Le résultat important à ce sujet est le théorème d'énumération 3.8 ci-après, qui consiste à traduire une grammaire d'objets complète et non-ambiguë en un système d'équations sur les fonctions génératrices des ensembles d'objets.

3.5.1 Théorème d'énumération

Soit $\mathbb{K}$ un corps, $X = \{x_1, \cdots .x_n\}$ un ensemble de variables, $\mathbb{K}[X]$ l'anneau des polynômes et $\mathbb{K}[\![X]\!]$ l'anneau des séries formelles sur $\mathbb{K}$ à variables dans X. Une série formelle $f \in \mathbb{K}[\![X]\!]$ est une expression de la forme $f(x_1, \cdots, x_n) = \sum_{k_1, \cdots .k_n \in \mathbb{N}} a_{k_1, \cdots, k_n} x_1^{k_1} \cdots x_n^{k_n}$ où $a_{k_1, \cdots, k_n} \in \mathbb{K}$ sont les cœfficients de f. On note $\left[x_1^{k_1} \cdots x_n^{k_n}\right] f = a_{k_1, \cdots, k_n}$.

Définition 3.11. Soit E un ensemble d'objets. On appelle *valuation d'objets* sur E toute application $\nu_E : E \to \mathbb{K}[X]$ telle que :

$$\forall\, k_1 . \cdots, k_n \in \mathbb{N},\ \{e \in E :\ \left[x_1^{k_1} \cdots x_n^{k_n}\right] \nu_E(e) \neq 0\} \text{ est fini.}$$

Remarque. La fonction génératrice associée à E est la série formelle $v_E(E) = \sum_{e \in E} v_E(e)$.

Exemple 3.12. Dans la classe E_p des polyominos, il y a un nombre fini d'objets ayant un périmètre donné ou une aire (taille) donnée. On peut donc définir deux valuations d'objets sur E_p, une selon le périmètre et une selon la taille :

$$v_p : E_p \longrightarrow \mathbb{Z}[x] \qquad o \mapsto x^{\text{périmètre}(o)}$$

$$v_a : E_p \longrightarrow \mathbb{Z}[x] \qquad o \mapsto x^{|o|}$$

Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P} \rangle$ une grammaire d'objets. Si pour tout $E \in \mathscr{E}$ on se donne une valuation d'objets v_E sur E, alors on dit que l'ensemble $\mathscr{V} = \{v_E, E \in \mathscr{E}\}$ de ces valuations est associé à G.

Le théorème d'énumération qui suit (Dutour, 1996) p 39, donne une interprétation systématique des grammaires d'objets complètes et non-ambiguës, sous forme d'équations sur les fonctions génératrices des classes d'objets.

Théorème 3.8 (d'énumération). *Si G est complète et non-ambiguë, alors pour tout $E \in \mathscr{E}$, son équation dans G, qui est de la forme*

$$E = \sum_{\delta \in T_E} \delta + \sum_{dest(\phi)=E} \phi(E_{i_1}, \cdots, E_{i_k}) \tag{3.1}$$

se traduit par l'équation

$$v_E(E) = v_E(T_E) + \sum_{dest(\phi)=E} v_E(\phi(E_{i_1}, \cdots, E_{i_k})) \tag{3.2}$$

Preuve. G étant complète et non-ambiguë, alors les termes de l'équation (3.1) sont deux à deux disjoints, et donc l'équation (3.2) en découle directement. □

Exemple 3.13. La grammaire G_2 sur E_{pp} est complète et non-ambiguë (exemples 3.6 et 3.9), et elle s'écrit $E_{pp} = \square + \phi_1(E_{pp}) + \phi_2(E_{pp}) + \phi_3(E_{pp}, E_{pp})$

alors en appliquant la valuation v_p selon le périmètre, on obtient l'équation :

$$v_p(E_{pp}) = v_p(\square) + v_p(\phi_1(E_{pp})) + v_p(\phi_2(E_{pp})) + v_p(\phi_3(E_{pp}, E_{pp})) \tag{3.3}$$

Le paragraphe suivant montre comment résoudre cette équation pour le cas particulier des valuations d'objets *linéaires*, dont v_p fait partie.

3.5.2 Valuations d'objets linéaires

En remarquant que la méthode d'énumération DSV de Schützenberger (Schützenberger, 1963), basée sur les mots, conduit à des équations algébriques sur les fonctions génératrices, et que cette algébricité découle de la linéarité de la valuation, dans ce cas la longueur des mots, par rapport à l'opération de concaténation, alors (Dutour et Fedou, 1998) ont l'idée des valuations linéaires, qui mènent à des équations algébriques sur les fonctions génératrices, qui sont généralement faciles à résoudre.

Considérons des ensembles d'objets E, $E_1, \cdots, E_k$, des valuations d'objets sur ces ensembles v_E, $v_{E_1}, \cdots, v_{E_k}$, et une opération d'objets $\phi : E_1 \times \cdots \times E_k \to E$.

Définition 3.12. On dit que v_E est linéaire par rapport à ϕ, lorsqu'il existe un polynôme $P_\phi \in K[X]$ tel que pour tout $(e_1 . \cdots, e_k) \in E_1 \times \cdots \times E_k$ on a :

$$v_E(\phi(e_1, \cdots, e_k)) = P_\phi \cdot v_{E_1}(e_1) \cdots v_{E_k}(e_k).$$

Remarque. Lorsque ϕ est injective on a : $v_E(\phi(E_1, \cdots, E_k)) = P_\phi \cdot v_{E_1}(E_1) \cdots v_{E_k}(E_k)$.

Exemple 3.14. La valuation d'objets v_p, définie sur E_{pp} selon le périmètre, est linéaire par rapport à l'opération d'objets ϕ_1. En effet, pour tout objet $e \in E_{pp}$ on a
périmètre $(\phi_1(e)) =$ *périmètre*$(e) + 2$, donc $v_p(\phi_1(e)) = x^2 v_p(e)$.
Et comme ϕ_1 est injective, alors $v_p(\phi_1(E_{pp})) = x^2 v_p(E_{pp})$.

On définit maintenant les ensembles de valuations linéaires par rapport à une grammaire d'objets. Soit $G = \langle \mathscr{E}, \mathscr{T}, \mathscr{P} \rangle$ une grammaire d'objets, et soit $\mathscr{V} = \{\nu_E,\ E \in \mathscr{E}\}$ un ensemble de valuations associé à G.

Définition 3.13. On dit que $\mathscr{V}$ est *G-linéaire* si pour tout $E \in \mathscr{E}$, ν_E est linéaire par rapport à toutes les opérations d'objets $\phi \in \mathscr{P}$, dont la destination est E.

Le corollaire suivant découle du théorème d'énumération 3.8 :

Corollaire. *Si G est complète et non-ambiguë et si $\mathscr{V}$ est G-linéaire, alors pour tout $E \in \mathscr{E}$ on a :*

$$E = \sum_{\delta \in T_E} \delta + \sum_{dest(\phi)=E} \phi(E_{i_1}, \cdots, E_{i_k}) \tag{3.4}$$

$$\nu_E(E) = \nu_E(T_E) + \sum_{dest(\phi)=E} P_\phi \nu_{E_{i_1}}(E_{i_1}) \cdots \nu_{E_{i_k}}(E_{i_k}) \tag{3.5}$$

On obtient un système d'équations algébriques où les inconnues sont les fonctions génératrices $\nu_E(E)$ des ensembles d'objets $E \in \mathscr{E}$. En particulier, si S est l'axiome alors $\nu_S(S)$ est la fonction génératrice des objets engendrés par G.

Exemple 3.15. On poursuit l'exemple précédent de la valuation ν_p sur E_{pp}.
On a montré que ν_p est linéaire par rapport à ϕ_1 et que $\nu_p(\phi_1(E_{pp})) = x^2 \nu_p(E_{pp})$.
On montre de la même manière que ν_p est linéaire par rapport à ϕ_2 et ϕ_3, et que :

$$\nu_p(\phi_2(E_{pp})) = x^2 \nu_p(E_{pp}) \quad \text{et} \quad \nu_p(\phi_3(E_{pp}, E_{pp})) = \nu_p(E_{pp})^2 .$$

On remplace ces trois relations dans l'équation (3.3) précédente, qu'on rappelle ici :

$$\nu_p(E_{pp}) = \nu_p(\square) + \nu_p(\phi_1(E_{pp})) + \nu_p(\phi_2(E_{pp})) + \nu_p(\phi_3(E_{pp}, E_{pp})).$$

On note $F = \nu_p(E_{pp})$, on obtient alors l'équation algébrique :

$$F^2 + (2x^2 - 1)F + x^4 = 0.$$

La résolution de cette équation fournit l'expression de la fonction génératrice des polyominos parallélogrammes selon le périmètre :

$$F(x) = \frac{1-2x^2-\sqrt{1-4x^2}}{2} = \sum_{n\geq 1} C_n x^{2n+2}$$

où $C_n = \frac{1}{n+1}\binom{2n}{n}$ sont les nombres de *Catalan*.

CHAPITRE IV

MÉTHODE *ECO*

Dans ce chapitre, on présente la méthode *ECO* (Enumerating Combinatorial Objects) introduite par Renzo Pinzani et son groupe de recherche en informatique théorique dans les années 90 à Florence. Les racines de cette méthode remontent au travail de (Chung *et al.*, 1978) où les auteurs étudient les permutations de Baxter et pour la première fois une construction combinatoire décrite par un arbre de génération est présentée. Mais un algorithme développé par (del Lungo, 1992) pour la génération exhaustive des polyominos dirigés a marqué le début de la méthode *ECO*. Ensuite, plusieurs applications de cette méthodologie ont suivi, en particulier pour l'énumération des chemins de Motzkin k-colorés (Barcucci *et al.*, 1995) et des arbres planaires (Barcucci *et al.*, 1998). Enfin, une formulation générale de la méthode *ECO* est présentée dans (Barcucci *et al.*, 1999), ainsi que plusieurs exemples d'énumération d'objets combinatoires. L'idée principale de cette méthode est de définir des constructions récursives des objets énumérés à partir d'opérateurs d'expansion locale. De telles constructions se traduisent par des équations fonctionnelles, dont la résolution fournit les fonctions génératrices qui énumèrent les objets selon des paramètres appropriés.

Étant donnée une classe d'objets combinatoires $\mathcal{O}$, sur laquelle est défini un paramètre p, considérons le sous-ensemble $O_n = \{x \in \mathcal{O} / p(x) = n\}$. On montre que l'existence d'un opérateur d'expansion locale, vérifiant deux propriétés particulières, permet de

déduire chaque objet $Y \in O_{n+1}$ à partir d'un unique objet $X \in O_n$, et que cette construction récursive des objets de $\mathcal{O}$ se traduit par une équation sur la fonction génératrice associée.

Cette approche d'expansion locale est particulièrement adaptée aux permutations. et plusieurs auteurs l'ont utilisée pour l'énumération de nombreuses classes de permutations à motifs exclus (Gire, 1993; West, 1990; West, 1995). Elle est ensuite étendue par la méthode *ECO* à l'énumération d'autres classes telles que les polyominos, les arbres et les chemins.

La méthode *ECO* permet non seulement d'énumérer des objets, mais aussi de les lister dans un certain ordre, sans répétitions ni omissions. Un algorithme de génération exhaustive a été conçu par (Bacchelli *et al.*, 2004), appliquable aux classes d'objets constructibles par la méthode *ECO*, qui sont alors décrites par un *ECO-système*. Pour les classes exponentielles, cet algorithme garantit un temps de calcul amorti constant (*CAT*) par objet généré.

4.1 Énumération par la méthode *ECO*

4.1.1 Principe de la méthode

Soit $\mathcal{O}$ une classe d'objets combinatoires, et $p : \mathcal{O} \to \mathbb{N}^!$ un paramètre sur $\mathcal{O}$. On considère la famille de sous-ensembles de $\mathcal{O}$: $O_n = \{o \in \mathcal{O} \,/\, p(o) = n\}$, $n \geq 1$

Définition 4.1. On appelle *opérateur* sur $\mathcal{O}$ toute application $\vartheta : \mathcal{O} \to 2^{\mathcal{O}}$ qui fait correspondre à chaque objet de O_n une famille d'objets de O_{n+1} : $\vartheta/O_n : O_n \to 2^{O_{n+1}}$, où $2^{\mathcal{O}}$ désigne l'ensemble des sous-ensembles de $\mathcal{O}$.

Les opérateurs sont définis dans le but de fournir une construction récursive de $\mathcal{O}$. Donc chaque objet de O_{n+1} doit être construit à partir d'un seul objet de O_n. La proposition

suivante donne une condition suffisante pour avoir de tels opérateurs.

Proposition 4.1. *Soit ϑ un opérateur sur $\mathscr{C}$. Si ϑ satisfait les deux conditions suivantes :*

1. *pour tout $Y \in O_{n+1}$, il existe $X \in O_n$ tel que $Y \in \vartheta(X)$.*
2. *pour tous $X_1, X_2 \in O_n$ avec $X_1 \neq X_2$, on a $\vartheta(X_1) \cap \vartheta(X_2) = \emptyset$.*

alors la famille d'ensembles $\mathscr{F}_{n+1} = \{\vartheta(X),\ X \in O_n\}$ est une partition de O_{n+1}.

Preuve. D'après la définition d'un opérateur, pour tout $X \in O_n$, $\vartheta(X) \subset O_{n+1}$, donc $\bigcup_{X \in O_n} \vartheta(X) \subset O_{n+1}$.
D'autre part, soit $Y \in O_{n+1}$ alors d'après la condition (1) il existe $X \in O_n$ tel que $Y \in \vartheta(X)$, donc $O_{n+1} \subset \bigcup_{X \in O_n} \vartheta(X)$, et parsuite on a $O_{n+1} = \bigcup_{X \in O_n} \vartheta(X)$.
Enfin la condition (2) exige que les ensembles $\vartheta(X)$ pour $X \in O_n$ sont deux à deux disjoints, ce qui montre que la famille $\mathscr{F}_{n+1}$ est une partition de O_{n+1}. □

Un opérateur qui satisfait les conditions de la proposition 4.1 est dit *valide*. Il fournit une description récursive de la classe $\mathscr{C}$, qui se traduit le plus souvent par une équation fonctionnelle vérifiée par la fonction génératrice correspondante. On trouve dans la littérature des méthodes et des techniques variées pour la résolution de telles équations, on peut par exemple se référer à (Banderier *et al.*, 2002).

4.1.2 Arbre de génération

Considérons une classe $\mathscr{C}$ paramétrée par la taille, et supposons qu'elle possède un seul objet de taille minimale 1. Alors on peut représenter la construction récursive de $\mathscr{C}$ par un arbre enraciné dont les sommets du *n-ième* niveau représentent les objets de O_n. La racine de l'arbre est l'objet minimal, et chaque objet $Y \in O_{n+1}$ est l'enfant de l'objet $X \in O_n$ tel que $Y \in \vartheta(X)$. Cet arbre est appelé *arbre de génération* de $\mathscr{C}$ associé à l'opérateur ϑ.

Les arbres de génération ont été utilisés pour la première fois dans (Chung *et al.*, 1978) pour l'énumération des permutations de *Baxter*. Ils ont été aussi utilisés pour établir des bijections entre des classes ayant des arbres de génération isomorphes (Gire, 1993).

4.1.3 Règles de succession

Un moyen simple et pratique pour décrire les constructions récursives, et donc la structure des arbres de génération, est la notion de *règles de succession* ou de *réécriture*. Chaque sommet de l'arbre de génération est étiqueté par une suite de symboles qui encapsule l'information sur le nombre de ses fils. Puis, à chaque étiquette on fait correspondre la suite ordonnée des étiquettes des fils correspondants. Cette correspondance s'appelle règle de réécriture, et on l'écrit comme suit :

$$RR: \ (p) \hookrightarrow (c_1)\cdots(c_k).$$

Plusieurs classes d'objets combinatoires ont des arbres de génération complètement caractérisés par une seule règle de réécriture. et même l'étiquette (p) d'un nœud est égale au nombre k de ses fils. Pour d'autres classes, la structure de l'arbre de génération peut être complètement déterminée par un système de règles de réécriture. Il faut noter qu'un même système *RR* peut décrire des constructions récursives de différentes classes d'objets, qui sont en bijection. La famille des classes associées à *RR* est appelée l'*espace combinatoire* de *RR*.

4.1.4 Exemples

On donne deux exemples d'application de la méthode *ECO*, un pour l'énumération des arbres planaires finis et l'autre pour l'énumération des polyominos convexes.

Les éléments unitaires de ces objets sont les sommets pour les arbres et les cellules

pour les polyominos. La première étape de la méthode consiste à définir l'opérateur d'expansion locale ϑ , qui produit un objet $Y \in O_{n+1}$ à partir d'un objet $X \in O_n$, en insérant à X un élément unitaire dans un certain emplacement admissible, où O_n est l'ensemble des objets énumérés de taille n.
Pour avoir une construction complète et sans redondance, et ainsi garantir que l'opérateur ϑ satisfait la condition de la proposition 4.1, il revient à déterminer pour chaque objet les emplacements à considérer parmi les emplacements admissibles, appelés emplacements *actifs*.

Arbres planaires

La classe des arbres planaires finis (définition 1.3) fait partie de l'espace combinatoire de *Catalan*, dans lequel on trouve aussi les chemins de *Dyck* et les polyominos parallélogrammes (Barcucci *et al.*, 1999).

Un arbre planaire non vide est défini récursivement par une partition de l'ensemble de ses sommets en une suite ordonnée d'arbres planaires non vides, dont le premier est réduit à la racine. On donnera une construction récursive de cette classe, décrite par la règle de succession : $(k) \hookrightarrow (2)\cdots(k+1)$, ce qui permettra de lui appliquer la méthode *ECO*.

Soit Υ la classe des arbres planaires finis et non étiquetés, paramétrée par la taille, et soit $A \in \Upsilon$. On note $\ell ast(A)$ le dernier sommet de A dans un parcours préfixe, et on appelle *branche de droite* de A le chemin reliant la racine à $\ell ast(A)$.

Opérateur d'expansion locale :

Si $A \in \Upsilon_n$ on peut l'étendre en un arbre $A' \in \Upsilon_{n+1}$, en lui attachant une feuille sur un sommet quelconque, dans la position la plus à droite, telle que cette feuille soit le fils le

plus à droite de son parent. On prend alors pour emplacements admissibles l'ensemble de toutes ces positions, et on a ainsi un emplacement admissible pour chaque sommet. Réciproquement, pour tout arbre $B' \in \Upsilon_{n+1}$, on a $\ell ast(B')$ est une feuille attachée à un emplacement admissible, et en le supprimant on obtient un arbre $B \in \Upsilon_n$. Donc on considère l'opérateur ϑ qui consiste à insérer une feuille dans l'un quelconque d'une sélection particulière d'emplacements admissibles de l'arbre, faisant de ϑ un opérateur valide. Cette sélection est l'ensemble des emplacements actifs, et la proposition suivante nous donne une solution :

Proposition 4.2. *Pour un arbre planaire fini, si on prend les sommets de la branche de droite comme emplacements actifs, alors l'opérateur ϑ défini précédemment sur la classe Υ est valide.*

Preuve. Chaque arbre $Y \in \Upsilon_{n+1}$ possède un unique antécédent $X \in \Upsilon_n$, obtenu en enlevant la feuille $\ell ast(Y)$, ce qui montre que ϑ est valide. □

La figure 4.1 montre un arbre planaire avec ses successeurs par l'opérateur ϑ.

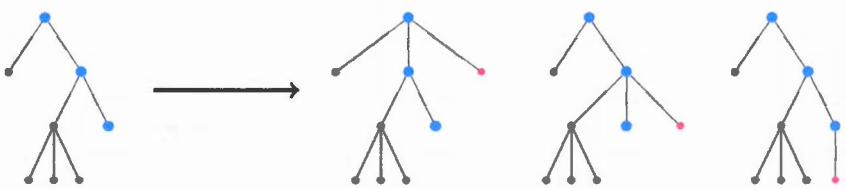


Figure 4.1: Un arbre planaire et ses successeurs par ϑ

L'opérateur ϑ définit une construction récursive de Υ, et une partie de l'arbre de génération correspondant est représentée à la figure 4.2. La racine étant l'arbre planaire de taille 1.

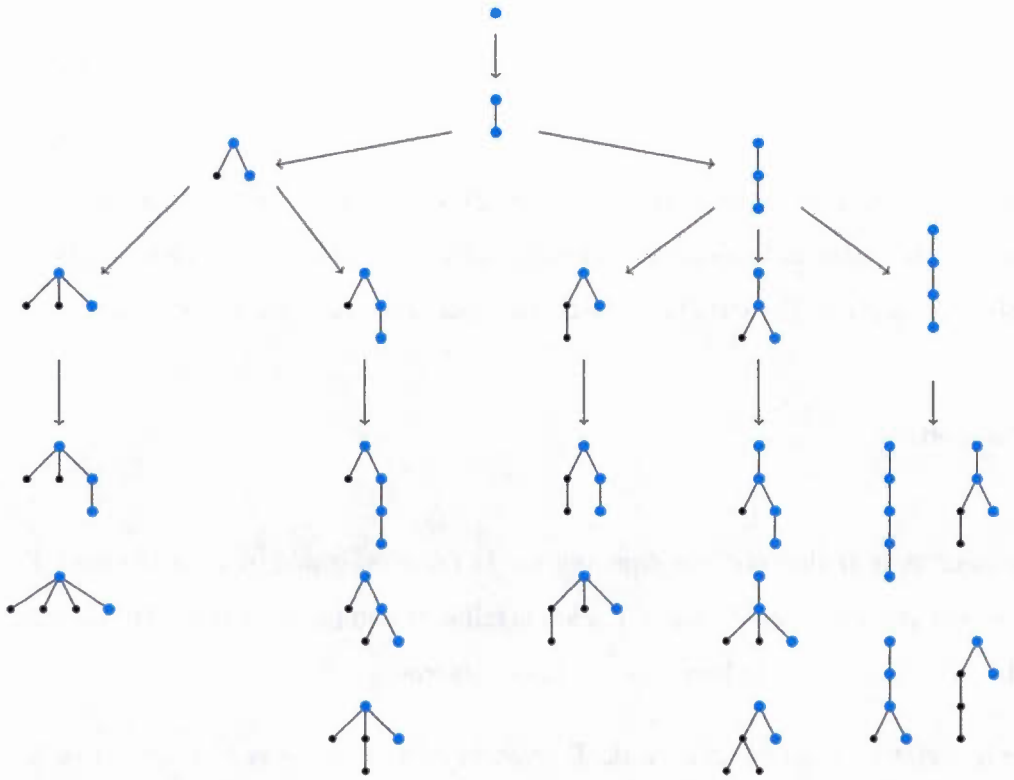


Figure 4.2: Arbre de génération des arbres planaires associé à ϑ

Règle de succession :

Soit A un arbre planaire. Le nombre d'emplacements actifs de A est égal au nombre de sommets de la branche de droite, soit k ce nombre. Alors $\vartheta(A)$ contient k arbres planaires, fils de A dans l'arbre de génération. De plus, les longueurs des branches de droite de ces k arbres sont respectivement $2,3,\cdots,k+1$ par définition de ϑ, et il en est de même pour les nombres des emplacements actifs correspondants. La proposition suivante résume cette propriété de la structure de l'arbre de génération, qu'on peut vérifier sur la figure 4.2.

Proposition 4.3. *L'arbre de génération des arbres planaires finis est isomorphe à l'arbre dont la racine est étiquetée par* (1)*, et défini récursivement par la règle de*

succession $(k) \hookrightarrow (2)\cdots(k+1)$.

On peut définir d'autres opérateurs valides, basés sur des parcours hiérarchiques des arbres planaires, tels que le parcours en largeur. On obtient alors d'autres constructions récursives de la classe Υ, mais elles sont décrites par la même règle de succession.

Énumération :

L'opérateur ϑ se traduit par une équation sur la fonction génératrice de la classe Υ. (Barcucci *et al.*, 1999) ont énuméré Υ selon la taille, le nombre de feuilles, la longueur de la branche de droite et la longueur du chemin interne.

On se restreint ici à l'énumération de Υ selon la taille $n = |A|$ et la longueur de la branche de droite $r(A)$ pour tout $A \in \Upsilon$, alors sa fonction génératrice s'écrit :
$F(s,x) = \sum_{A\in\Upsilon} s^{r(A)}x^n$.

Soit $A \in \Upsilon$ et $A' \in \vartheta(A)$. A' s'obtient en insérant une feuille à l'un des emplacements actifs, ce qui revient à l'attacher à l'un des sommets $e_1,\cdots,e_{r(A)}$ de la branche de droite de l'arbre A, où e_1 est la racine et e_k est le sommet de niveau k, $k = 1,\cdots,r(A)$.
On a $|A'| = n+1$. D'autre part, soit e_k le sommet qui a donné A', alors $r(A') = k+1$.
L'opérateur ϑ étant valide, donc $\vartheta(\Upsilon_n) = \Upsilon_{n+1}$ pour tout $n \geq 1$, et alors on a
$\vartheta(\Upsilon) = \bigcup_{n\geq 1} \vartheta(\Upsilon_n) = \bigcup_{n\geq 2} \vartheta(\Upsilon_n)$ ou encore $\Upsilon = \Upsilon_1 \cup \vartheta(\Upsilon)$.
La fgo de $\vartheta(\Upsilon)$ s'écrit :

$$\sum_{A\in\Upsilon}\sum_{A'\in\vartheta(A)} s^{r(A')}x^{|A'|} = \sum_{A\in\Upsilon}\sum_{k=1}^{r(A)} s^{k+1}x^{n+1} = \frac{s^2x}{1-s}\left(F(1,x)-F(s,x)\right).$$

Il faut remarquer que $F(1,x) = \sum_{A\in\Upsilon} x^n$ est la fgo de Υ selon la taille seulement.

On en déduit l'équation vérifiée par F :

$$F(s,x) = sx + \frac{s^2 x}{1-s}\left(F(1,x) - F(s,x)\right),$$

ou encore

$$(xs^2 - s + 1)F(s,x) = sx(1 - s + sF(1,x)).$$

si $xs^2 - s + 1 = 0$ alors $s = s_0(x) = \frac{1-\sqrt{1-4x}}{2x}$ et parsuite $F(1,x) = \frac{s_0 - 1}{s_0} = xs_0 = \frac{1-\sqrt{1-4x}}{2}$.
On retrouve la fonction génératrice des nombres de *Catalan.*

Corollaire. *Le nombre d'arbres planaires de taille $n+1$ est égal au n-ième nombre de Catalan* $C_n = \frac{1}{n+1}\binom{2n}{n}$.

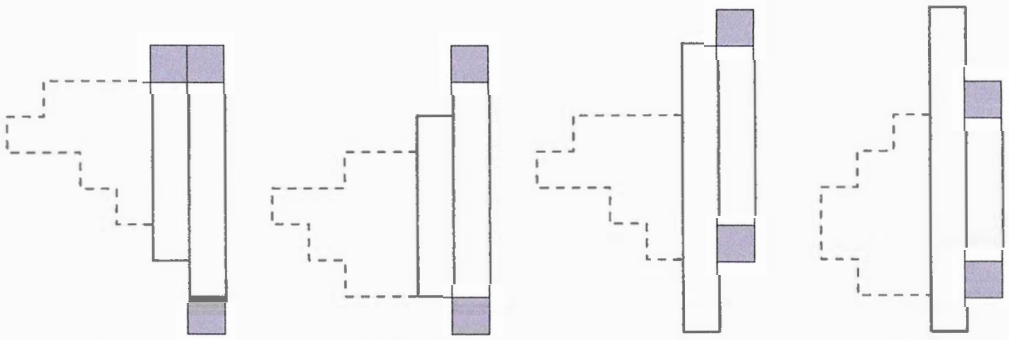
Polyominos convexes

Dans cet exemple on définit un opérateur *ECO* qui décrit une construction récursive de la classe $\mathscr{C}$ des polyominos convexes (section 1.1.2).

On partitionne $\mathscr{C}$ en quatre sous-ensembles disjoints : C_b, C_a, C_r et C_g.

1. C_b contient les polyominos convexes ayant au moins deux colonnes, et tels que :
   (a) les deux cellules les plus en haut des deux colonnes les plus à droite,ainsi que d'autres cellules éventuellement, ont la même ordonnée qui est maximale à travers le polyomino ;
   (b) la cellule la plus en bas de la colonne la plus à droite,parmi d'autres cellules éventuellement, a l'ordonnée minimale à travers le polyomino.
2. C_a est l'ensemble des polyominos convexes tels que :
   (a) la cellule la plus en haut de la colonne la plus à droite, et seulement elle, a l'ordonnée maximale parmi toutes les cellules ;
   (b) la cellule la plus en bas de la colonne la plus à droite, possiblement avec d'autres cellules, a l'ordonnée minimale parmi toutes les cellules.

3. C_r est l'ensemble des polyominos convexes pour lesquels, soit la cellule la plus en haut de la colonne la plus à droite a l'ordonnée maximale, soit la cellule la plus en bas de la colonne la plus à droite a l'ordonnée minimale, mais pas les deux en même temps.
4. C_g contient les polyominos convexes pour lesquels les ordonnées des cellules la plus en haut et la plus en bas de la colonne la plus à droite ne sont ni maximale ni minimale à travers le polyomino.



(a) C_b (b) C_a (c) C_r (d) C_g

Figure 4.3: Partition de la classe des polyominos convexes

Opérateur :

La classe $\mathscr{C}$ est paramétrée par le demi-périmètre, somme des nombres des lignes et des colonnes. On considère l'opérateur ϑ qui effectue des expansions locales sur la colonne la plus à droite du polyomino convexe.
Soit $P \in \mathscr{C}$ de demi-périmètre $n+2$, et ayant k cellules dans sa colonne la plus à droite, l'opérateur ϑ effectue les transformations suivantes :

Pour $i = 1, \cdots, k$, on colle à la colonne la plus à droite, entre ses deux extrêmités, une colonne de longueur i. Il y a $k-i+1$ possibilités, et donc cette opération produit $1+2+\cdots+k$ polyominos de demi-périmètre $n+3$.

En plus des opérations précédentes, ϑ effectue des opérations supplémentaires selon la sous-classe du polyomino :

- si $P \in C_b$, alors ϑ prolonge sa colonne la plus à droite d'une seule cellule, une fois par le haut et une fois en bas, ce qui produit deux polyominos ;
- si $P \in C_a$, alors ϑ prolonge sa colonne la plus à droite d'une seule cellule par le haut ;
- si $P \in C_r$, alors ϑ prolonge sa colonne la plus à droite d'une seule cellule par le côté de l'extrémité dont l'ordonnée est maximale ou minimale ;
- l'opérateur ϑ n'effectue aucune opération supplémentaire pour C_g.

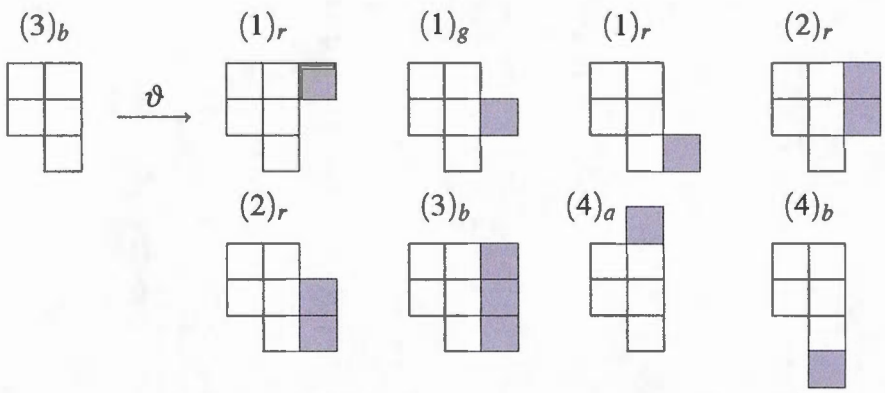


Figure 4.4: L'opérateur ϑ pour la classe C_b

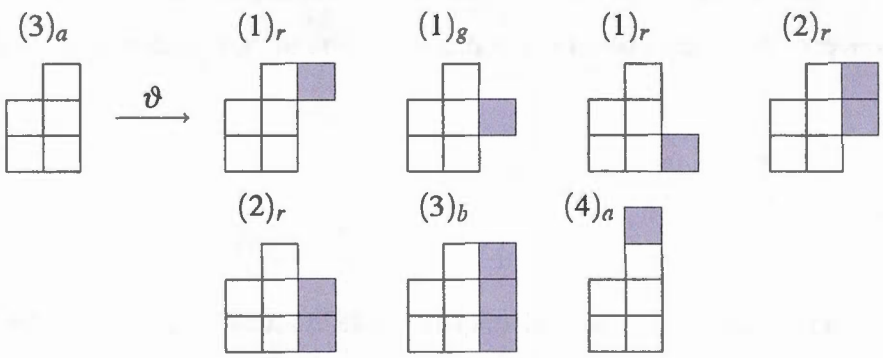


Figure 4.5: L'opérateur ϑ pour la classe C_a

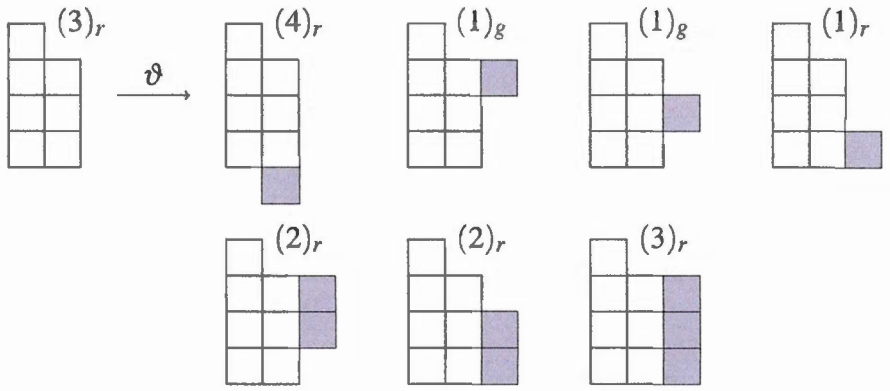


Figure 4.6: L'opérateur ϑ pour la classe C_r

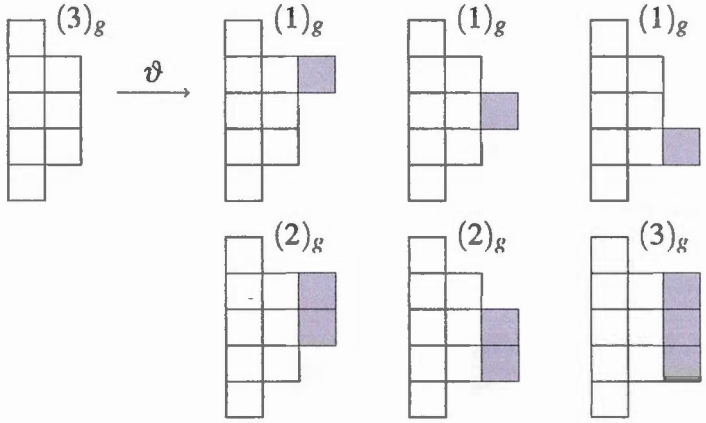


Figure 4.7: L'opérateur ϑ pour la classe C_g

On vérifie pour tous les cas possibles, selon la colonne la plus à droite, que tout polyomino convexe peut être construit par ϑ d'une manière unique, et donc ϑ est valide.

Règle de succession :

La définition de l'opérateur ϑ dépend uniquement de la sous-classe C_α , $\alpha \in \{b.a, r, g\}$ du polyomino convexe considéré, et de la longueur k de sa colonne la plus à droite. Donc on le représente par l'étiquette $(k)_i$. On prend pour axiome $(1)_a$, et on peut

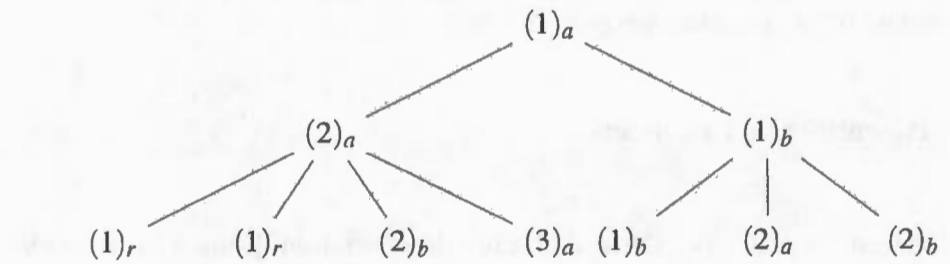
vérifier (Barcucci *et al.*, 1999) que les productions de ϑ sont comme suit :

$$\Gamma \begin{cases} (k)_g \hookrightarrow \prod_{j=1}^{k} (j)_g^{k-j+1} \\ (k)_r \hookrightarrow \prod_{j=1}^{k-1} (j)_g^{k-j} \prod_{j=1}^{k+1} (j)_r \\ (k)_a \hookrightarrow \prod_{j=1}^{k-2} (j)_g^{k-j-1} \prod_{j=1}^{k-1} (j)_r^2 (k)_b (k+1)_a \\ (k)_b \hookrightarrow \prod_{j=1}^{k-2} (j)_g^{k-j-1} \prod_{j=1}^{k-1} (j)_r^2 (k)_b (k+1)_a (k+1)_b \end{cases}$$

où l'exponentiation est utilisée pour représenter les répétitions.

Donc la règle de succession de l'opérateur ϑ est donnée par les productions Γ et l'axiome $(1)_a$. L'arbre de génération correspondant est construit de la manière suivante :

- la racine est étiquetée par $(1)_a$.
- chaque nœud étiqueté par $(k)_i$ produit les fils dont les étiquettes sont données par la règle de succession Γ.



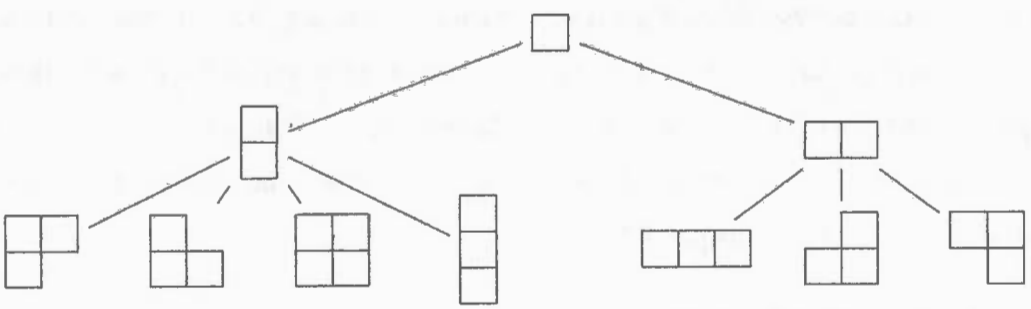


Figure 4.8: Arbre de génération des polyominos convexes associé à l'opérateur ϑ

Énumération :

La construction donnée par l'opérateur ϑ, et par la règle de succession associée, se traduit par des équations dont la solution est la fonction génératrice des polyominos convexes selon le demi-périmètre. Pour les détails de la résolution on peut se référer à (del Lungo *et al.*, 2004). La solution obtenue est donnée par :

$$f(x) = \frac{x^2(1-6x+11x^2-4x^3-4x^2\sqrt{1-4x})}{(1-4x)^2}.$$

4.2 Génération exhaustive par la méthode *ECO*

On présente un algorithme pour la génération exhaustive des objets d'une classe combinatoire, basé sur les règles de succession et adapté à un grand nombre de classes (Bacchelli *et al.*, 2004). Cet algorithme est efficace dans le sens qu'il coûte un temps amorti constant (*CAT*) par objet généré.

4.2.1 Présentation de l'algorithme

L'algorithme est un parcours préfixe de l'arbre de génération, jusqu'à la profondeur désirée. Un objet du n-ieme niveau peut être représenté par le mot $w = w_1 \cdots w_n$, qui code le chemin correspondant à partir de la racine, et dont les étiquettes sont données par la règle de succession. w_1 est donné par l'axiome, et w_i est un fils de w_{i-1}, donc il prend l'une des valeurs données par la production (w_{i-1}). Dans une instance du mot w, on dit que le terme w_i est *modifiable* lorsque sa valeur actuelle n'est pas la dernière parmi la liste de ses valeurs possibles.

L'algorithme se déroule comme suit :

(i) Il commence par initialiser w, de manière à ce que chaque terme w_i , $i > 1$ prenne

la première valeur engendrée par la production (w_{i-1}).

(ii) Ensuite, il répète l'itération suivante :

- trouver le terme modifiable w_i d'indice le plus grand possible, lui donner la valeur qui succède sa valeur actuelle dans la production (w_{i-1}), puis initialiser successivement les termes w_j , $j = i+1, \cdots, n$ à la première valeur de la production (w_{j-1}).

(iii) L'algorithe se termine quand il n'y a plus de terme modifiable dans w.

Algorithme :

1. Lire la règle de succession Γ et la profondeur de génération n.
2. Initialiser le mot $w = w_1 \cdots w_n$ comme suit :
   a) w_1 prend la valeur de l'axiome ;
   b) initialiser successivement w_j , $j = 2. \cdots, n$, à la première valeur, donnée par la production (w_{j-1}) .
3. Tant qu'il y a un terme modifiable dans w faire
   a) trouver le terme w_i modifiable ayant le plus grand indice.
   b) mettre à jour w_i et initialiser successivement w_j , $j = i+1, \cdots, n$.

4.2.2 Analyse de la complexité

Cet algorithme a été conçu par (Bacchelli *et al.*, 2004) qui ont montré que son temps de calcul est proportionnel au nombre d'objets générés. Il est dit de type *CAT* (de l'anglais constant amortized time). Elles ont montré qu'il est efficace pour les classes d'objets exponentielles, dans le sens que le coût amorti moyen par objet est constant.

Soit C_n le coût de génération des $|O_n|$ objets de taille n.
Chaque objet de taille n se déduit d'un objet de taille $n-1$ en temps constant, donc $C_n = C_{n-1} + |O_n|$ et $C_1 = 0$, et par suite $\frac{C_n}{|O_n|} \leq \frac{|O_2|+\cdots+|O_n|}{|O_n|}$.

Lorsque $|O_n|$ est exponentiel, $|O_n| \propto a^n$, alors $\frac{C_n}{|O_n|}$ reste borné.

Remarque. Un algorithme de génération exhaustive doit avoir la propriété *CAT* pour être considéré efficace. Dans la preuve de l'efficacité de l'algorithme présenté, il a été supposé que la classe d'objets est exponentielle, mais on ne connaît pas à l'heure actuelle une condition nécessaire et suffisante pour qu'un *ECO-système* donne un algorithme ayant la propriété *CAT*.

CHAPITRE V

GÉNÉRATION EXHAUSTIVE DES POLYCUBES-ARBRES

Ce chapitre présente un travail expérimental pour la génération exhaustive des polycubes-arbres multidimensionnels (Blondin-Massé *et al.*, 2017) sous la forme libre, une sous-classe des polycubes du réseau hypercubique $\mathbb{Z}^d$. Un arbre de génération de ces objets est élaboré en s'inspirant de la méthode *ECO*.

D'abord, pour chaque polycube-arbre fixe, on construit un arbre planaire étiqueté et ordonné qui le représente de manière unique, puis on définit un encodage de ces arbres par des chaînes de caractères. Cet encodage est utilisé pour définir une représentation canonique des polycubes-arbres libres, ce qui permet un stockage économique en mémoire des objets générés.

Par la suite, on définit la structure de l'arbre de génération exhaustive par une procédure d'expansion locale basée sur un parcours en largeur de l'arbre canonique correspondant à l'objet générateur.

Enfin, une implémentation Java est réalisée pour les dimensions 2, 3 et 4, et exécutée sur un ordinateur portable ordinaire de 12Gb de RAM, et doté d'un processeur core i5 de 2GHz. Les résultats obtenus sont résumés à la fin du chapitre.

5.1 Définitions

Un polycube d-dimensionnel (définition 1.2) est un ensemble fini connexe d'hypercubes unitaires de $\mathbb{Z}^d$. Autrement dit, c'est un sous-graphe induit fini et connexe du graphe dual d'adjacence des cellules de $\mathbb{Z}^d$. Si de plus le sous-graphe induit est sans cycle, alors il est appelé *polycube-arbre*. Étant des polycubes particuliers, les polycubes-arbres sont considérés *fixes* ou *libres*.

Une *feuille* d'un polycube-arbre est toute cellule ayant une seule cellule adjacente, et l'ensemble de toutes ses feuilles est appelé son *feuillage*. Le nombre maximal de feuilles que peut avoir un polycube-arbre d-dimensionnel de taille n est noté $L_d(n)$. (Blondin-Massé *et al.*, 2017) ont donné des formules récursives pour $L_2(n)$ et $L_3(n)$, alors que $L_d(n)$ pour $d > 3$ reste inconnu à ce jour. Les d-polycubes-arbres dont le nombre de feuilles est maximal sont dits *pleinement feuillus*.

(a) Polyomino-arbre (b) $3d$-Polycube-arbre

Figure 5.1: Polycubes-arbres

5.2 État de l'art

Comme pour les polycubes d-dimensionnels, l'énumération des polycubes-arbres reste encore un problème ouvert. Mais, l'amélioration continue des algorithmes et de la puissance de calcul permettent de les dénombrer pour des tailles de plus en plus grandes.

Soient $T_d(n)$ et $t_d(n)$ les nombres de polycubes-arbres de taille n respectivement fixes et libres. Leur *taux d'accroissement limite* s'écrit :

$$\lambda_d^t = \lim_{n\to\infty} \frac{T_d(n+1)}{T_d(n)} = \lim_{n\to\infty} \frac{t_d(n+1)}{t_d(n)}.$$

Les derniers termes calculés à l'heure actuelle pour les dimensions 2, 3 et 4 sont : $t_2(17)$ par David Radcliffe 2017 (OEIS A131482), $T_2(44)$ par Iwan Jensen 2000 (OEIS A066158), $T_3(17)$ et $T_4(10)$ par Gill Barequet 2011 (OEIS A118356 et A191094). Quelques termes sont aussi calculés pour d'autres dimensions.

(Aleksandrowicz et Barequet, 2011) ont établi certains résultats asymptotiques pour λ_d^t. Ils ont estimé et conjecturé que $\lambda_d^t = (2d - 3.5)e + O(1/d)$ où e est le nombre exponentiel.

(Madras, 1995) et (van Rensburg, 1992) ont établi les bornes suivantes sur $T_d(n)$:

$$A' \exp\left(-\delta \log^2 n\right) \lambda_0^n \leq T_d(n) \leq A n^{\frac{1}{d}-1} \lambda_0^n$$

où A, A' et δ sont des constantes positives, et $\lambda_0 = \lambda_d^T$.

5.3 Codage des polycubes-arbres

La manipulation des objets considérés nécessite de définir une notion de tri topologique sur leur classe, permettant d'identifier chaque objet par une représentation qui lui est unique. Dans notre cas, on s'intéresse aux polycubes-arbres libres, et donc on choisit naturellement de les représenter par des structures d'arbres planaires.

On définit d'abord une représentation unique pour chaque objet fixe par un arbre planaire étiqueté pour lequel on définit un codage par un parcours en profondeur que nous noterons *CPP*. Ensuite on choisit l'un des objets fixes équivalents pour être la *forme*

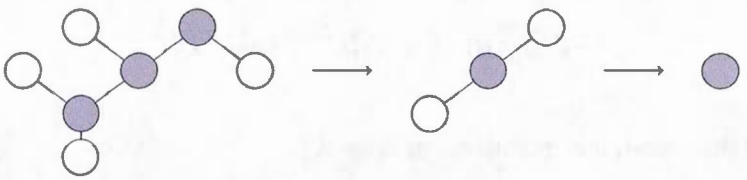
canonique de l'objet libre correspondant.

Pour définir un arbre planaire étiqueté, on a besoin d'une racine, d'un étiquetage et d'un ordre sur les étiquettes.

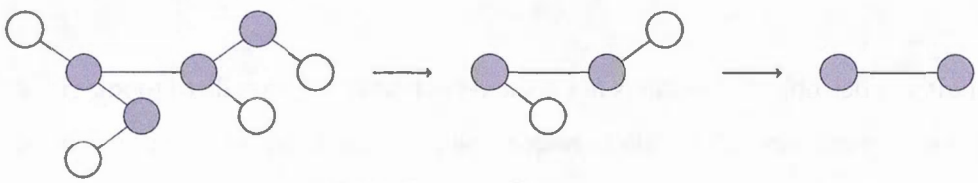
5.3.1 Racine d'un polycube-arbre

Un polycube-arbre est un graphe non-orienté, connexe et sans cycles. Il est donc non-enraciné, mais on peut lui définir une racine de manière unique :
À chaque étape et tant que la taille de l'arbre est > 2, on supprime toutes les feuilles. À la fin il reste ou bien un seul sommet, appelé *centre* de l'arbre, ou bien deux sommets adjacents appelés *bicentre* (Aldous et Wilson, 2000).

Cette procédure prend un temps linéaire avec la taille de l'arbre.



a) Arbre avec centre



b) Arbre avec bicentre

Figure 5.2: Définition de racines pour les arbres non-enracinés

Si l'arbre admet un centre alors on le choisit comme racine, sinon, on étend la notion de racine pour permettre un bicentre.

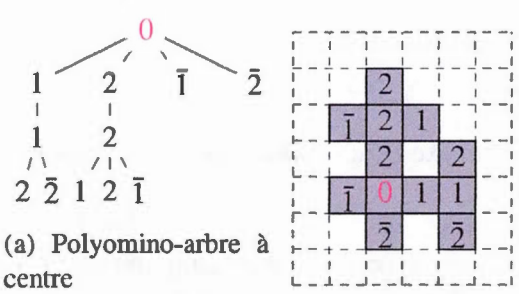
5.3.2 Étiquetage d'un polycube-arbre fixe

Un tel objet est un sous-graphe induit de $\mathbb{Z}^d$, connexe et acyclique, donc chaque paire de sommets est reliée par un chemin unique.
On considère l'alphabet $A = \{0,1,2,\cdots,d,\overline{1},\overline{2},\cdots,\overline{d}\}$ et l'ordre lexicographique sur $A : 0 < 1 < 2 < \cdots < d < \overline{1} < \overline{2} < \cdots < \overline{d}$.
On munit $\mathbb{Z}^d$ d'un repère orthonormé $(O,\vec{x}_1,\cdots,\vec{x}_d)$. Pour tout couple $(u.v)$ de sommets adjacents, il existe un unique $k \in \{1,\cdots,d\}$ tel que $\vec{uv} = \vec{x}_k$ ou $\vec{uv} = -\vec{x}_k$. On dit alors que (u,v) est un *pas* dans $\mathbb{Z}^d$, et qu'il est orienté vers le sens k si $\vec{uv} = \vec{x}_k$, ou vers le sens $\overline{k}$ si $\vec{uv} = -\vec{x}_k$. On définit alors un étiquetage unique pour chaque polycube-arbre fixe. de la manière suivante :

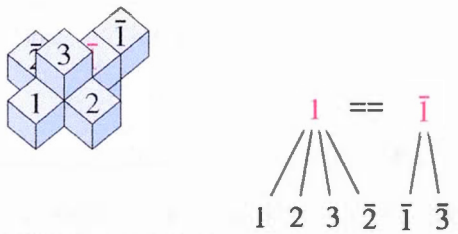
- Si la racine est un centre, on lui donne l'étiquette 0. Si c'est un bicentre alors l'étiquette de chaque extrémité est donnée par le sens du pas de l'autre extrémité vers elle.
- Pour tout sommet autre que la racine, son étiquette est donnée par le sens du dernier pas de l'unique chemin orienté menant de la racine vers lui.

5.3.3 Ordre sur les sommets de l'arbre

Si la racine est un bicentre, alors les deux extrémités sont ordonnées de gauche à droite selon l'ordre de leurs étiquettes ($k < \overline{k}$). Les fils d'un même sommet sont ordonnés de gauche à droite selon l'ordre lexicographique défini sur l'alphabet A des étiquettes. Ainsi, pour chaque polycube-arbre fixe on a construit un unique arbre planaire étiqueté. Inversement, si un tel arbre est valide, alors il détermine de manière unique la structure et l'orientation de l'objet représenté, et donc il représente un unique polycube-arbre fixe. On en déduit que chaque objet est représenté de manière unique par un arbre planaire étiqueté.



(a) Polyomino-arbre à centre



(b) $3d$-Polycube-arbre à bicentre

5.3.4 Codage *DFSE* des polycubes-arbres fixes

DFSE provient de l'anglais depth-first string encoding. On définit ce codage des arbres planaires étiquetés par des mots en effectuant un parcours en profondeur de l'arbre à coder.

En plus de l'alphabet de l'étiquetage on utilise le symbole ' :' pour désigner un retour en arrière, le symbole '=' pour séparer les deux parties du code dans le cas de bicentre, et le symbole '#' pour représenter la fin de la chaîne de caractères qui code l'arbre. On étend l'ordre lexicographique de l'alphabet à ces symboles de la manière suivante : ' :' < '=' < '#' et ces trois symboles sont plus grands que tous les symboles de l'alphabet.

Le parcours en profondeur débute à la racine. Dans le cas d'un centre, on ajoute à la fin de la chaîne obtenue le symbole '#' et c'est le code *DFSE* de l'arbre. Dans le cas d'un bicentre, on supprime l'arête qui relie ses extrémités et on effectue un parcours en profondeur de chacun des deux arbres qui en résultent. Ensuite, les deux chaînes obtenues sont concaténées dans l'ordre de leurs racines et séparées par le symbole '='. On ajoute à la fin le symbole de fin '#' et c'est le code *DFSE*.

Cette construction est une adaptation d'une construction donnée par (Chi *et al.*, 2004), qui ont énoncé le lemme suivant, en appelant arbre valide tout arbre planaire étiqueté associé à un polycube-arbre fixe.

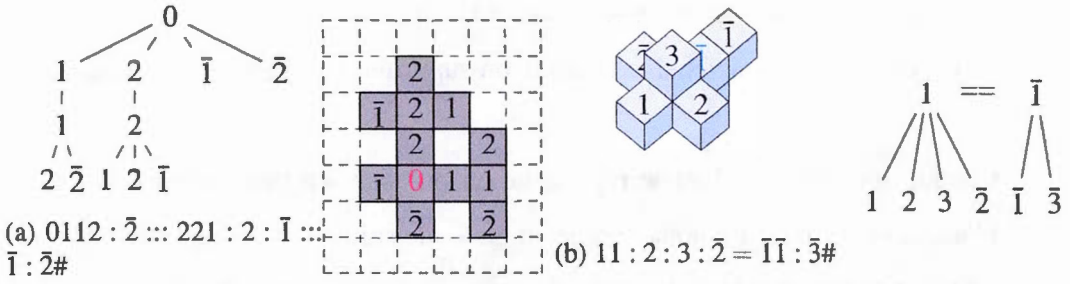


Figure 5.4: Code *DFSE* de polycube-arbre fixe

Lemme 5.1. *Le code DFSE d'un arbre valide est unique et le détermine de manière unique.*

En d'autres termes, chaque arbre valide possède un code *DFSE* unique, et ce code ne peut être associé à un autre arbre valide.

Preuve. L'unicité du code d'un arbre valide est assurée par l'unicité du parcours préfixe d'un arbre planaire étiqueté et ordonné.
Inversement, considérons le code *DFSE* d'un arbre valide, on montre alors dans ce qui suit que ce code détermine cet arbre de manière unique, en prouvant que sa structure détermine de manière unique la racine de l'arbre ainsi que la liste ordonnée des sous-arbres qui lui sont attachés. En effet, deux cas se présentent pour la structure du code :

(i) il est de la forme $0\alpha_1 \cdots : \alpha_2 \cdots : \cdots \alpha_k \cdots \beta\#$ où α_i , β sont des symboles de l'alphabet, différents de 0, $\alpha_i < \alpha_{i+1}$, $k \leq 2d$
Dans ce cas, la racine est un centre étiqueté par 0, et on partitionne la chaîne de la manière suivante : $0|\alpha_1 \cdots : |\alpha_2 \cdots : | \cdots |\alpha_k \cdots \beta|\#$

- la première sous-chaîne est réduite au symbole 0
- on répète l'opération suivante tant que possible :
  La sous-chaîne courante commence au symbole α_i qui suit la sous-chaîne précédente, et elle est la moins longue possible qui contient le même nombre de symboles ' :' que de symboles de l'alphabet A. La dernière sous-chaîne

se termine juste avant le symbole de fin '#'.

- pour chaque sous-chaîne trouvée, on supprime les symboles ' :' situés à la fin.

Chaque sous-chaîne obtenue représente un parcours en profondeur d'un arbre planaire ordonné, il est donc unique, et on a une suite finie et ordonnée de sous-arbres uniques, avec la racine qui est unique, ils forment un arbre unique.

(ii) si le code n'est pas de la forme précédente alors il doit être de la forme $\alpha s = \bar{\alpha} s' \#$ où $\alpha \in \{1, \cdots, d\}$ et s et s' des chaînes telles que $0s\#$ et $0s'\#$ sont du type (i). Dans ce cas, la racine de l'arbre est $(\alpha, \bar{\alpha})$, donc elle est unique, et d'après (i), s et s' déterminent des sous-arbres uniques et dans un ordre bien déterminé.

(i) et (ii) prouvent que chaque arbre valide est caractérisé de manière unique par son code *DFSE*.

Remarque. On déduit de ce lemme que chaque polycube-arbre fixe est caractérisé de manière unique par le code *DFSE* de l'arbre valide qui lui est associé.

5.3.5 Forme canonique *DFCS* des polycubes-arbres libres

DFCS est l'acronyme de depth-first canonical string. Notre objectif étant de réaliser une génération exhaustive des polycubes-arbres libres, il est donc nécessaire de définir un codage pour cette classe d'objets. Rappelons que deux polycubes fixes sont équivalents lorsqu'ils sont identiques à translation, rotation et réflexion près (définition 1.2), et qu'un polycube libre est une classe d'équivalence de polycubes fixes. Alors pour chaque polycube-arbre libre on choisit l'un de ses représentants fixes comme représentant canonique, et le code correspondant est appelé *forme canonique DFCS*.

D'abord on étend l'ordre lexicographique défini sur les symboles, aux chaînes comme suit : pour deux chaînes différentes, on compare successivement à partir de la première

position, les symboles de même indice, et le premier plus petit que l'autre détermine la chaîne la plus petite. Cet ordre est total, et donc parmi les représentants fixes d'un objet libre, un seul possède le code $DFSE$ le plus petit. C'est donc lui le représentant canonique, et son code est la forme canonique de l'objet libre(figure 5.5). Donc, à partir d'un représentant quelconque de l'objet libre, on doit énumérer ses autres représentants et comparer leurs codes. Pour ce faire, on applique successivement à ce représentant tous les éléments du groupe B_d de symétries de l'hypercube de dimension d. Pour des détails sur ce groupe, et en particulier sur le nombre de ses éléments $|B_d| = 2^d d!$, on pourra consulter (Braake, 1984) et (Lunnon, 1972).

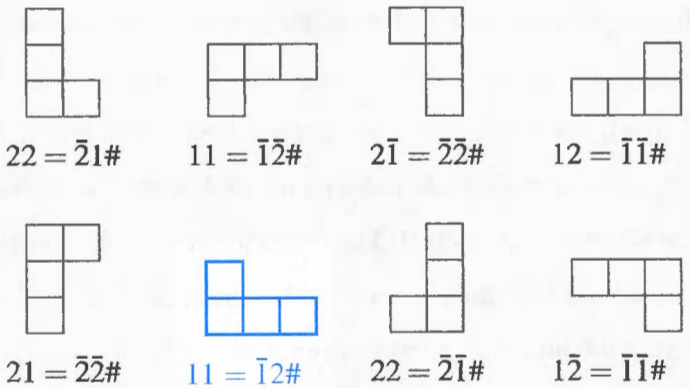


Figure 5.5: Forme canonique d'un polyomino-arbre libre

5.4 Arbre de génération

Dans cette section, on introduit un algorithme pour la génération exhaustive des polycubes-arbres libres basé sur une structure d'arbre de génération. Il produit récursivement, et à chaque étape, tous les objets d'une taille donnée $k > 1$ à partir de ceux de la taille qui précède $k-1$. L'ensemble de départ contient le seul objet de taille 1.

La structure de l'arbre de génération, ainsi que la procédure d'expansion locale sont extraites et adaptées à partir d'un algorithme dû à (Chi *et al.*, 2004) qui l'ont conçu

pour l'extraction de sous-arbres fréquents dans une base de données d'arbres.

5.4.1 Structure de l'arbre de génération

Pour construire un arbre de génération, l'idée principale est de définir un *parent* unique pour chaque objet. Pour ce faire, la technique adoptée consiste à enlever à chaque objet une feuille particulière. Pour un arbre planaire, la *profondeur* d'un sommet est la longueur de l'unique chemin le reliant à la racine, le *niveau* d'indice p est l'ensemble des sommets de profondeur p et la *hauteur* de l'arbre est l'indice du niveau le plus élevé. Chaque objet étant représenté de manière unique par sa forme canonique *DFCS* et par l'arbre canonique correspondant, donc chaque sommet de cet arbre désigne une cellule particulière de l'objet. On s'intéresse à la cellule associée à la feuille la plus à droite du dernier niveau, qu'on appelle la *dernière feuille* de l'arbre. En enlevant cette cellule on obtient un parent unique de l'objet. Donc chaque niveau de l'arbre de génération contient tous les objets d'une taille donnée, et le parent de chacun d'eux s'obtient en supprimant la dernière feuille de son arbre canonique.

5.4.2 Opérateur d'expansion locale

On veut définir un opérateur ϑ sur la classe des polycubes-arbres libres qui permet de construire l'arbre de génération, défini au paragraphe précédent, niveau après niveau jusqu'au niveau désiré, et celà sans omission ni redondance. La structure de cet arbre exige que les enfants d'un objet soient produits en lui ajoutant une cellule dans un emplacement particulier, de façon que la feuille correspondante dans l'arbre canonique soit la dernière feuille.

Soit un polycube-arbre libre quelconque P, donné par sa forme canonique *DFCS*. Les emplacements où l'on peut ajouter une cellule à P sans créer de cycle, afin de produire

un polycube-arbre, forment la *bordure* de l'objet P.
On construit l'arbre canonique A associé à P, ainsi que son représentant canonique, qui est un polycube-arbre fixe implémenté par un tableau multidimensionnel M qui englobe l'objet et sa bordure. Une cellule ajoutée à un emplacement de la bordure se traduit au niveau de l'arbre canonique par une feuille bien déterminée. Parmi les emplacements de la bordure, ceux qui produisent les successeurs de l'objet P par l'opérateur d'expansion locale ϑ, sont appelés les *emplacements actifs*. Autrement dit, un emplacement de la bordure de P est un emplacement actif si et seulement si la feuille correspondante dans l'arbre canonique A est la dernière feuille de P.

Par construction de l'arbre canonique, l'insertion d'une feuille dans un emplacement associé à la bordure et tel que le sommet auquel elle est attachée ne soit pas dans le dernier niveau de l'arbre, donne l'arbre canonique de l'objet produit. Et donc pour que cette feuille ajoutée soit la dernière feuille de l'arbre, il faut que son parent soit dans l'avant dernier niveau, le nœud interne le plus à droite ou l'une des feuilles situées à sa droite. Cependant, si la feuille est attachée à un sommet du dernier niveau alors la structure de l'arbre qui en résulte n'est plus valide, du fait que la nature de la racine est changée, et il faut donc consolider et canoniser l'arbre pour obtenir l'arbre canonique de l'objet produit. Enfin il faut vérifier au niveau de cet arbre canonique si la feuille ajoutée est sa dernière feuille, auquel cas l'emplacement est actif et l'objet produit est accepté.

Pour résumer, les emplacements actifs se trouvent dans les deux derniers niveaux de l'arbre canonique parmi les emplacements suivants, qu'on appelle *emplacements admissibles* : dans un parcours en largeur de l'arbre, on détermine le dernier nœud interne et les feuilles qui suivent. Pour chacun de ces sommets on détermine à l'aide du tableau M les emplacements de la bordure qui lui sont adjacents. C'est l'ensemble des emplacements admissibles.

Il reste à reconnaître les emplacements actifs parmi ces emplacements admissibles :

(i) Tous les emplacements admissibles de l'avant dernier niveau de l'arbre canonique sont des emplacements actifs, puisque l'insertion d'une feuille dans l'un de ces emplacements donne l'arbre canonique de l'objet produit où la feuille ajoutée est la dernière feuille.

(ii) Pour les emplacements admissibles du dernier niveau, on construit l'arbre canonique de l'objet produit et on vérifie si la feuille ajoutée est sa dernière feuille, auquel cas l'emplacement est actif et l'objet produit est accepté, sinon il est rejeté.

Exemple 5.1. Cet exemple donne une illustration de l'application de l'opérateur ϑ à un polyomino-arbre libre donné par sa forme canonique *DFCS* égale à $011:2::\bar{1}:\bar{2}\bar{2}\#$ D'abord il faut construire son arbre canonique A ainsi que son représentant canonique M qui est un tableau de dimension deux. La recherche des emplacements actifs s'effectue en trois étapes (Figure 5.6) :

i) La détermination de la bordure de l'objet se fait en même temps que la construction du tableau M (couleur jaune).

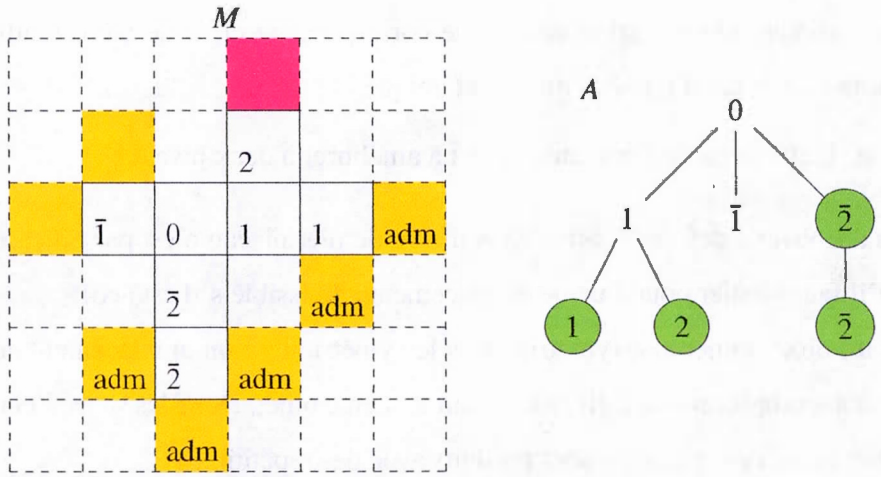
ii) Parmi les cellules de la bordure il faut déterminer celles qui forment les emplacements admissibles. Pour ce faire, on effectue un parcours en largeur de l'arbre canonique A pour déterminer le dernier nœud interne dans ce percours ainsi que les feuilles qui suivent (couleur verte au niveau de l'arbre). Alors les cellules de la bordure qui sont adjacents à l'un de ces sommets sont les emplacements admissibles (distinguées par le symbole *adm*).

iii) Les emplacements admissibles qui sont adjacents à des sommets de l'avant dernier niveau de l'arbre canonique sont des emplacements actifs. Pour les autres, il faut construire l'arbre canonique de l'objet produit et vérifier si la feuille insérée est sa dernière feuille, auquel cas l'emplacement est actif et l'objet produit est accepté, sinon il est rejeté. Dans l'exemple il y a un seul emplacement actif (en

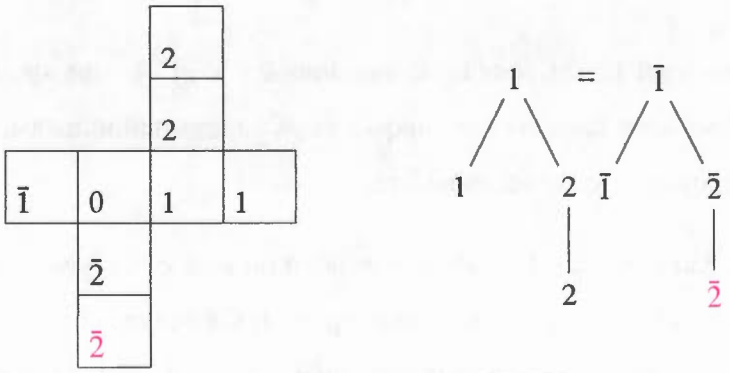
rouge), et donc l'objet considéré génère un seul successeur par l'opérateur ϑ.

Il faut bien remarquer que :

$$\{\textit{Emplacements actifs}\} \subset \{\textit{Emplacements admissibles}\} \subset \textit{Bordure}.$$



(a) Objet source : $011 : 2 :: \bar{1} : \bar{2}\bar{2}\#$



(b) Objet produit : $11 : 22 = \bar{1}\bar{1} : \bar{2}\bar{2}\#$

Figure 5.6: Illustration de l'opérateur ϑ

Conclusion. *L'opérateur d'expansion locale ϑ de la procédure précédente est valide.*

En effet il vérifie les deux conditions de la proposition 4.1 :

1. Tout objet de taille $n+1 > 1$ possède un parent. En effet, si on lui enlève la dernière feuille alors on obtient un objet de taille n, et l'emplacement de la feuille enlevée devient un emplacement admissible pour l'expansion, et puisque l'opérateur ϑ considère tous les emplacements admissibles alors on retrouve obligatoirement l'objet de départ. On a donc $\vartheta(\mathcal{O}_n) = \mathcal{O}_{n+1}$
2. Pour chaque objet, l'arbre canonique correspondant et sa dernière feuille sont uniques, et donc il possède un parent unique.

Remarque. L'efficacité de l'opérateur ϑ est à améliorer à deux niveaux :

1. La découverte des emplacements actifs du dernier niveau n'est pas efficace puisqu'il faut vérifier pour chaque emplacement admissible s'il est actif.
2. Si un objet admet une symétrie alors le symétrique d'un emplacement actif est un autre emplacement actif qui produit le même objet. Donc les successeurs d'un objet ayant des symétries sont produits avec des répétitions.

5.5 Résultats obtenus

L'algorithme est implémenté pour les dimensions 2 , 3 et 4. Une structure d'arbre *2d-aire* est utilisée pour les arbres canoniques et un tableau multidimensionnel pour le représentant canonique de l'objet considéré.

Les objets de chaque niveau de l'arbre de génération sont conservés dans un tableau associatif dont les clés sont les formes canoniques *DFCS* de type chaine de caractères, et dont les valeurs sont des entiers qui représentent les nombres de feuilles des objets. Pour chaque itération de l'algorithme, deux tableaux sont utilisés : l'un doit contenir tous les objets d'une taille donnée, appelé *niveau source*, et l'autre pour contenir les objets produits, appelé *niveau produit*. La structure tableau associatif est choisie pour éliminer les redondances des objets qui pourraient être produits plusieurs fois. À la fin de chaque itération, le niveau produit est enregistré dans un fichier texte, et devient le

niveau source pour l'itération suivante.

L'explosion exponentielle du nombre d'objets de taille donnée limite le niveau atteignable, par défaut de mémoire et aussi par le temps d'exécution qui devient rapidement énorme. Les tableaux suivants résument les résultats d'énumération obtenus. Pour chaque taille n on donne respectivement le nombre de polycubes-arbres libres, le nombre de ceux qui sont pleinement feuillus (section 5.1) et le nombre maximal de feuilles $L_d(n)$, dont l'expression est connue dans le cas des dimensions 2 et 3 (Blondin-Massé *et al.*, 2017). Il faut remarquer que l'algorithme produit effectivement les objets énumérés, mais on présente seulement leurs nombres.

Remarque. On trouve dans la littérature plusieurs travaux d'énumération des polycubes-arbres multidimensionnels, mais ils s'intéressent surtout à la forme fixe de ces objets, alors que dans ce travail ils sont énumérés sous leur forme libre. Pour la dimension 2, ils sont énumérés jusqu'à la taille 16 (on trouve dans la littérature des valeurs jusqu'à la taille 17). Pour les dimensions 3 et 4, des termes sont calculés jusqu'à la taille 12 et la taille 11 respectivement, mais aucun calcul précédent n'est trouvé pour ces termes.

Tableau 5.1: Polyominos-arbres libres

Taille n	Polyominos-arbres libres	Pleinement feuillus	$L_2(n)$
1	1	1	1
2	1	1	2
3	2	2	2
4	4	1	3
5	11	1	4
6	27	2	4
7	83	12	4
8	255	3	5
9	847	1	6
10	2829	6	6
11	9734	74	6
12	33724	11	7
13	118245	2	8
14	416816	21	8
15	1478602	408	8
16	5267171	40	9
17	-	4	10
18	-	76	10
19	-	2053	10
20	-	148	11
21	-	11	12
22	-	279	12

Tableau 5.2: $3d$-Polycubes-arbres libres

Taille n	$3d$-Polycubes-arbres libres	Pleinement feuillus	$L_3(n)$
1	1	1	1
2	1	1	2
3	2	2	2
4	6	2	3
5	21	2	4
6	91	1	5
7	484	1	6
8	2817	4	6
9	17788	100	6
10	116741	42	7
11	788081	16	8
12	5414701	3	9
13	-	1	10
14	-	31	10
15	-	1	11
16	-	989	11
17	-	164	12
18	-	17	13
19	-	2	14
20	-	384	14
21	-	10	15

Tableau 5.3: 4d-Polycubes-arbres libres

Taille n	4d-Polycubes-arbres libres	Pleinement feuillus	$L_4(n)$
1	1	1	1
2	1	1	2
3	2	2	2
4	6	2	3
5	24	3	4
6	122	2	5
7	838	2	6
8	6759	1	7
9	61600	1	8
10	600875	6	8
11	6139448	602	8
12	-	324	9
13	-	148	10
14	-	48	11
15	-	16	12
16	-	3	13
17	-	1	14
18	-	186	14
19	-	30	15
20	-	6	16
21	-	1	17

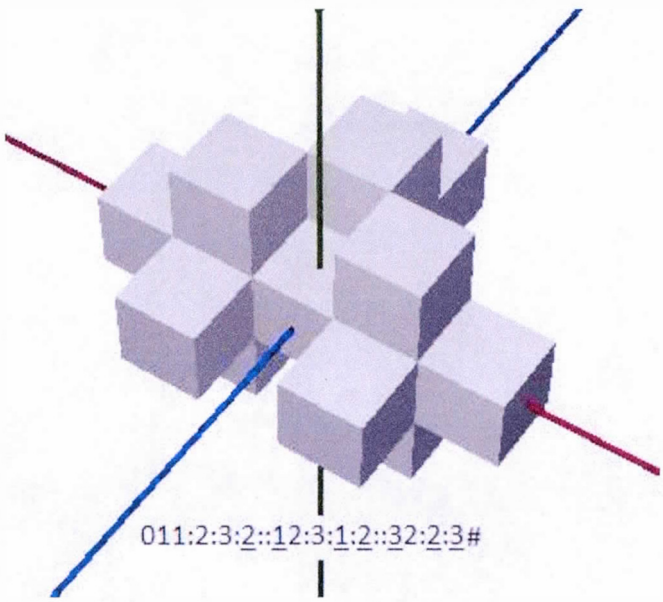


Figure 5.7: 3d-Pleinement-feuillu de taille 15

CONCLUSION

Les problèmes relevant de la combinatoire énumérative sont en général des problèmes difficiles à aborder. Certains d'entre eux sont connus depuis longtemps, comme celui de l'énumération des chemins auto-évitants, des polygones et des polyominos dans les réseaux réguliers du plan, ou celui de l'énumération des polycubes multidimensionnels en général. Ces derniers sont d'une importance grandissante et ont suscité un intérêt particulier des chercheurs du fait de leurs liens avec de nombreux domaines, tels que l'analyse des algorithmes et des structures de données, la physique statistique et la biologie computationnelle. Mais, malgré toute cette motivation, ces problèmes sont encore ouverts. Sauf que les cadres théoriques élaborés pour traiter ces problèmes ainsi que les techniques utilisées ne cessent de progresser et de se multiplier, ce qui a permis des résolutions partielles à travers l'exploration de sous-classes de plus en plus variées.

L'un des progrès importants réalisés récemment concerne les méthodes récursives, avec l'objectif clair de l'unification des formalismes théoriques et de l'élaboration d'outils systématiques pour l'étude de certaines classes combinatoires. Ce mémoire se situe à ce niveau, et forme une exploration de l'essentiel de ce progrès sans rentrer beaucoup dans les détails, pour trois approches principales de la méthodologie récursive, à savoir, la méthode symbolique, les grammaires d'objets et la méthode *ECO*.

La méthode symbolique, développée par F. Flajolet en 2009, est une approche algébrique unifiée qui fournit une collection de règles simples et générales, servant de mécanisme de traduction systématique et complètement formel entre les constructions combinatoires et les opérations sur les fonctions génératrices, qui encapsulent l'information exacte sur les suites de dénombrement des classes correspondantes.

Les grammaires d'objets ont été introduites par (Dutour et Fedou, 1998) dans le but d'étendre la notion de grammaires des mots aux objets combinatoires. Cette approche est fondée sur la même idée de déduire des propriétés combinatoires à partir de descriptions récursives des objets. Elle consiste à construire les objets d'une classe à partir d'objets plus petits, moyennant des opérations de composition, le plus souvent exprimées sous forme de dessins. Par la suite, lorsque la grammaire d'objets est complète et non-ambiguë, on est assuré que tous les objets de la classe sont engendrés sans redondance ni omission, garantissant ainsi un coût de calcul amorti constant (*CAT*) par objet généré, ce qui est considéré efficace pour les algorithmes de génération exhaustive. D'autre part, en appliquant des valuations, les grammaires d'objets se traduisent directement par un système d'équations vérifié par la fonction génératrice de la classe énumérée. De cette manière, les grammaires d'objets permettent la résolution systématique de nombreux problèmes d'énumération.

Une autre technique intéressante qui fait partie de la méthodologie récursive est la méthode *ECO* introduite par (Barcucci *et al.*, 1999). Elle construit chaque objet en faisant grandir localement un objet plus petit, et souvent l'opérateur d'expansion locale définit une construction récursive décrite par une règle de succession qui se traduit par des équations fonctionnelles sur la fonction génératrice de la classe énumérée. De plus, lorsqu'elle est appliquée à des classes exponentielles, la méthode *ECO* fournit un algorithme efficace de type *CAT* pour la génération exhaustive d'objets combinatoires.

La contribution de ce mémoire est l'application de la méthode *ECO* pour l'énumération et la génération exhaustive des polycubes-arbres libres de dimensions 2, 3 et 4 respectivement. Les résultats obtenus confirment la simplicité de la méthode qui repose sur un opérateur généralement intuitif, ainsi que son efficacité du fait de la génération de chaque objet une seule fois et en temps amorti constant. Enfin il faut remarquer que ce travail pourra être amélioré par la construction d'un opérateur mieux adapté à la classe des polycubes-arbres libres, et aussi par la conception d'algorithmes mieux optimisés.

BIBLIOGRAPHIE

Aldous, J. M. et Wilson, R. J. (2000). *Graphs and Applications*. Springer.

Aleksandrovicz, G. et Barequet, G. (2006). Counting d-dimensional polycubes. *Lecture Notes in Computer Sc., 4112.*

Aleksandrowicz, G. et Barequet, G. (2011). The growth rate of high-dimensional tree polycubes. *Elec. Notes in Discrete Math., 38.*

Autebert, J. M. (1987). *Langages algébriques*. Masson.

Bacchelli, S., Barcucci, E., Grazzini, E. et Pergola, E. (2004). Exhaustive generation of combinatorial objects by eco. *Acta Informatica, 40.*

Banderier, C., Mélou, M. B., Denise, A., Flajolet, P., Gardy, D. et Beauchamps, D. G. (2002). Generating functions for generating trees. *Discrete Math., 246.*

Barcucci, E., del Lungo, A., Pergola, E. et Pinzani, R. (1995). A construction for enumerating k-coloured motzkin paths. *Lecture Notes in Comp. Sc., 959.*

Barcucci, E., del Lungo, A., Pergola, E. et Pinzani, R. (1998). Towards a methodology for tree enumeration. *Discrete Math., 180.*

Barcucci, E., del Lungo, A., Pergola, E. et Pinzani, R. (1999). Eco : a methodology for the enumeration of combinatorial objects. *J. Diff. Eq. and Appl., 5.*

Barcucci, E., Pinzani, R. et Spugnoli, R. (1991). Génération aléatoire des animaux dirigés. *publication LaCIM uqam, 10.*

Barequet, G., G.Rote et Shalah, M. (2016). $\lambda > 4$: An improved lower bound on the growth constant of polyominoes. *Comm. of the ACM (CACM), 59(7).*

Barequet, R., Barequet, G. et Rote, G. (2010). Formulae and growth rates of high-dimensional polycubes. *Combinatorica, 30.*

Bergeron, F., Labelle, G. et Leroux, P. (1998). *Combinatorial species and tree-like structures*. Cambridge University Press.

Blondin-Massé, A., de Carufel, J., Goupil, A. et Samson, M. (2017). Fully leafed tree-like polyominoes and polycubes. *In Combinatorial Algorithms, volume 10765*

of Lect. Notes Comput. Sci. 28th International Workshop, IWOCA 2017, Newcastle, NSW, Australia, Springer,2018.

Braake, M. (1984). Structure and representations of the hyperoctahedral group. *J. of Math. Phys., 25.*

Broadbent, S. R. et Hammersley, J. M. (1957). *Percolation processes.* Proc. Cambridge Philosophical Society.

Chi, Y., Yang, Y. et Muntz, R. (2004). Canonical forms for labeled trees. *University of California.*

Chung, F., Graham, R., Hoggatt, V. et Kleiman, M. (1978). The number of baxter permutations. *J. Comb. theory, 24.*

deBruijn, N. G. (1964). *Pólya's theory of counting.* Wiley.

del Lungo, A. (1992). *Algoritmi per la generazione di animali direzionati.* (Mémoire de maîtrise). Firenze.

del Lungo, A., Duci, E., Frosini, A. et Rinaldi, S. (2004). On the generation and enumeration of some classes of convex polyominoes. *Mathematics subject classifications, 05A15.*

Denise, A. (2001). Structures aléatoires, modèles et analyse de génomes. *Université de Paris-sud.*

Dutour, I. (1996). *Grammaires d'objets.* (Thèse de doctorat). Université Bordeaux I.

Dutour, I. et Fedou, J. M. (1998). Object grammars and random generation. *Discrete Math. and Theor. Comp. Sc, 2.*

Finch, S. (2003). *Mathematical Constants.* Cambridge University Press.

Flajolet, P. (1980). Combinatorial aspects of continued fractions. *Discrete Math., 32.*

Flajolet, P., Françon, J. et Vuillemin, J. (1980). Sequences of operations analysis for dynamic data structures. *J. of Algorithms, 1.*

Flajolet, P. et Sedgewick, R. (2009). *Analytic Combinatorics.* Cambridge University Press.

Gire, S. (1993). *Permutations à motifs exclus et cartes planaires.* (Thèse de doctorat). Université de Bordeaux I.

Golomb, S. W. (1996). *Polyominoes : puzzles, patterns, problems and packings,*

2nd Ed. Princeton University Press.

Goulden, I. P. et Jackson, D. M. (1983). *Combinatorial Enumeration.* John Wiley.

Graham, R. L., Knuth, D. E. et Patashnik, O. (1989). *Concrete Mathematics.* Addison Wesley.

Guttmann, A. J. (2009). *Polygons, Polyominoes and Polycubes.* Springer.

Guttmann, A. J. et Gaunt, D. S. (1978). On the asymptotic number of lattice animals in bond and site percolation. *J. Phys. A, 11.*

Jensen, I. (2003). Counting polyominoes. *Lecture Notes in Computer Sc., 2659.*

Jensen, I. et Guttmann, A. J. (2000). Statistics of lattice animals and polygons. *J. Phys. A, 33.*

Joyal, A. (1981). Une théorie combinatoire des séries formelles. *Advances in Math., 42.*

Kemp, R. (1984). *Fundamentals of the average case analysis of particular algorithms.* Springer Fachmedien Wiesbaden.

Klarner, D. A. et Rivest, R. L. (1973). A procedure for improving the upper bound for the number of n-ominoes. *Canad. J. Math., 25.*

Lunnon, W. F. (1972). *Symmetry of cubical and general polyominoes.* in Graph theory and computing, R. C. Read, Academic Press.

Luther, S. et Mertens. S. (2011). Counting lattice animals in high dimensions. *Journal of statistical mechanics.*

Madras, N. (1995). A rigorous bound on the critical exponent for the number of lattice trees, animals and polygons. *J. Stat. Physics, 78.*

Madras, N. (1999). A pattern theorem for lattice clusters. *Annals of Combinatorics, 3.*

Parisi, G. et Sourlas, N. (1981). Critical behaviour of branched polymers and the lee-yang edge singularity. *Phys. Rev., 46.*

Pergola, E., Pinzani, R. et Rinaldi, S. (2002). Approximating algebraic functions by means of rational ones. *Theoretical Computer Science, 270.*

Provençal, X. (2008). Combinatoire des mots, géométrie discrète et pavages. *Université du Québec à Monréal.*

Roberts, F. S. et Tesman, B. (2005). *Applied Combinatorics 2nd Ed.* Pearson.

Schützenberger, M. P. (1963). On context-free languages and push-down automata. *Information and Control, 6.*

Stanley, R. (2010). *Enumerative Combinatorics Vol.2.* Cambridge University Press.

van Rensburg, J. (1992). On the number of trees in z^d. *J. Phys. A : Math. Gen. 3523, 25.*

Viennot, X. G. (1985). Enumerative combinatorics and algebraic languages. *Lecture Notes in Computer Sc., 199.*

West, J. (1990). *Permutations with forbidden subsequences.* (Thèse de doctorat). Massachusetts Institute of Technology.

West, J. (1995). Generating trees and the catalan and schröder numbers. *Discrete Math., 146.*

Wilf, H. S. (1994). *Generating Functionology 2nd Ed.* Academic Press.

INDEX

arbre, 10
de Catalan,de Cayley, 11
de dérivation, 35
de génération, 53

chemin, 11
de Dyck,de Motzkin, 12
classe combinatoire, 14
construction admissible, 21

fonction génératrice, 15
forme canonique, 74

grammaire d'objets, 34

non-ambiguïté, 42

opérateur, 52
opération d'objets, 31

parametre
multidimensionnel, 16
paramètre
discriminateur, 1
polycube, 8
polycube-arbre, 68
pleinement feuillu, 68
polyomino, 5
parallélogramme, 33

règle de réécriture, 54
réseaux réguliers, 5

série formelle, 18
spécification combinatoire, 26

taux d'accroissement limite, 7, 9, 69

valuation, 45