Constructions par greffe, combinatoire analytique et génération analytique

Alice Jacquot

Université Paris Nord - Paris 13
Sorbonne Paris Cité
École doctorale Galilée
Laboratoire d'Informatique de Paris Nord

THÈSE

pour l'obtention du grade de **Docteur de l'université Paris 13**,
discipline Informatique

Constructions par greffe, combinatoire analytique et génération aléatoire

présentée et soutenue publiquement par

Alice JACQUOT

le 1er avril 2014

devant le jury composé de :

Olivier BODINI	**directeur de thèse**
Dominique ROSSIN	rapporteur
Srecko BRLEK	rapporteur
Frédérique BASSINO	examinatrice
Alain DENISE	examinateur
Jean MAIRESSE	examinateur
Brigitte VALLÉE	examinatrice

Remerciements

Si je suis fière d'arriver enfin au bout de *ma* thèse, je reconnais que je dois énormément à tous ceux qui m'ont si bien entouré pendant ces années, et qui ont permis mon épanouissement à la fois scientifique et personnel. Avant tout autre chose, je tiens donc à remercier chaleureusement tous ceux qui ont, d'une manière ou d'une autre, participé à ma découverte de la recherche et de son environnement. J'y ai trouvé un monde bien plus humain que ce que les théorèmes laissent parfois percevoir et dans lequel je me suis sentie chez moi.

Merci à Olivier Bodini, pour tout. Je crains malheureusement ne pas connaître assez de superlatifs pour pouvoir exprimer tout le bien que je pense de lui, mais je vais tout de même essayer. Tout du long de cette thèse, et même avant, il a été pour moi un directeur formidable. Avec un enthousiasme débordant il a su me proposer des problèmes sur mesure, me montrer des voies sans m'en dévoiler les chemins et m'apprendre à devenir autonome. Il a été un directeur, un guide, un collègue, un ami et un soutien lorsque j'en avais besoin.

Je remercie mes rapporteurs, Srecko Brlek et Dominique Rossin pour avoir accepté d'accorder autant de temps à mes travaux malgré leurs nombreuses occupations. Merci aux membres de mon jury, Frédérique Bassino, Alain Denise, Jean Mairesse, Michèle Soria et Brigitte Vallée. Je suis honorée qu'ils aient accepté d'examiner mes travaux et les remercie.

Je souhaite, en toute généralité, remercier tous organisateurs et orateurs de séminaires, conférences, groupes de travail ou autres journées de rencontres auxquels j'ai eu l'occasion de participer. Ces moments de partage sont toujours, en plus d'être agréables, extrêmement enrichissants et ont été ma principale source de culture scientifique durant ces trois années de thèse.

Le groupe Alea est celui parmi lequel j'ai eu le plus d'échanges, à Luminy comme à Paris ou à Bordeaux. J'y ai rencontré de nombreux amis et encore plus de gens formidables dont je vais malheureusement oublier plusieurs noms. Je remercie le groupe dans son ensemble pour son ambiance et sa dynamique scientifique, dans le partage constant. Je remercie plus spécifiquement : Adeline pour les grandes discussions en toutes circonstances et pour me rappeler le plaisir des choses simples dès qu'il fait beau, Jérémie Lumbroso parce qu'on est comme ça, Gwendal Collet qui résiste à toutes les pressions, Matthieu Josuat-Vergès, Julien Courtiel, très bon guide bordelais, Basile Morcrette et son alto, Anne Briquet juste parce que, Yann Ponty avec ou sans chocolat, Cécile Mailler qui aide à voir les choses du bon coté, Thu Hien Nguyen Thi pour le MPRI, Julien Clément pour la musique chaque année, Elie de Panafieu et son sourire permanent, Mathieu Lecomte pour les folles journées en Amérique, Carine Pivoteau et Mathilde Bouvel qui ont guidé mes premiers pas vers le CIRM, Laura Giambruno pour ma tout première chambre partagée, Jehanne Dousse pour les ping-pong, Julien David malgré toutes nos réactions singulières, Éric Rémila pour la petite balade tranquille, Philippe Marchal, Adrien Boussicault, Kerstin Weller, Christelle Rovetta, Mireille Bousquet-Mélou, Viviane Pons, Danièle Gardy, Antoine Genitrini, Irène Marcovici, Sandrine Dasse-Hartaut, Julien Bureaux, Matteo Silimbani, Florent Le Gac,

Hanane Tafat, Olivier Roussel, Johan Oudinet, Omar Aït Mous, Hayat Cheballah...

Je remercie tous les collègues du LIPN qui en font un endroit chaleureux. Merci à l'équipe CALIN et en particulier aux réguliers du thé combinatoire : Axel Bacher, Cyril Banderier, Frédérique Bassino, Nick Beaton, Olivier Bodini, Valentin Bozom, Matthieu Deneufchâtel, Gérard Duchamps, Thomas Fernique, Hoang Ngoc Minh, Pierre Nicodème, Andrea Sportiello, Adrian Tanasa, Christophe Tollu. Et en merci encore plus grand à ceux qui y ont un jour apporté un gâteau ou un « thé ». Je remercie également mes coburreaux, Laurent Poinsot et Ferhan Pekergin. Merci à Nicolas Rolin, mon petit frère de thèse, et à Alexandra Ugolnikova et Quentin de Mourgues. Je remercie tous ceux avec qui j'ai eu le plaisir d'enseigner, ou qui ont gentiment accepté de me remplacer lors d'une absence. Je remercie tout particulièrement Christophe Tollu et Olivier Bodini qui m'ont fait confiance pour la préparation de mes TD - sans être jamais loin en cas de problème. Je remercie également mes étudiants, pour le meilleur et pour le pire : j'ai beaucoup appris grâce à eux. Merci à Camille Vauchel, mon petit stagiaire. Merci à Camille Coti pour la grande classe de ses blagues, merci à Laure pour être une super directrice de labo qui offre des bonbons quand on discute dans son bureau, merci à Brigitte et Nathalie, merci au comité des thèses et aux participants du séminaire junior.

Merci bien sur à mes co-auteurs, Axel Bacher, Olivier Bodini, Gwendal Collet, Julien David, Philippe Duchon, Danièle Gardy, Bernhard Gittenberger et Ljuben Mutafchiev sans qui cette thèse ne serait évidement pas la même. Merci pour tous les moments de travail intense et l'émulation réciproque !

Merci à Olivier Bodini (encore lui ?), Philippe Duchon, Cyril Nicaud et Yann Ponty pour m'avoir accueillie en stage.

Merci à l'ANR MAGNUM, pour les sous et les missions mais aussi pour les réunions et l'esprit d'équipe.

Merci aussi tous les jeunes (et moins jeunes) chercheurs venant d'un peu partout et que j'ai rencontré à l'EJC-IM 2013, à CANT, lors des rencontres franco-britannique de combinatoire ou même lors des formations du CFDIP.

Je remercie également mes divers autre amis pour avoir fait de ces années de thèses de belles années de découvertes, de rencontres et de moments partagés, que ce soit autour de jeux, de tapis d'accro, de hoops, d'un verre ou même sur gtalk. Je pense notamment à Fabrice, Gloups, Sylvain, Guillaume, Denise, Jérémie, Julie, ceux que j'oublie et plus généralement aux moisséens et tourangeaux, aux cirqueux et à tous les spammés de Jeux à Quincampoix, sans oublier toute ma grande famille.

Et merci à Axel, puisque tu tiens à être mentionné, mais je crois que tu sais déjà tout le bien que je pense de toi et que je peux décemment écrire ici.

Table des matières

Introduction

Une structure combinatoire est un objet mathématique discret. Il en existe de très nombreuses sortes, des listes chaînées aux polynômes sur des corps finis, en passant par les arbres, les λ-termes et les polyominos. L'objet de la combinatoire est l'étude de ces structures. Parmi les problèmes les plus classiques en combinatoire, on trouve les problèmes d'énumération : combien y a-t-il de structures vérifiant certaines propriétés ? Comme l'on traite généralement des familles infinies d'objets, on va souvent utiliser une fonction taille qui associe un entier à chaque objet et la question précédemment énoncée devient : combien y a-t-il de structures de taille n vérifiant certaines propriétés ? Rien que sur ce type de questions, de nombreux types d'approches différentes existent. On peut, selon les cas, trouver une formule close donnant explicitement le nombre de structures de taille n pour tout n, chercher à déterminer la croissance asymptotique du nombre d'objets en fonction de la taille, développer des outils de calcul formel pour pouvoir calculer efficacement les premières valeurs, étudier les propriétés algébriques de la série génératrice... Mais l'énumération n'est pas la seule question qui se pose en combinatoire. On peut par exemple s'intéresser aux propriétés moyennes de classes d'objets, à l'existence d'une structure vérifiant certaines contraintes, à la complexité d'algorithmes basés sur des structures discrètes ou à la génération exhaustive ou aléatoire. Les domaines d'applications de tels résultats sont très dépendants des structures étudiées et contiennent entre autres l'algorithmique, la bio-informatique, la physique statistique...

Pour cela on va souvent - et ce sera le cas dans cette thèse - chercher à décomposer des objets en objets plus simples ou plus petits. Ou, en raisonnant dans l'autre sens, à *combiner* des objets pour obtenir des objets plus gros ou plus complexes. Ce principe est celui que l'on retrouve à la base des processus récursifs. Mais toutes les décompositions ne se valent pas : certaines seront plus naturelles, d'autres permettront une analyse asymptotique, le calcul efficace des premiers termes ou la génération aléatoire efficace. Il faut donc choisir avec soin sa décomposition en fonction des résultats que l'on souhaite obtenir.

Cette thèse se situe dans le domaine de la combinatoire analytique, décrit par Philippe Flajolet et Robert Sedgewick dans [FS09a]. L'idée maîtresse est d'utiliser des *spécifications* pour décrire la manière dont on décompose les structures que l'on étudie. On en déduit des relations sur des séries génératrices, ce qui conduit, par des méthodes analytiques, à des propriétés asymptotiques. La génération aléatoire vient compléter l'étude de ces propriétés asymptotiques. On y cherche à tirer selon une distribution -souvent uniforme - une structure combinatoire vérifiant certaines

propriétés, comme par exemple des arbres de taille donnée. La combinatoire analytique est un cadre agréable pour la génération aléatoire, car on peut également s'appuyer sur les spécifications pour définir des générateurs aléatoires, comme les générateurs de Boltzmann [DFLS04] par exemple, en utilisant la fonction analytique associée à la série génératrice des objets.

Dans cette thèse, nous rencontrerons différents types d'objets combinatoires pour lesquels nous donnerons de nouvelles décompositions. Nous allons notamment définir des opérateurs de greffes qui permettent d'obtenir un grand objet en ajoutant des appendices de tailles bornées à un endroit donné d'un objet plus petit, par opposition aux processus branchants qui combinent plusieurs objets de tailles arbitraires pour en former un plus gros. L'exploration de ces nouvelles spécifications apporte une nouvelle vision de ces objets, et de cette vision nous obtiendrons de nouveaux résultats d'analyse asymptotique ou de génération aléatoire, inaccessibles par des spécifications plus classiques.

La philosophie générale de cette thèse peut donc être résumée de la manière suivante : étant donnée une classe combinatoire, peut-on trouver des bijections permettant d'obtenir une vision différente et utile sur ces structures. Cette étape est loin d'être automatique en général, et nous chercherons à exploiter au mieux la structure des objets pour avoir des descriptions alternatives, afin de pouvoir y appliquer des méthodes différentes et ainsi obtenir des résultats nouveaux. A partir de ces décompositions originales, nous utiliserons des méthodes classiques pour l'énumération et on exploitera leurs non-ambiguïtés constructives pour obtenir des générateurs aléatoires efficaces.

Plan de la thèse

Cette thèse est divisée en quatre parties. La première concerne les définitions classiques qui reviendront tout au long de ce manuscrit (classes combinatoires, spécifications, générateurs de Boltzmann...). La deuxième partie se concentre sur l'algorithme de Rémy et plus particulièrement sur des spécifications holonomes pour les arbres binaires et unaires-binaires basées sur des opérateurs de greffe. La troisième a trait aux λ-termes et l'on y utilise des spécifications faisant intervenir des produits de Hadamard dans un cas et à la fois des substitutions et des branchements dans l'autre. Enfin, la quatrième partie regroupe des travaux sur deux objets différents : les arbres comptés selon le nombre d'occurrences d'un motif donné, pour lesquels la spécification est un système fini évoluant à la fois suivant la profondeur et suivant la largeur de l'arbre spécifié, et les polyominos digitalement convexes où la spécification donnée est une ré-interprétation d'un algorithme de reconnaissance [Brl] et prend comme classe élémentaire la classe des couples d'entiers premiers entre eux.

Dans tous ces cas, ces nouvelles spécifications mènent à de nouveaux résultats, d'énumération ou de génération aléatoire. Ces résultats sont obtenus par des méthodes variées et classiques en combinatoire analytique (point col, transformée de Mellin, théorème de Bender, théorèmes de transfert de Flajolet-Odlysko,...).

Nous pouvons maintenant donner un plan plus détaillé de ces différentes parties et des résultats qu'elles contiennent.

Partie I : Préliminaires

Chapitre 1 Classes combinatoires et spécifications

Ce chapitre reprend les définitions des classes combinatoires étiquetées et non-étiquetées, des séries génératrices ordinaires et exponentielles, ainsi que celles de tous les constructeurs utilisés au cours de cette thèse. Une brève analogie avec la théorie des espèces y est également présentée.

Chapitre 2 Arbres

Les arbres enracinés planaires apparaissant dans presque tous les chapitres de cette thèse, aussi nous avons décidé de leur consacrer un chapitre introductif regroupant les définitions et des faits élémentaires ou bien connus sur différentes classes d'arbres. Le point de vue adopté ici reste celui de la combinatoire analytique.

Chapitre 3 Génération aléatoire

Ce chapitre présente la génération aléatoire en général et s'attarde plus particulièrement sur la méthode de Boltzmann. Nous y présenterons en particulier la complexité en nombre de bits aléatoires consommés.

Partie II : Algorithme de Rémy

L'algorithme de Rémy est un algorithme *ad hoc* classique de génération aléatoire d'arbres binaires dont la complexité en temps est linéaire et celle en bits aléatoires est en $O(n \log n)$. Cette partie présente des travaux effectués avec Axel Bacher et Olivier Bodini [BBJ13, BBJ14] autour de cet algorithme.

Chapitre 4 Introduction

L'algorithme de Rémy, rappelé dans ce chapitre, repose sur une récursion simple sur le nombre d'arbres. Nous réinterprétons ici cette équation en termes d'équation différentielle sur les séries génératrices. Cette ré-interprétation nous permet de faire un parallèle avec le générateur de Boltzmann qui suit cette spécification différentielle.

Chapitre 5 Amélioration de l'algorithme de Rémy

Nous utilisons ici la vision pointée de l'algorithme de Rémy et une correspondance entre feuilles et nœuds internes des arbres binaires pour obtenir un générateur d'arbres binaires de complexité linéaire en temps et en nombre de bits aléatoires utilisés. Cet algorithme est le premier algorithme optimal en nombre de bits aléatoires pour la génération des arbres binaires. Nous modifions ensuite cet algorithme pour obtenir un générateur aléatoire *quasi-optimal* en nombre de bits aléatoires : la génération uniforme d'un arbre de taille n consomme $2n + \Theta(\log(n)^2)$ bits aléatoires en moyenne.

Chapitre 6 Extension aux arbres de Motzkin

En suivant le même schéma que pour les arbres binaires, nous allons trouver et exploiter une équation différentielle linéaire sur la série génératrice des arbres de Motzkin afin d'obtenir un algorithme « à la Rémy ». Cet algorithme a une complexité linéaire en temps et en bits aléatoires, ce qui en fait le meilleur algorithme actuel pour générer des arbres de Motzkin.

Partie III : λ-termes

Les λ-termes sont des objets fondamentaux de l'informatique théorique. Cette partie présente deux ensembles de résultats sur la spécification, l'énumération et la génération aléatoire de différentes classes de λ-termes.

Chapitre 7 Introduction

Nous introduisons dans ce chapitre les classes de λ-termes étudiés ainsi que quelques faits élémentaires sur ceux-ci.

Chapitre 8 Méthode bijective pour λ-termes linéaires et affines

Nous étudions ici deux classes de λ-termes : les λ-termes linéaires et les λ-termes affines. Dans ces deux cas, nous exhibons des bijections avec des cartes combinatoires à degrés contraints, reposant sur des parcours en profondeur d'abord. Grâce à ces bijections, on obtient les comportements asymptotiques ainsi que des générateurs aléatoires efficaces pour les classes de λ-termes étudiées. Ces résultats sont tirés d'un travail en commun avec Olivier Bodini et Danièle Gardy [BGJ10, BGJ13].

Chapitre 9 Spécification monovariée des λ-termes

Nous présentons ici une spécification monovariée pour les λ-termes généraux, linéaires, affines et ceux dont le nombre de liens des nœuds unaires est borné. Cette spécification repose sur un principe de greffe, comme dans la partie précédente. En revanche, les équations obtenues ici ne sont pas holonomes, ce qui empêche d'utiliser la plupart des techniques de combinatoire analytique classiques. Ces nouvelles spécifications permettent dans tout les cas un calcul plus efficace des premiers termes des séries génératrices. Dans le cas des λ-termes multi-linéaires, on parvient également à obtenir le comportement asymptotique des coefficients de la série génératrice. On y présente également de nouvelles bornes sur le nombre asymptotique de λ-termes clos. Ce chapitre est tiré de [BGGJ13], écrit en collaboration avec Olivier Bodini, Danièle Gardy et Bernhard Gittenberger.

Partie IV : Autres travaux autour de la génération aléatoire

Ce chapitre regroupe deux travaux n'ayant pas leur place dans les deux parties précédentes mais s'inscrivant pourtant dans le cadre de cette thèse : on y retrouve le même principe de nouvelle décomposition bijective permettant l'analyse et la génération aléatoire.

Chapitre 10 Arbres avec occurrences restreintes de motifs

Nous nous intéressons dans ce chapitre à construire automatiquement une spécification pour les arbres enracinés planaires comptés selon leur nombre total de nœuds et selon leur nombre d'occurrences d'un motif donné. La spécification obtenue est algébrique et repose sur un processus de branchements. Mais ce processus n'est pas forcément branchant à chaque étape : plusieurs nœuds de même profondeur devront parfois être dérivés simultanément. Une fois cette spécification calculée, on déduit un générateur de Boltzmann et un nouvel algorithme linéaire de reconnaissance de motifs dans un

arbre. Tous ces résultats s'étendent aux arbres à arités restreintes, aux arbres colorés et aux ensembles de motifs. Ils ont été obtenus avec Gwendal Collet et Julien David, avec qui un article sur le sujet est en cours de rédaction [CDJ].

Chapitre 11 Polyominos digitalement convexes

Dans ce chapitre, on utilise une caractérisation de Brlek *et al.* des polyominos digitalement convexes pour obtenir une spécification symbolique de ces polyominos. De cette spécification, on déduit la forme limite des chemins NW-convexes, *i.e.*, les quarts de polyominos digitalement convexes, et un générateur aléatoire de polyominos digitalement convexes, utilisant un générateur de Boltzmann. Expérimentalement, en utilisant ce générateur aléatoire, nous avons pu observer une forme limite déterministe pour les polyominos digitalement convexes. Ces travaux ont été effectués avec Olivier Bodini, Philippe Duchon et Ljuben Mutafchiev [BDJM13].

Première partie

Préliminaires

Chapitre 1

Classes combinatoires et spécifications

Dans ce chapitre nous allons commencer par décrire les classes combinatoires non-étiquetées : ce cadre permet de définir facilement les classes combinatoires en toute généralité, sans la notion d'atomes ou de constructions.

1.1 Classe combinatoire non-étiquetée

Une *classe combinatoire* est un couple (E, t) tel que :
- E est un ensemble (fini ou infini)
- t est une fonction taille de E dans l'ensemble des entiers naturels telle que pour chaque taille n il y a un nombre fini d'objets de cette taille.

Exemple. L'ensemble des mots sur 2 lettres muni de la fonction taille "longueur du mot" est une classe combinatoire : pour tout $n > 0$, il y a 2^n mots de taille n.

Soit $\mathcal{A}$ l'ensemble des arbres enracinés planaires, et t la fonction qui associe à un arbre son nombre de nœuds. Alors $(\mathcal{A}, t)$ est une classe combinatoire.

D'autres exemples de classes combinatoires peuvent être l'ensemble des entiers naturels ayant pour taille leur valeur, les chemins sur un graphe donné ayant pour taille leur longueur, les permutations, les formules logiques avec leur longueur pour taille, les séquences ADN avec leur nombre de nucléotides pour taille, les pavages par un ensemble de tuiles donné d'une surface finie où la taille est celle de cette surface, les urnes composées de boules noires et blanches où la taille est le nombre de boules...

On déduit directement de la définition que tout ensemble dénombrable, associé à une fonction taille, peut-être vu comme une classe combinatoire. C'est donc, a priori, un cadre très vaste. Dans la suite, nous allons restreindre ce cadre aux classes constructibles, *i.e.*, que l'on peut définir récursivement.

On notera en général, par abus de notation, $\mathcal{A}$ une classe combinatoire pour la classe $(\mathcal{A}, t)$. Soit $\mathcal{A}$ une classe combinatoire et $a \in \mathcal{A}$. On note $|a|$ la taille de a dans la classe combinatoire $\mathcal{A}$, ou $|a|_{\mathcal{A}}$ en cas d'ambiguïté, $\mathcal{A}_n$ l'ensemble des objets de taille n dans $\mathcal{A}$ (et donc, $\forall n \in \mathbb{N}$, $\mathcal{A}_n$ est fini), et a_n le nombre d'objets de taille n dans $\mathcal{A}$.

Munis de ces définitions et notations, nous allons maintenant pouvoir décrire les séries génératrices et les constructeurs, qui sont au cœur de la combinatoire analytique.

1.2 Séries génératrices ordinaires

Soit $\mathcal{A}$ une classe combinatoire. La série formelle :

$$A(x) = \sum_{a \in \mathcal{A}} X^{|a|} = \sum_{n \in \mathbb{N}} a_n X^n.$$

est appelée *série génératrice ordinaire* (ou *OGF* pour Ordinary Generating Function) de la classe $\mathcal{A}$.

On peut voir cette définition comme représentant énumérativement la fonction taille de la classe $\mathcal{A}$: on associe à chaque objet a de $\mathcal{A}$ un monôme $X^{|a|}$, puis on fait la somme de ces monômes pour tout objet de $\mathcal{A}$. On a ainsi perdu toutes les autres informations sur les objets, c'est-à-dire les informations structurelles. La série génératrice à elle seule ne permet donc pas de définir une classe combinatoire. Cependant, la connaissance de la série génératrice permet déjà de connaître beaucoup de choses sur les objets : elle va souvent nous indiquer, par la croissance de ses termes, quels sont les processus récursifs possibles pour la construction des structures. On sait également que toutes les classes combinatoires ayant la même série génératrice sont en bijection les unes avec les autres, et que l'on peut donc traduire les résultats énumératifs obtenus sur l'une aux autres classes comptées par la même série. Conjecturer que deux classes ont la même série (par exemple, en entrant les premiers termes d'une série dans l'OEIS [Slo]) permet de chercher, et parfois de trouver, des bijections explicites (isomorphismes) qui permettent souvent de traduire également les propriétés structurelles d'une classe à l'autre.

De plus, la manipulation de séries génératrices est souvent plus simple que l'énumération directe. En effet dans de nombreux cas, lorsque la classe a une structure récursive forte, la série génératrice admet une forme close. Par exemple, la série génératrice de la classe des mots à deux lettres $\mathcal{L}$, définie comme :

$$L(X) = \sum_{n \geq 0} 2^n X^n$$

admet comme forme close

$$L(X) = \frac{1}{1 - 2X}.$$

Il existe également de nombreux cas où il est relativement aisé de trouver un système d'équations que vérifie la série génératrice (notamment grâce aux constructeurs, définis ci-dessous) mais où l'énumération exacte est a priori difficile pour une taille quelconque.

Lorsque la série génératrice admet un rayon de convergence ρ non nul, on peut alors considérer la fonction (réelle ou complexe) associée. Des propriétés analytiques de cette fonction se déduisent alors des propriétés énumératives, notamment asymptotiques, sur la classe combinatoire. Par exemple dans le cas très (trop) simple de la classe $\mathcal{L}$, le

seul fait que le rayon de convergence de $L(X)$ est $1/2$ (et que la singularité dominante en $z = 1/2$ est un pôle simple) nous permet de conclure que le nombre de mots à deux lettres de taille n sera asymptotiquement équivalent à 2^n, sans aucun argument combinatoire. Dans de très nombreux cas, la série génératrice ordinaire ne converge pas, comme par exemple pour les permutations dont la série est $\sum_{n\geq 0} n!X^n$. Nous verrons un peu plus loin la définition de classes étiquetées et de séries génératrices exponentielles qui sont des notions plus adaptées pour certains de ces cas.

1.3 Constructeurs

Cette thèse se concentre principalement sur les classes combinatoires *constructives* : une classe combinatoire constructive est un ensemble d'objets que l'on peut construire récursivement à partir d'un ensemble fini d'objets (la section 1.3.1 décrit deux ensembles élémentaires) et des *constructeurs* (dont les plus classiques sont décrits dans ce chapitre) qui permettent de créer des liens entre des sous-objets. Dans tout ce manuscrit l'égalité entre spécifications représente l'isomorphisme entre les classes ainsi définies, et sous-entend que l'on connaît une bijection explicite permettant de passer d'une classe à l'autre.

1.3.1 Classe atomique et classe neutre

On appelle *classe atomique* une classe réduite à un seul objet de taille 1, et *atome* un objet de taille 1.
Par exemple, en considérant que la taille d'un arbre est son nombre total de nœuds, le singleton contenant un arbre réduit à une feuille est une classe atomique. Si on regarde les mots sur l'alphabet $\{0, 1\}$ en comptant comme taille leur nombre de lettres, le singleton contenant le mot 0 est une classe atomique et le singleton contenant le mot 1 en est une autre.

Par abus de langage, on parlera généralement de *la* classe atomique, notée $\mathcal{Z}$, bien que celle-ci renvoie à plusieurs réalités combinatoires.

La série génératrice ordinaire de $\mathcal{Z}$ est X.

De la même manière, on appellera un objet de taille 0 un *objet neutre* et la classe associée la *classe neutre*, notée $\mathbf{1}$. Sa série génératrice ordinaire est 1.

On peut bien sûr définir ainsi n'importe quel ensemble fini. Par exemple, les mots sur un alphabet à deux lettres sont les suites d'objets atomiques a et b. La classe constituée de deux objets de taille 1 distincts est notée $2\mathcal{Z}$, de série génératrice $2X$.

1.3.2 Produit cartésien et union disjointe

Soient $\mathcal{A}$ et $\mathcal{B}$ deux classes combinatoires. Le *produit cartésien* de $\mathcal{A}$ et $\mathcal{B}$, noté $\mathcal{A} \times \mathcal{B}$, est la classe combinatoire $\mathcal{C}$ des couples d'objets de $\mathcal{A}$ et $\mathcal{B}$. Plus formellement,

$$\mathcal{C} = \{\ (a, b) \mid a \in \mathcal{A}, b \in \mathcal{B}\}$$

La taille du couple ainsi formé est la somme des tailles de ses deux composantes, *i.e.*, $|(a, b)| = |a| + |b|$. Combinatoirement, cela signifie que le nombre total d'atomes dans l'objet est la somme des nombres d'atomes dans ses sous-parties. Par exemple, le nombre total de nœuds d'un arbre binaire, qui est le produit d'un atome (la racine) et de deux sous-arbres, est la nombre de nœuds de son sous-arbre gauche, plus celui de son sous-arbre droit, plus un pour la racine.
La série génératrice de $\mathcal{C}$ est :

$$\begin{aligned} C(X) =& A(X) \cdot B(X) \\ =& \sum_{n\geq 0} \sum_{i=0}^{n} a_i b_{n-i} X^n \end{aligned}$$

Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes combinatoires. L'*union disjointe* de $\mathcal{A}$ et $\mathcal{B}$, notée $\mathcal{A} + \mathcal{B}$, est une classe combinatoire $\mathcal{D}$, constituée de tous les objets de $\mathcal{A}$ et tous les objets de $\mathcal{B}$ avec la mémoire de leur origine, c'est-à-dire que si un élément apparaît à la fois dans $\mathcal{A}$ et dans $\mathcal{B}$ on garde les deux exemplaires et on les distingue. Plus formellement,

$$\mathcal{D} = (A \times \{0\}) \cup (B \times \{1\}).$$

On omettra par la suite le marquage des objets. En fait, on se contente en général de considérer l'ensemble des objets d'une classe combinatoire comme un multi-ensemble, et ce problème de non-ambiguïté n'apparait alors plus de cette manière (mais il faut toujours veiller à ne pas inclure plusieurs fois un objet que l'on voudrait ne voir qu'une fois).

La série génératrice de $\mathcal{D}$ est :

$$D(X) = A(X) + B(X)$$

Par exemple, la classe $2\mathcal{Z}$ qui contient deux objets de taille 1 peut être vue comme $\mathcal{Z} + \mathcal{Z}$, l'union disjointe de deux classes à un élément de taille 1. Sa série génératrice est $2X$.

Avec ces deux opérateurs, on peut définir les classes algébriques. Ce sont les même que celles définies par les langages hors contexte non-ambigus.

Exemple. Reprenons ici encore l'exemple de la classe $\mathcal{L}$ des mots à deux lettres. On a

$$\mathcal{L} = 1 + \mathcal{Z} \times \mathcal{L} + \mathcal{Z} \times \mathcal{L}$$

qui est l'équivalent de la grammaire

$$L = \varepsilon | aL | bL.$$

On en déduit que la série génératrice $L(x)$ de $\mathcal{L}$ vérifie

$$L(X) = 1 + 2XL(X),$$

d'où on retrouve

$$L(X) = \frac{1}{1 - 2X}.$$

Des exemples sur les arbres seront également développés dans le chapitre 2.

1.3.3 Séquence

L'opérateur *séquence*, noté SEQ peut se définir à partir de l'union disjointe et du produit cartésien. Comme c'est un opérateur très courant, on préfère disposer d'un constructeur propre. Soit $\mathcal{A}$ une classe combinatoire sans objet de taille 0 :

$$\textsc{Seq}\,\mathcal{A} = \mathbf{1} + \mathcal{A} \times \textsc{Seq}\,\mathcal{A}.$$

On en déduit immédiatement que :

$$\textsc{Seq}\,\mathcal{A} = \sum_{k\geq 0} \mathcal{A}^k.$$

La série génératrice de $\mathcal{B} = \textsc{Seq}\,\mathcal{A}$ est :

$$B(X) = \frac{1}{1 - A(X)}.$$

Cette notation formelle appelée *quasi-inverse* représente le développement limité de $B(X)$, c'est à dire $B(X) = 1 + A(X) + A(X)^2 + A(X)^3 + ... + A(X)^n + ...$

Par exemple, la classe $\mathcal{L}$ des mots sur un alphabet à deux lettres se décompose comme $\textsc{Seq}\,2\mathcal{Z}$. On retrouve bien que la série génératrice de $\mathcal{L}$ est

$$L(X) = \frac{1}{1 - 2X}.$$

Soit E un ensemble d'entiers naturels. On définit $\textsc{Seq}_E$ comme l'ensemble des séquences de longueurs incluses dans E :

$$\textsc{Seq}_E\,\mathcal{A} = \sum_{k\in E} \mathcal{A}^k.$$

Remarque. Il n'est pas possible de faire une séquence d'une classe contenant un objet de taille nulle. En effet, on pourrait sinon faire un n-uplet aussi long qui l'on souhaite de cet objet de taille nulle, et l'on obtiendrait un nombre infini d'objets de taille nulle (en fait, un nombre infini d'objets de toutes les tailles pour lesquels au moins un objet existe). L'ensemble ainsi obtenu ne constituerait donc pas une classe combinatoire.

1.3.4 Produit de Hadamard

Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes combinatoires. Le *produit de Hadamard* de $\mathcal{A}$ et $\mathcal{B}$, noté $\mathcal{A} \odot \mathcal{B}$ est l'ensemble des couples d'objets (a, b) où a et b sont de même taille et $a \in \mathcal{A}$ et $b \in \mathcal{B}$. On note alors $\mathcal{H} = \mathcal{A} \odot \mathcal{B}$, on a donc :

$$\mathcal{H} = \{(a, b) | a \in \mathcal{A}, b \in \mathcal{B}, |a| = |b|\}.$$

La taille du couple est la taille de son premier élément (qui est la même que celle du deuxième).

La série génératrice de $\mathcal{H}$ est :

$$H(X) = \sum_{n\geq 0} a_n b_n X^n.$$

Le produit de Hadamard est très classique en théorie des espèces où il a été introduit par Joyal dès le fondement de cette théorie [Joy81] dans le cas étiqueté. En combinatoire analytique, dans le cas général, la série génératrice obtenue est moins simple à analyser que pour les autres opérateurs de base. Il faut cependant remarquer que les classes holonomes sont closes par produit de Hadamard et qu'ils apparaissent dans plusieurs études récentes sur l'holonomie (voir par exemple [Mis05]).

1.3.5 Cycles, Multiset et opérateurs de Pólya

Les *opérateurs de Pólya* sont des constructeurs qui introduisent des symétries sur des objets : il s'agit de séquences d'objets vus à une équivalence près. Les opérateurs les plus classiques sont les cycles et les multi-ensembles, mais il est également possible de considérer d'autres groupes de Coxeter [PRA11, PTW09].

L'opérateur *cycle* d'une classe combinatoire $\mathcal{A}$ sans objet de taille nulle, noté $\textsc{Cyc}(\mathcal{A})$, est l'ensemble des séquences non vides d'objets de $\mathcal{A}$ vues à permutations circulaires près. La taille d'un cycle est la taille d'un de ses représentants. La série génératrice de $\mathcal{C} = \textsc{Cyc}(\mathcal{A})$ est :

$$C(X) = \sum_{k\geq 1} \frac{\varphi(k)}{k} \ln \frac{1}{1 - A(X^k)}$$

où φ est la fonction indicatrice d'Euler.

L'opérateur *multi-ensemble* d'une classe combinatoire $\mathcal{A}$ sans objet de taille nulle, noté $\textsc{MSet}(\mathcal{A})$, est l'ensemble des séquences d'objets de $\mathcal{A}$ vues à permutations près. Une façon de le voir est de fixer un ordre canonique sur les objets de $\mathcal{A}$, puis de faire une séquence d'objets de $\mathcal{A}$, potentiellement répliqués plusieurs fois, dans leur ordre canonique. Autrement dit :

$$\textsc{MSet}\,\mathcal{A} = \prod_{a\in\mathcal{A}} \textsc{Seq}\{a\}.$$

La taille d'un multi-ensemble est la taille d'un de ses représentants. La série génératrice de $\mathcal{L} = \textsc{MSet}(\mathcal{A})$ est :

$$M(X) = \exp\left(\sum_{k\geq 1} \frac{1}{k} A(X^k)\right).$$

Comme pour la séquence, on note $\textsc{Cyc}_E$ (resp. $\textsc{MSet}_E$) l'ensemble des cycles (resp. multi-ensemble) de longueurs incluses dans E. Soit $\mathcal{C}_E = \textsc{Cyc}_E(\mathcal{A})$, on a :

$$C_E(X) = \sum_{k\in E} \frac{\varphi(k)}{k} \ln \frac{1}{1 - A(X^k)}.$$

1.4 Classes étiquetées et séries exponentielles

Une *classe étiquetée* est une classe combinatoire où les atomes d'un objet sont distingués. Une façon de le voir est, pour un objet de taille n, d'associer à chaque atome une étiquette entre 1 et n, chaque étiquette n'apparaissant qu'une seule fois. Notons tout de même que l'on introduit ainsi un ordre linéaire sur les étiquettes [1] (exploité par exemple pour compter les records d'une permutation).

Dans le cas étiqueté, l'usage des séries exponentielles se révèle souvent plus adéquat. La *série génératrice exponentielle* (ou *EGF p*our Exponential Generating Function) de $\mathcal{A}$, notée $\hat{A}$, est défini comme :

$$\hat{A}(X) = \sum_{n\geq 0} \frac{a_n}{n!} X^n$$

où a_n est le nombre d'objets étiquetés de taille n.

1.4.1 Somme, produit, séquence

La somme est définie comme dans le cas non-étiqueté.

Le *produit cartésien étiqueté* de deux classes étiquetées $\mathcal{A}$ et $\mathcal{B}$ est l'ensemble des couples d'objets de $\mathcal{A}$ et d'objets de $\mathcal{B}$ réétiquetés sur leur taille totale, en conservant leur ordre d'origine.

Exemple. On considère le produit de deux objets, $(1,2)$ et $(3,1,2)$, tous deux appartenants à $\textsc{Seq}\,\mathcal{Z}$.

$$\begin{aligned}(1,2)\times(3,1,2) = \{\,&((1,2),(5,3,4)),\\ &((1,3),(5,2,4)),\\ &((1,4),(5,2,3)),\\ &((1,5),(4,2,3)),\\ &((2,3),(5,1,4)),\\ &((2,4),(5,1,3)),\\ &((2,5),(4,1,3)),\\ &((3,4),(5,1,2)),\\ &((3,5),(4,1,2)),\\ &((4,5),(3,1,2))\}\end{aligned}$$

Soit $\mathcal{C} = \mathcal{A}\times\mathcal{B}$ le produit cartésien étiqueté de $\mathcal{A}$ et $\mathcal{B}$, la série génératrice exponentielle de $\mathcal{C}$ est :

$$\hat{C}(X) = \hat{A}(X)\hat{B}(X).$$

1. Cette supposition n'est pas restrictive et ne modifie pas les résultats. Un ordre linéaire est équivalent à donner, en plus de la structure, une séquence des mêmes étiquettes ; ce qui revient à faire un produit de Hadamard avec une classe de série $\sum_{n\geq 0} 1x^n$.

La définition de la séquence étiquetée en découle directement. Soit $\mathcal{A}$ une classe combinatoire étiquetée sans objet de taille 0, soit $\mathcal{S} = \textsc{Seq}\,\mathcal{A}$, la série génératrice exponentielle de $\mathcal{S}$ est :

$$\hat{S}(X) = \frac{1}{1 - \hat{A}(X)}.$$

1.4.2 Produit de Hadamard étiqueté

Soit deux classes $\mathcal{A}$ et $\mathcal{B}$ étiquetées sur le même ensemble E, le *produit de Hadamard étiqueté* de $\mathcal{A}$ et $\mathcal{B}$, noté $\mathcal{A} \odot \mathcal{B}$, est l'ensemble des couples de même taille. La taille du couple est la taille de ses parties. Plus formellement :

$$A \odot \mathcal{B} = \{(a, b) | a \in \mathcal{A}, b \in \mathcal{B} \text{ et} |a| = |b|\}$$

avec $|(a, b)| = |a| = |b|$. Notons que les étiquettes originales sont conservées, chaque étiquette apparaissant donc dans chacun des deux éléments du couple.

Soit $\mathcal{C} = \mathcal{A} \odot \mathcal{B}$,

$$\hat{C}(X) = \sum_{n \geq 0} \frac{a_n b_n}{n!} X^n$$

La définition d'un atome à l'intérieur d'un produit de Hadamard demande des précautions : un atome est l'ensemble de tout les éléments de même étiquette (qui doivent donc être dans des parties différentes du couple).

Ce produit a été introduit dans [Joy81] dans le cadre de la théorie des espèces.

1.4.3 Rigidité

Une classe combinatoire $\mathcal{A}$ est dite *rigide* si à chaque objet de taille n non étiqueté correspondent $n!$ objets étiquetés. En particulier la série génératrice ordinaire de la classe non-étiquetée est la même que la série exponentielle de la classe étiquetée. Cela revient à dire que ses objets n'ont pas de symétries.

Toute classe algébrique est rigide : les opérateurs $+, \times, \textsc{Seq}$ conservent la rigidité. Le produit de Hadamard conserve également la rigidité. On vérifie facilement cette propriété en comparant les formes des séries génératrices ordinaires et exponentielles associées.

Dans la suite nous définissons les opérateurs ne conservant pas la rigidité dans le cas étiqueté.

1.4.4 Cycles et Ensembles

Comme dans le cas non-étiqueté, on peut définir des séquences à équivalences près. Dans le cas étiqueté, comme tous les atomes sont différents, ces équivalences créent des automorphismes triviaux et donc toutes les structures de même longueur auront le même nombre de représentants.

Soit $\mathcal{A}$ une classe combinatoire étiquetée. La série génératrice exponentielle de la classe étiquetée $\mathcal{C} = \textsc{Cyc}\,\mathcal{A}$ est :

$$\hat{C}(X) = \ln \frac{1}{1 - \hat{A}(X)}.$$

L'opérateur ensemble est l'opérateur séquence vu à permutation près. Contrairement au cas non-étiqueté un même objet ne peut pas être répliqué plusieurs fois, puisque si c'était le cas alors chacune des occurrences aurait des étiquettes différentes et serait donc distinguées. La série génératrice exponentielle de la classe étiquetée $\mathcal{E} = \textsc{Set}\,\mathcal{A}$ est :

$$\hat{E}(X) = \exp(\hat{A}(X)).$$

1.5 Classes Multivariées

On a défini dans la section 1.3.1 une classe atomique et une classe neutre. Or, dans beaucoup d'applications on a envie de pouvoir distinguer certaines catégories d'atomes. Par exemple, pour un arbre unaire-binaire on peut vouloir distinguer les nœuds unaires des nœuds binaires. Pour cela, on peut ajouter une marque de poids nul, ce qui revient à faire un produit cartésien avec une nouvelle classe neutre, notée $\mathcal{U}$.

Dans le cas multivarié, on peut vouloir, pour plus de clarté, indiquer explicitement les dépendances aux classes élémentaires. Par exemple, soit la classe $\mathcal{A}$ définie à partir d'une classe atomique $\mathcal{Z}$, d'une classe neutre 1 et d'une classe neutre marquée $\mathcal{U}$, on la note également $\mathcal{A}(\mathcal{Z}, \mathbf{1}, \mathcal{U})$ ou $\mathcal{A}(\mathcal{Z}, \mathcal{U})$, la classe neutre non-marquée étant souvent omise.

Exemple. Soit $\mathcal{A}(\mathcal{Z}, \mathcal{U})$ la classe des arbres enracinés planaires unaires-binaires (*i.e.*, dont tous les nœuds internes ont un ou deux fils) ayant pour taille le nombre de nœuds internes et dont les nœuds unaires sont marqués par $\mathcal{U}$. Une spécification de $\mathcal{A}$ est :

$$\mathcal{A}(\mathcal{Z}, \mathcal{U}) = \mathbf{1} + \mathcal{Z} \times \mathcal{A}(\mathcal{Z}, \mathcal{U}) \times \mathcal{A}(\mathcal{Z}, \mathcal{U}) + \mathcal{U} \times \mathcal{Z} \times \mathcal{A}(\mathcal{Z}, \mathcal{U}).$$

Notons que l'on peut également compter de cette manière des configurations qui ne sont pas associées à un atome. Dans l'exemple ci-dessus on pourrait par exemple vouloir compter les arbres unaires-binaires selon deux paramètres : leur nombre de nœuds internes et leur nombre de feuilles. La spécification deviendrait alors :

$$\mathcal{A}(\mathcal{Z}, \mathcal{U}) = \mathcal{U} + \mathcal{Z} \times \mathcal{A}(\mathcal{Z}, \mathcal{U}) \times \mathcal{A}(\mathcal{Z}, \mathcal{U}) + \mathcal{Z} \times \mathcal{A}(\mathcal{Z}, \mathcal{U}).$$

À partir de là, on peut définir une série bivariée (ou multi-variée si on veut utiliser plusieurs marques différentes). Soit une classe $\mathcal{A}(\mathcal{Z}, \mathcal{U})$, sa série génératrice ordinaire est :

$$A(x, u) = \sum_{n,k \geq 0} a_{n,k} x^n u^k$$

où $a_{n,k}$ est le nombre d'objets de $\mathcal{A}(\mathcal{Z},\mathcal{U})$ de taille n ayant k marques $\mathcal{U}$.
De même, sa série génératrice exponentielle est :

$$\hat{A}(x,u) = \sum_{n,k\geq 0} \frac{a_{n,k}}{n!} x^n u^k.$$

On peut extraire la série génératrice monovariée des objets sans marque de cette série en posant $u = 1$ ou interdire les configurations qui possèdent des marques en posant $u = 0$.

1.6 Dérivée combinatoire

La notion de dérivée combinatoire est centrale pour cette thèse. On définit ici la dérivée des classes étiquetées, ou non-etiquetées rigides. Le cas non-étiqueté comportant des opérateurs de Pólya est plus compliqué et ne rentre pas dans le cadre de cette thèse, mais a été étudié dans [BFKV07].

La dérivée d'une classe combinatoire $\mathcal{A}$ est l'ensemble des objets de $\mathcal{A}$ dont un atome a été retiré, et où l'on a marqué l'endroit d'où il a été retiré.

L'opérateur de *pointage* d'une classe consiste à la dériver puis à en faire un produit cartésien par $\mathcal{Z}$: de cette manière, on obtient la classe des objets dont un atome est distingué (celui qui a été retiré puis remis).

La dérivation et le pointage conservent la rigidité des classes.

1.6.1 Règles de dérivation

Comme pour les fonctions, il y a des règles de dérivation des classes combinatoires qui se composent. Que ce soit dans le cas étiqueté ou non-étiqueté rigide on a les règles suivantes pour l'opérateur point, noté $^\bullet$:

- $\mathbf{1}^\bullet = \varnothing$: on ne peut pas distinguer un atome d'une classe d'objet sans atome,
- $(\mathcal{A}+\mathcal{B})^\bullet = \mathcal{A}^\bullet + \mathcal{B}^\bullet$: un objet pointé dans l'union disjointe de $\mathcal{A}$ et $\mathcal{B}$ est soit un objet pointé de $\mathcal{A}$ soit un objet pointé de $\mathcal{B}$,
- $(\mathcal{A}\times\mathcal{B})^\bullet = \mathcal{A}^\bullet \times \mathcal{B} + \mathcal{A}\times\mathcal{B}^\bullet$: un couple pointé peut l'être soit sur sa première soit sur sa deuxième partie,
- $(\textsc{Seq}\,\mathcal{A})^\bullet = \textsc{Seq}\,\mathcal{A} \times \mathcal{A}^\bullet \times \textsc{Seq}\,\mathcal{A}$: une séquence pointée a un de ses éléments qui est pointé. Les éléments d'avant et d'après ne le sont pas.
- $(\textsc{Seq}_E\,\mathcal{A})^\bullet = \sum_{\substack{i,j \\ i+j+1\in E}} \textsc{Seq}_i\,\mathcal{A} \times \mathcal{A}^\bullet \times \textsc{Seq}_j\,\mathcal{A}$,
- $(\textsc{Cyc}\,\mathcal{A})^\bullet = \mathcal{A}^\bullet \times \textsc{Seq}\,\mathcal{A}$: pointer un cycle permet d'un distinguer le premier sous-objet (celui sur lequel est le point), ce qui nous donne une séquence dont le premier élément est pointé,
- $(\textsc{Cyc}_E\,\mathcal{A})^\bullet = \mathcal{A}^\bullet \times \textsc{Seq}_{E-1}\,\mathcal{A}$ où $E-1 = \{i | i+1 \in E, i \neq -1\}$,
- $(\textsc{Set}\,\mathcal{A})^\bullet = \mathcal{A}^\bullet \times \textsc{Set}\,\mathcal{A}$: pointer un multi-ensemble permet d'en distinguer un sous-objet, celui qui est pointé. Ceci n'induit pas d'ordre sur les autres éléments.
- $(\textsc{Set}_E\,\mathcal{A})^\bullet = \mathcal{A}^\bullet \times \textsc{Set}_{E-1}\,\mathcal{A}$ où $E-1 = \{i | i+1 \in E, i \neq -1\}$,

– $(\mathcal{A}\odot\mathcal{B})^\bullet = \mathcal{A}^\bullet\odot\mathcal{B} = \mathcal{A}\odot\mathcal{B}^\bullet$: le produit de Hadamard est l'ensemble des couples de structures sur le même ensemble d'atomes. Si un atome est distingué, on peut considérer qu'il a été distinguer dans $\mathcal{A}$ et transmis à $\mathcal{B}$ ou l'inverse.

Ces relations se traduisent automatiquement sur les séries génératrices. Elles restent vraies en remplaçant l'opérateur point par une dérivée.

1.6.2 Dérivées partielles

Dans le cas de classes multi-variées, on peut également vouloir distinguer une marque. On peut pour cela utiliser des dérivées partielles sur la marque concernée.

Exemple. Une spécification de la classe $\mathcal{A}(\mathcal{Z},0,0)$ des mots sur l'alphabet $\{a,b\}$ dont un a est distingué, est :

$$\begin{aligned}\mathcal{B}(\mathcal{Z},\mathcal{U}_a,\mathcal{U}_b) =& \mathcal{U}_a\mathcal{Z}\mathcal{B}(\mathcal{Z},\mathcal{U}_a,\mathcal{U}_b) + \mathcal{U}_b\mathcal{Z}\mathcal{B}(\mathcal{Z},\mathcal{U}_a,\mathcal{U}_b\\ \mathcal{A}(\mathcal{Z},\mathcal{U}_a,\mathcal{U}_b) =& \frac{\partial\mathcal{B}(\mathcal{Z},\mathcal{U}_a,\mathcal{U}_b)}{\partial\mathcal{U}_a}\end{aligned}$$

1.7 Analogie avec la théorie des espèces

Dans le formalisme différent de la théorie des espèces, ces idées ont été introduites par Joyal [Joy81] en 1981, et reprises par Bergeron Labelle Leroux dans [BLL98]. Le cadre naturel est alors le cadre étiqueté.

Une *espèce* est un foncteur F qui associe à un ensemble E de taille n (*i.e.*, l'ensemble des atomes dans le formalisme Flajolet) un ensemble de structures sur cet ensemble (*i.e.*, l'ensemble des objets de taille n de la classe combinatoire dans le formalisme Flajolet). On appelle les éléments de cet ensemble des F-structures.

Le produit étiqueté y trouve alors une définition simple : il suffit de prendre deux ensembles à supports disjoints pour chacune des 2 F-structures dont on veut faire le produit. Si l'on considère que l'on a un ordre total sur l'union de ces deux supports, on retrouve la définition du produit étiqueté donnée plus haut.

C'est un cadre très agréable pour parler de dérivée combinatoire et de produit de Hadamard : la dérivée est un foncteur qui associe à une structure sur E une structure sur $E\cup\{\star\}$. Le produit de Hadamard étiqueté consiste à associer deux F-structures à l'ensemble E.

Pour parler de classe non-étiquetée on doit ensuite voir l'ensemble muni de sa structure à automorphisme près. On peut également utiliser d'autres classes d'équivalence pour parler d'opérateurs de Pólya ou de classes colorées.

Chapitre 2

Arbres

Les arbres, sous différentes formes, apparaissent dans presque tous les chapitres de cette thèse. Aussi, nous avons choisi de consacrer un chapitre de ces préliminaires à leurs définitions et à exposer des faits simples qui nous seront utiles dans la suite. De plus, c'est une première occasion de rencontrer un objet combinatoire et d'en décrire différentes propriétés.

Au sens de la théorie des graphes, les arbres sont des graphes acycliques. Ces objets classiques ont fait l'objet de nombreuses études dans ce cadre mais nous allons nous intéresser ici à des familles d'arbres plus proches des structures informatiques.

2.1 Arbres enracinés planaires

Un arbre sera pour nous une structure récursive finie, qui se compose comme suit. Un arbre T est soit :

- une *feuille*, *i.e.*, un atome.
- un *nœud interne* n, *i.e.*, un tuple composé d'un atome et d'une séquence d'arbres. La longueur de la séquence est appelée *arité* du nœud. Les arbres de la séquence sont appelés *sous-arbres* de T issus de n.

Les arbres de cette famille seront appelés *arbres enracinés planaires* : en effet, si on se replace dans le cadre de la théorie des graphes, cette définition correspond à celles des graphes plongés (ou cartes) acycliques enracinés. Un informaticien remarquera que cette définition n'est rien d'autre que celle des structures définies par pointeurs, pour un peu que l'on autorise les atomes à porter des informations.

Soit $T = (x, T_1, T_2, ..., T_k)$ un arbre. La *racine* de l' arbre est x. Les arbres T_1, T_2,... ,T_k sont appelés *fils* de $\mathcal{Z}_1$. Réciproquement, $\mathcal{Z}_1$ est le *père* de T_1, de T_2,... et de T_k. L'ensemble des *ancêtres* (resp. *descendants*) est l'ensemble obtenu par clôture transitive de la relation père (*resp.* fils[1]). Le *sous-arbre enraciné* en un nœud y est l'arbre de T ayant pour racine y. La figure 2.1 illustre ces définitions.

1. Comme la relation fils fait correspondre un nœud à des arbres, il faut assimiler pour pouvoir faire la clôture transitive un sous-arbre et sa racine.

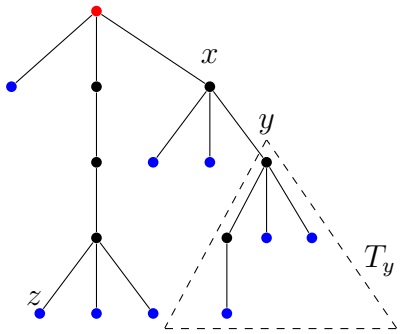


FIGURE 2.1 – Un arbre enraciné planaire à 7 nœuds internes et 9 feuilles. Tous les nœuds internes ont une arité de 1 ou 3. Sa racine s apparaît en rouge, ses feuilles en bleu. x, y et z sont des nœuds de l'arbres, x et y sont des nœuds internes, z est une feuille. Le sous-arbre enraciné en y est T_y.

La taille d'un arbre est définie comme le nombre total de nœuds, internes ou feuilles, qu'il contient. La spécification combinatoire correspondante est :

$$\mathcal{A} = \mathcal{Z} + \mathcal{Z} \times \textsc{Seq}\, \mathcal{A}$$

Dans le cas général, on ne peut pas se contenter de compter uniquement les nœuds internes ou uniquement les feuilles : il existerait alors un nombre infini d'objets de taille donnée. En effet, dans le cas où l'on ne compte que les nœuds internes on peut définir, pour tout n, un arbre ayant une racine d'arité n et dont tous les fils sont des feuilles, qui auraient tous taille 1. Dans le cas où l'on ne compte que les feuilles on peut définir, pour tout n, un arbre constitué d'une chaine de n nœuds unaires suivis d'une feuille et on obtient également un nombre infini d'objets de taille 1. Il faut donc être prudent lors de la définition de la taille d'une structure combinatoire.

Nous allons maintenant présenter des faits triviaux, mais néanmoins utiles, sur les arbres enracinés planaires.

Fait 1. Le degré d'un nœud est égal à son arité si le nœud est la racine de l'arbre, et à l'arité plus un sinon.

On peut définir plusieurs familles d'arbres en restreignant l'arité des nœuds à appartenir à un ensemble donné. Cette restriction est très naturelle du point de vue informatique. On peut fixer l'arité des nœuds internes à un certain ensemble $E \subseteq \mathbb{N}^*$, ce qui mène à la spécification pour la classe des arbres enracinés planaires dont tous les nœuds internes ont une arité dans E munie de la fonction taille nombre totale de nœuds, notée $\mathcal{T}_E$:

$$\mathcal{T}_E = \mathcal{Z} + \mathcal{Z} \times \sum_{i \in E} \mathcal{T}_E^i.$$

Dans le cas où E est fini, on peut prendre le nombre de nœuds internes comme fonction taille en restant consistant avec la définition de classe combinatoire, ce qui donnerait la spécification :

$$\widetilde{\mathcal{T}_E} = 1 + \mathcal{Z} \times \sum_{i \in E} \widetilde{\mathcal{T}_E}^i,$$

et, dans le cas où $1 \notin E$, on peut également utiliser le nombre de feuilles comme fonction taille.

Fait 2. Pour tout $E \subseteq \mathbb{N}^*$, la classe $\mathcal{T}_E$ est rigide.

Ce fait découle directement de sa spécification. Remarquons qu'il reste vrai pour les autres définitions de la taille évoquées dans ce chapitre.

Fait 3. Soit $k \in \mathbb{N}$. Un arbre de $T_{\{k\}}$ à n nœuds internes aura $(k-1)n+1$ feuilles.

Soit T en arbre, v un nœud de cet arbre. Nous distinguons ici les notions de hauteur et de profondeur d'un nœud dans un arbre. La *hauteur* du nœud v dans T est :

- 0 si v est une feuille,
- $1 + \max_{S \text{ fils de} T}$ hauteur S sinon.

La hauteur de T est la hauteur de sa racine.
La *profondeur* de v est :

- 0 si v est la racine ,
- 1+ la profondeur du père de v sinon.

La profondeur de T est le maximum sur tous ses nœuds de leur profondeur (et est égale à la hauteur).
La largeur d'un arbre est le maximum du nombre de nœuds de même profondeur.

Pour tout $E \subseteq \mathbb{N}^*$, la hauteur moyenne des arbres de $\mathcal{T}_E$ est en $O(\sqrt{n})$, ou n est la nombre total de nœud de l'arbre (voir [DBRK71] ou [FS09a, p.329]).

2.2 Arbres de Catalan

Les *arbres de Catalan*, ou *arbres binaires* sont les arbres enracinés planaires dont tous les nœuds internes ont une arité de 2. Il s'agit de la classe la plus simple contenant effectivement un phénomène de branchement. Ces arbres ont été très largement étudié depuis très longtemps, et peuvent être rapprochés des processus de Galton-Watson finis. Ils sont comptés par la suite de Catalan $(1, 2, 5, 14, 42, 132, 429, 1430, 4862, ...)$ qui comptent également un très grand nombre de structures combinatoires variées, pour lesquels on connait de nombreuses bijections explicites (arbres planaires enracinés généraux, parenthèsages,...). Les termes de la suite de Catalan sont

$$C_n = \binom{2n}{n} \cdot \frac{1}{n+1}$$

On utilise habituellement le nombre de nœuds internes comme taille pour les arbres, afin de vraiment coïncider avec la suite de Catalan (sinon, un coefficient sur deux est nul, d'après le fait 3). Cependant, dans les parties suivantes, nous allons utiliser le nombre total de nœuds de l'arbre comme taille (notamment pour pouvoir utiliser un opérateur de pointeur simple permettant de distinguer indifféremment un nœud interne ou une feuille). La fonction taille utilisée sera rappelée lors de l'introduction des arbres dans chacun des chapitres où ils apparaissent.

Une spécification des arbres de Catalan comptés selon leur nombre de nœuds internes est :

$$\mathcal{C} = \mathbf{1} + \mathcal{Z} \times \mathcal{C}^2$$

La série génératrice (ordinaire ou exponentielle) des arbres de Catalan est l'unique solution analytique en 0 de l'équation :

$$C(X) = 1 + X \cdot C(X)^2$$

. On obtient donc :

$$C(X) = \frac{1 - \sqrt{1 - 4X}}{2X}$$

Lorsque n tend vers l'infini, on a l'équivalence suivante pour le nombre d'arbres de Catalan à n nœuds internes :

$$C_n \sim 4^n n^{-\frac{3}{2}}.$$

Ce résultat peut être prouvé par la combinatoire analytique (voir [FS09a, Section 1.1] pour plus de détails) ou simplement en utilisant la formule de Stirling.

2.3 Arbres de Motzkin

Les *arbres de Motkzin*, ou *arbres unaires-binaires* sont les arbres enracinés planaires dont tous les nœuds internes ont une arité de 1 ou de 2. Il s'agit également d'une classe largement étudiée, pour laquelle il existe de nombreuses bijections (mais moins que pour Catalan). Les premiers termes de la suite sont : $1, 1, 2, 4, 9, 21, 51, 127, 323, 835, 2188, 5798$. La taille considérée ici est le nombre total de nœuds. Il n'existe pas de forme close simple pour les coefficients mais une expression obtenue par inversion de Lagrange : $M_n = \frac{1}{n} \sum_{k=0}^{n} \binom{n}{k} \binom{n-k}{k-1}$.

Une spécification des arbres de Motzkin est :

$$\mathcal{M} = \mathcal{Z} + \mathcal{Z} \times \mathcal{M} + \mathcal{Z} \times \mathcal{M}^2$$

Le série génératrice de Motzkin est :

$$M(X) = \frac{1 - X - \sqrt{(1 + X)(1 - 3X)}}{2X}$$

Lorsque n tend vers l'infini, on a l'équivalence suivante pour le nombre d'arbres de Motzkin à n nœuds :

$$M_n \sim 3^n n^{-\frac{3}{2}}.$$

2.4 Autres types d'arbres enracinés

Il existe de nombreux autres types d'arbres enracinés, qu'ils soient planaires ou non. Par exemple les arbre binaires de recherche (ABR) ou les arbres croissants qui sont des arbres enracinés planaires étiquetés avec des contraintes sur l'ordre des étiquettes. Ces arbres ont des comportements asymptotiques différents de ceux des arbres généraux ou à arités fixées.

On peut également relâcher la contrainte de planarité, c'est voir la séquence des fils d'un nœud à équivalence près. Il s'agit des arbres d'Otter, des mobiles, des arbres de Cayley,...

Signalons également que de nombreux processus de branchements probabilistes peuvent, dans les cas finis, être vus comme des classes combinatoires pondérées. C'est par exemple le cas des processus de Galton-Watson et des arbres de Ford [For06].

Chapitre 3

Génération aléatoire

3.1 Introduction à la génération aléatoire

Un générateur aléatoire de structures combinatoires est un algorithme qui tire aléatoirement un objet a d'une classe combinatoire $\mathcal{A}$ selon une distribution donnée. Le cas le plus classique est la génération uniforme en taille exacte : l'algorithme prend un entier n en entrée et renvoie un objet de $\mathcal{A}$ de taille n de telle manière que tous les objets de taille n de $\mathcal{A}$ aient la même probabilité d'être tirés.

Un intéret de la génération uniforme est de savoir à quoi ressemble un objet typique de grande taille de la classe qui nous intéresse. Pour les petites tailles, on pourra préférer la génération exhaustive, rapidement impossible pour toutes les classes dont le nombre d'objet croît rapidement (de très nombreuses classes combinatoires croissent exponentiellement vites) par rapport à la taille. Pour nous, un objet de grande taille est en général un objet ayant de quelques centaines à quelques millions d'atomes, en fonction de la complexité de la structure (et de la capacité à le dessiner et d'y voir quelque chose).

Selon les classes regardées, les applications peuvent être variées. Tout d'abord, pouvoir échantillonner des grands objets combinatoires permet d'obtenir des conjectures sur la distribution moyenne de certains paramètres ou sur des propriétés limites. La génération aléatoire procure également un moyen de tester empiriquement la complexité moyenne d'algorithmes prenant en entrée des données structurées. Un autre type d'application est le raffinement de modèles : on connaît un ensemble de règles (physiques par exemple) qui donne de la structure à un objet. Si on génère un objet selon ces règles, ressemble-t-il aux objets observés dans la réalité ? Si non, c'est que les système de règles est incomplet (ou faux).

Nous reviendrons sur les applications à la fin de cette section.

3.1.1 Bits et nombres aléatoires

La génération de nombres pseudo-aléatoires n'a pas grand chose à voir avec une génération de structures aléatoires. Elle se fait à base de suites chaotiques, initialisées à partir d'une *graine*, généralement dépendante du temps et de la machine, permettant

de différencier chaque instance.

Les suites de nombres aléatoires produites par un ordinateur sont pseudo-aléatoires, en particulier elles sont souvent périodiques. Dans toute la suite on considèrera qu'un ordinateur est capable de donner une suite infinie de bits aléatoires non corrélés. En pratique, la période est normalement plus grande que le nombres de bits aléatoires utilisés pour engendrer un ensemble d'objets de grande taille.

Remarque. Il existe aujourd'hui des générateurs aléatoires basés sur des instruments de mesure haute précision de grandeurs physiquement fluctuantes (telle que la température en un point).

3.1.2 Complexité

On parle ici d'algorithmes, on va donc très naturellement se poser la question de la complexité associée. Cette complexité, comme les algorithmes de génération aléatoire, se décompose souvent en deux temps : complexité du pré-calcul et complexité du tirage. En effet, lorsque l'on fait de la génération aléatoire on va souvent avoir envie de tirer plusieurs grandes structures de même taille. On peut donc accepter un pré-calcul plutôt coûteux, si on peut ensuite obtenir une structure efficacement. Pour le tirage, il y a trois types de complexité qui nous intéressent :

- *complexité en temps* : le nombre d'opérations unitaires (la définition d'opération unitaire varie en fonction du modèle de complexité utilisé),
- *complexité en espace* : la taille des structures stockées en mémoire pendant le déroulement de l'algorithme (on considère dans cette thèse une case mémoire comme unité, les pointeurs et les entiers sont considérés de taille constants) ; la sortie du pré-calcul est incluse dans cette complexité,
- *complexité en nombre de bits aléatoires* : le nombre de bits aléatoires utilisés par l'algorithme.

Comme on a besoin d'écrire les structures, les complexités en temps et en espace seront au moins linéaires, pour tout modèle de complexité raisonnable [1]. Pour le pré-calcul, seul les complexités en temps et espace font sens.

Pour les bits aléatoires, tous les modèles ne les considèrent pas comme une ressource privilégiée. Par exemple, pour certains générateurs de Boltzmann présenté un peu plus tard, on considère que l'on tire une variable aléatoire uniformément dans $[0, 1]$. En pratique, cela pose des problèmes de précision et de complexité bit-à-bit en temps mal maîtrisée. Ces problèmes ont déjà été considérés dans [FPS11, Duc11, Lum13], en ayant notamment recours à de l'évaluation paresseuse. Dans la partie 1 de cette thèse, nous attaquerons les choses sous cet angle : nous allons construire un algorithme qui n'utilise pas des lois de probabilité mais directement des bits aléatoires.

Notons ici que pour un générateur aléatoire uniforme de structures de taille n, le nombre minimal moyen de bits aléatoires requis est le logarithme en base 2 du nombre de structure de taille n (car il y en faut au moins autant que le nombre de bits nécessaire à différencier les structures). Ce nombre est appelé l'*entropie* de l'ensemble [Sha48].

1. C'est à dire, même haut-niveau

3.1.3 Génération exhaustive

La *génération exhaustive* de structures, *i.e.*, écrire explicitement toutes les structures d'une taille donnée, est également un problème intéressant et souvent difficile. Cependant, pour les structures dont le nombre croît exponentiellement, il est rapidement impossible de stocker toutes les structures en mémoire, même pour des tailles raisonnables.

Remarquons néanmoins que la génération exhaustive d'objets d'une taille donnée permet d'obtenir un générateur aléatoire optimal en nombre de bits aléatoires : en effet, il suffit alors de tirer un entier entre 1 et le nombre a_n de structures générées ce qui peut être fait avec $\lceil log_2(a_n) \rceil$ bits aléatoires.

3.2 Méthodes ad-hoc et méthode récursive

On désigne par *ad-hoc* un algorithme qui ne se base pas sur une méthode généralisable.

3.2.1 Knuth shuffle : génération aléatoire de permutations

Le *Knuth shuffle* [Knu97b] pour générer uniformément des permutations de $\{1, 2, ..., n\}$ est une méthode ad-hoc en temps linéaire (dans un modèle où tirer un entier aléatoire et échanger deux éléments d'un tableau se font en temps constant), utilisant $\log_2(n!)$ bits aléatoires. On considère que tirer un nombre aléatoire est une opération élémentaire. L'algorithme fonctionne comme ceci : on part d'un tableau (trié) de taille n , puis pour tout entier i allant de 1 à n on tire un entier j entre i et n et on échange le contenu des cases d'indices i et j (voir Algorithme 1).

Algorithm 1: Knuth Shuffle

Input: la taille n de la permutation
Output: une permutation uniforme de taille n.

```
1 T := [1, 2, .., n]
2 for i = 1 à n do
3 |   j := un entier uniforme entre i et n
4 |   Échanger T[i] et T[j]
5 return T
```

Cet algorithme est optimal en nombre de bits aléatoire utilisés : il y a $n!$ permutations de taille n, on a donc besoin de $\log_2(n!)$ bits pour pouvoir toutes les différencier.

3.2.2 Méthode récursive

La *méthode récursive* [NW78, FZC94] est une méthode de génération aléatoire automatique d'objets combinatoires. À partir de la spécification d'une classe combinatoire, on peut automatiquement obtenir un générateur aléatoire pour cette classe en composant des générateurs élémentaires.

Pour générer un objet de $\mathcal{A}+\mathcal{B}$ de taille n, on génère un objet de $\mathcal{A}$ de taille n avec probabilité $\dfrac{a_n}{a_n+b_n}$, de $\mathcal{B}$ sinon.

Pour générer un objet de $\mathcal{A}\times\mathcal{B}$ de taille n, on génère avec probabilité

$$\frac{a_k b_{n-k}}{\sum_{i=0}^{n} a_i b_{n-i}}$$

un objet de taille k dans $\mathcal{A}$ et un objet de taille $n-k$ dans $\mathcal{B}$.

Pour pouvoir utiliser la méthode récursive, il faut donc connaître explicitement les coefficients ce qui entraîne en général un pré-calcul coûteux (en temps mais aussi en espace). La génération s'effectue ensuite en temps quadratique pour les processus branchants pour lesquels on ne peut pas appliquer de stratégie Boustrophédon efficace (*i.e.*, choisir intelligemment l'ordre d'évaluation des $\frac{a_k b_{n-k}}{\sum_{i=0}^{n} a_i b_{n-i}}$).

Exemple. Nous donnons ici deux générateurs d'arbres de Catalan par méthode récursive. L'algorithme 2 a une complexité en temps de génération quadratique en moyenne, alors que l'algorithme Boustrophédon (algorithme 3) a une complexité en temps de génération linéaire en moyenne. On considère que les opérations arithmétiques et le tirage d'un réel uniforme sur $[0,1]$ se font en temps constant.
Les preuves de correction et de complexité de ces deux algorithmes se trouvent dans [FZC94].

Algorithm 2: Binaire1(n)

Input: n le nombre total de nœuds de l'arbre
Output: un arbre binaire uniforme de taille n.

```
1 s := 0
2 for i = 0 à n − 1 do
3 |   r := un réel uniforme sur [0, 1]
4 |   s := s + (c_i c_{n−i−1}) / c_n
5 |   if s > r then
6 |   |   return (Z, Binaire1(i), Binaire1(n − i − 1))
```

Algorithm 3: Binaire2(n)

Input: n le nombre total de nœuds de l'arbre
Output: un arbre binaire uniforme de taille n.

```
1 s := 0
2 for i = 0 à (n-1)/2 do
3     r := un réel uniforme sur [0, 1]
4     s := s + c_i c_{n-i-1} / c_n
5     if s > r then
6         return (Z, Binaire2(i), Binaire2(n - i - 1))
7     s := s + c_{n-i-1} c_i / c_n
8     if s > r then
9         return (Z, Binaire2(n - i - 1), Binaire2(i))
```

3.3 Générateur de Boltzmann

Les générateurs de Boltzmann ont été introduits par Duchon Flajolet Louchard Schaeffer [DFLS04], puis étendus aux premiers opérateurs de Pólya dans [FFP07]. Une grande littérature existe maintenant sur le sujet, à la fois pour l'extension des générateurs [Piv08, Bod11, BJ08, BRS12, BP10, BFKV07, BGR10, BL12], que pour leurs applications à la génération de structures données [BDN08, CD09, BMW13, BBM11, BN07, BFP10, Fus05]. Ces générateurs fonctionnent, dans la plupart des cas, en temps linéaire en taille approchée.

3.3.1 Générateurs d'objets non-étiquetés

Soit $\mathcal{A}$ une classe combinatoire non-étiquetée. Un *générateur de Boltzmann* de $\mathcal{A}$ de paramètre x, noté $\Gamma_x\mathcal{A}$, est un algorithme qui tire aléatoirement un élément a de $\mathcal{A}$ avec probabilité :

$$\mathbb{P}(a) = \frac{x^{|a|}}{A(x)}$$

où $A(x)$ est l'évaluation de la série génératrice ordinaire de $\mathcal{A}$ en x, avec x réel dans son rayon de convergence.

On remarque que tous les objets ont une probabilité positive d'être tirés, mais que deux objets de même taille sont générés avec la même probabilité, ce qui est à rapprocher d'une notion d'uniformité. On peut choisir judicieusement le paramètre pour viser une taille donnée, et utiliser du rejet pour obtenir des générateurs en taille exacte ou approchée. Plus de détails seront donnés sur le sujet en sous-section 3.3.4.

3.3.2 Générateur d'objets étiquetés

Dans le cas d'une classe combinatoire étiquetée, Un *générateur de Boltzmann* de $\mathcal{A}$ de paramètre x, noté $\Gamma_x\mathcal{A}$, est un algorithme qui tire aléatoire un élément a de $\mathcal{A}$ avec probabilité :

$$\mathbb{P}(a) = \frac{x^{|a|}}{|a|!A(x)}$$

La méthode pour générer un objet étiqueté est en général [2] de tirer aléatoirement un squelette d'objet étiqueté puis de tirer une permutation aléatoire et d'associer à chaque atome un nombre (celui correspondant à sa position dans la permutation, si on regarde une représentation linéaire de la structure). Notons que cette méthode est correcte même quand on utilise des opérateurs de Pólya : on génère dans ce cas un représentant linéaire (car il doit de toute façon être représenté en machine) dont on utilise l'ordre pour la correspondance entre atome et étiquette.

3.3.3 Générateurs des constructeurs

Les générateurs de Boltzmann se basent sur les spécifications et, comme les séries génératrices, peuvent être construits automatiquement à partir de celles-ci dans les cas favorables. Nous décrivons ici les générateurs des différents constructeurs utilisés dans la suite de ce manuscrit, à composer pour obtenir un générateur de Boltzmann d'une classe donnée.

Les générateurs d'unions disjointes, de produits cartésiens et de séquences sont les mêmes dans les cas étiqueté et non étiqueté. Notons que pour les classes étiquetées il faudra, après génération complète de l'objet sans ses étiquettes, étiqueter uniformément les atomes en tirant une permutation.

Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes combinatoires. Le générateur de Boltzmann de $\mathcal{A} \times \mathcal{B}$ (*resp.* $\mathcal{A} + \mathcal{B}$, *resp.* $\textsc{Seq}\,\mathcal{A}$) de paramètre x est noté $\Gamma_x\mathcal{A} \times \mathcal{B}$ (*resp.* $\Gamma\mathcal{A} + \mathcal{B}$, *resp.* $\Gamma_x\,\textsc{Seq}\,\mathcal{A}$) et est décrit dans l'algorithme 4 (*resp.* l'algorithme 5, *resp.* l'algorithme 6). Dans ces algorithmes, $A(x)$ (*resp.* $B(x)$) est la fonction génératrice ordinaire de $\mathcal{A}$ (*resp.* de $\mathcal{B}$) dans le cas d'une spécification non-étiquetée et exponentielle de le cas d'une spécification étiquetée.

Algorithm 4: $\Gamma_x\mathcal{A} \times \mathcal{B}$

Input: x le paramètre
Output: un objet de $\mathcal{A} \times \mathcal{B}$ avec probabilité de Boltzmann
1 **return** $(\Gamma_x\mathcal{A}, \Gamma_x\mathcal{B})$

2. à part pour les structures ordonnées comme les arbres croissants [RS09]

Algorithm 5: $\Gamma_x \mathcal{A} + \mathcal{B}$

Input: x le paramètre
Output: un objet de $\mathcal{A} + \mathcal{B}$ avec probabilité de Boltzmann

```
r := 0 ou 1 tiré selon une loi de Bernoulli de paramètre A(x)/(A(x)+B(x))
if r = 0 then
    return Γ_x A
else
    return Γ_x B
```

Les générateurs de séquences, de cycles et d'ensembles étiquetés fonctionnent tous les trois de la même manière : on tire le nombre d'éléments de la séquence (du cycle, de l'ensemble) puis on tire chacun de ces éléments indépendamment selon un générateur de Boltzmann de même paramètre.

Algorithm 6: $\Gamma_x \textsc{Seq}\, \mathcal{A}$

Input: x le paramètre
Output: un objet de $\textsc{Seq}\, \mathcal{A}$ avec probabilité de Boltzmann

```
r := un entier tiré selon une loi géométrique de paramètre A(x)
s := ϵ une chaîne vide
for i = 1 à r do
    s := s.Γ_x A, la concaténation d'un objet tiré par Γ_x A à s
return (s)
```

Lorsque les opérateurs ne conservent pas la rigidité, les générateurs de Boltzmann seront différents dans le cas étiqueté du cas non-étiqueté. Nous donnons ici les générateurs $\Gamma_x \textsc{Cyc}\, \mathcal{A}$ et $\Gamma_x \textsc{Set}\, \mathcal{A}$ des cycles et ensembles étiquetés.

Algorithm 7: $\Gamma_x \textsc{Cyc}\, \mathcal{A}$

Input: x le paramètre
Output: un objet de $\textsc{Cyc}\, \mathcal{A}$ avec probabilité de Boltzmann

```
r := un entier tiré selon une loi logarithmique de paramètre A(x)
s := ϵ une chaîne vide
for i = 1 à r do
    s := s.Γ_x A, la concaténation d'un objet tiré par Γ_x A à s
return (s)
```

Algorithm 8: $\Gamma_x \textsc{Set}\,\mathcal{A}$

Input: x le paramètre
Output: un objet de $\textsc{Set}\,\mathcal{A}$ avec probabilité de Boltzmann

```
r := un entier tiré selon une loi de Poisson de paramètre A(x)
s := ϵ une chaîne vide
for i = 1 à r do
    s := s.Γ_x A, la concaténation d'un objet tiré par Γ_x A à s
return (s)
```

Nous aurons également besoin, dans le chapitre 11, du générateur de Boltzmann de multi-ensembles non étiquetés. L'idée est de tirer des éléments du multi-ensemble avec des multiplicités de tirage, indépendamment les uns des autres. Un même élément peut ainsi être tiré plusieurs fois (avec différentes multiplicités ou pas) lors de la génération : sa multiplicité dans le multi-ensemble sera alors la somme des multiplicités avec lesquelles il a été tiré. On détermine pour cela une multiplicité maximum de tirage, que nous utilisons comme limite de boucle dans l'algorithme 9. Nous présentons ici cet algorithme sans plus d'explications, pour plus de détail voir [FFP07], [BFP10] ou [Piv08].

On note $M_{\mathcal{A}}(x)$ l'évaluation de la série génératrice ordinaire de $\textsc{MSet}\,\mathcal{A}$ en x. Soit $\text{IndiceMax}\,(A, x)$ un algorithme tirant un entier positif selon la distribution :

$$\mathbb{P}(K \leq k) = \prod_{j>k} \exp\left(\frac{1}{j} A(x^j)\right).$$

Ceci peut être réalisé par méthode inverse [Dev86, Section 2.1][Knu97b, Section 4.1].

Algorithm 9: Γ_x MSET $\mathcal{A}$

Input: x le paramètre
Output: un objet de MSET $\mathcal{A}$

$i_{max} :=$ IndiceMax (A, x)
$s := \epsilon$ une chaîne vide
if $i_{max} \neq 0$ **then**
  for $j = 1$ *à* $i_{max} - 1$ **do**
    $p :=$ un entier tiré selon une loi de Poisson de paramètre $\frac{A(x^j)}{j}$
    for $i = 1$ *à* p **do**
      $a := \Gamma_{x^j}\mathcal{A}$
      $s := s.a.a...a$, la concaténation de j copies de a à s
  $p :=$ un entier tiré selon une loi de Poisson positive de paramètre $\frac{A(x^{i_{max}})}{i_{max}}$
  for $i = 1$ *à* p **do**
    $a := \Gamma_{x^{i_{max}}}\mathcal{A}$
    $s := s.a.a..a$ la concaténation de i_{max} copies de a à s
return s

Remarquons qu'ici le paramètre de Boltzmann change au cours de la génération : il est donc nécessaire de pouvoir évaluer rapidement la série génératrice en plusieurs points.

De nombreux travaux ont étendu les générateurs de Boltzmann à d'autres opérateurs ou classes : produit de Hadamard [BGR10], produit de mélange (*shuffle*) [DPRS10b, DPRS10a], intégrales (opérateur *box*) [BRS12], quotient [BBM11], classes colorées [BJ08], cycles équilibrés [BJ13]...

On remarquera que si l'on peut construire un générateur de Boltzmann *ad-hoc*, *i.e.*, un algorithme tirant un objet d'une classe avec probabilité de Boltzmann pour une classe non-constructive (par exemple), on peut ensuite utiliser ce générateurs dans les algorithmes ci-dessus. Cette remarque nous sera utile dans le chapitre 11. A l'inverse, dans le chapitre 8 on utilise un générateur de Boltzmann à l'intérieur d'un générateur *ad-hoc* afin d'obtenir une distribution uniforme à taille fixée mais ne suivant pas une probabilité de Boltzmann.

3.3.4 Choix du paramètre et complexité

Comme nous l'avons vu plus haut, un générateur de Boltzmann génère tous les objets d'une classe avec une probabilité positive. Cependant, la probabilité de tirer un objet de taille n dépend grandement du paramètre x. En particulier, l'espérance de la taille de l'objet généré par un appel à un générateurs de Boltzmann de paramètre x de la classe $\mathcal{A}$ est :

$$\mathbb{E}_x(N) = x\frac{A'(x)}{A(x)},$$

et sa variance :

$$\mathbb{E}_x(N^2) = \frac{x^2 A''(x) x A'(x)}{A(x)}.$$

En fait, on peut calculer tous les moments de manière similaire.

L'espérance de la taille est croissante en fonction du paramètre. On peut fixer ce paramètre pour viser des objets d'une taille donnée. On peut ensuite rejeter les objets tirés jusqu'à obtenir un objet de la taille souhaitée, c'est ce que l'on appellera un *générateur de Boltzmann en taille exacte*. Une autre possibilité, en fonction des applications visées, est d'accepter une tolérance relative sur la taille visée. Dans ce cas, on fixe la tolérance $\varepsilon > 0$ et on rejette et recommence l'algorithme jusqu'à obtenir un objet dont la taille est comprise entre $(1-\varepsilon)n$ et $(1+\varepsilon)n$, où n est la taille visée. On parle dans ce cas de *générateurs de Boltzmann en taille approchée*. Ces rejets conservent la propriété que deux objets de même taille ont la même probabilité d'être tirés.

Intéressons nous maintenant à la complexité des générateurs. On se place pour cela dans un modèle où l'on peut tirer une loi uniforme dans $[0, 1]$ et faire des calculs arithmétiques réels en temps $O(1)$. La complexité en espace mémoire est linéaire en la taille de la sortie (on ne construit rien d'autre). La complexité en nombre de bits aléatoires n'a pas initialement été étudiée, mais a suscité de l'intérêt depuis [Duc11].

La complexité en temps de la génération sera en général linéaire pour les générateurs de Boltzmann en taille approchée de classes décomposables, linéaire à quadratique en taille exacte. Pour analyser ces générateurs on distingue trois types de distribution pour la taille de l'objet tiré, de la plus favorable à la moins favorable :

- centrée (bumpy)
- plate (flat)
- pointue (peak).

Nous ne présentons ici que la distribution centrée, pour les distributions plate et pointue (qui joueront un rôle moins important dans cette thèse) le lecteur est invité à se référer à [DFLS04].

Définition 3.1. La distribution de la sortie d'un générateur de Boltzmann d'une classe $\mathcal{A}$ est dite *centrée* quand les conditions suivantes sont satisfaites :

- $\mu_1(x) \to \infty$ quand $x \to \rho^-$
- $\sigma(x)/\mu_1(x) \to 0$ quand $x \to \rho^-$,

où $\mu_1(x)$ (resp. $\sigma^2(x)$) est l'espérance (resp. la variance) de taille de la sortie, et ρ est la singularité dominante de $A(x)$.

Lorsque la distribution est centrée, on rejette en moyenne $O(1)$ fois pour le générateur en taille exacte, menant à une complexité linéaire.

Pour compléter la description des générateurs de Boltzmann, il faut remarquer que l'on a besoin, pour qu'ils soient implémentables, de pouvoir calculer l'évaluation de la série génératrice en un point (pour les générateurs sans symétries) ou en plusieurs points. Lorsque l'on a pas de formule close simple, il s'agit d'un problème complexe qui a été traité dans [PSS08, PSS12] pour les spécifications usuelles.

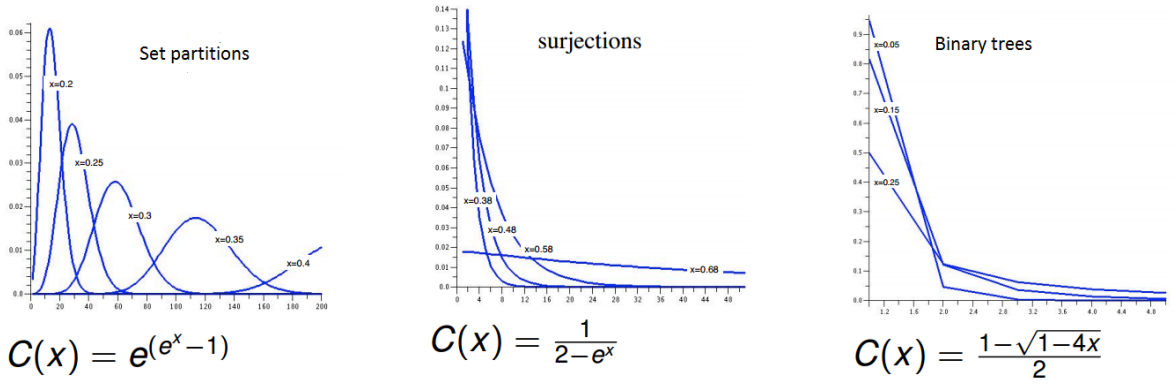


FIGURE 3.1 – De gauche à droite : la distribution centrée des partitions d'ensembles, le distribution plate des surjection et la distribution pointue des arbres de Catalan pour différentes valeurs du paramètre. Cette figure est due à Carine Pivoteau.

3.4 Applications de la génération aléatoire

Comme évoqué plus haut, les applications de la génération aléatoire sont très variées. La plus évidente, mais souvent plus utile qu'il n'y paraît, est de pouvoir observer un ensemble d'objets représentatifs d'une classe combinatoire. Cela permet une première étude empirique des propriétés moyennes des objets de la classe. Cette phase d'observation peut mener à des conjectures. Par exemple, dans le chapitre 11 la forme limite que nous donnons pour les chemins NW-convexes a d'abord été observée sur les objets produits aléatoirement, puis prouvée. De manière plus générale, cette observation peut donner sur les propriétés moyennes à étudier, sur la validité des conjectures énoncées ou sur la pertinence du modèle de distribution choisi.

D'autres applications plus concrètes existent également, nous présentons ici le test logiciel à grande échelle [MDBS09, CD09, Dar08], la simulation de contexte [CDJ] et le raffinement de modèle [DPT10].

Dans le cas du test logiciel, on se place dans le cadre d'algorithmes prenant des structures de données, potentiellement grandes, en entrée. Pour attester de la robustesse on peut vouloir tester la charge : envoyer de très grosses données, ou beaucoup de données ou les deux à la fois. Sans génération aléatoire, cela nécessite de conserver de grandes bases de données de structures, ce qui est coûteux en espace. Un générateur aléatoire pourra à la place fournir (rapidement) une grande structure aléatoire avec très peu de stockage préalable.

Dans la même veine, la *simulation de contexte* a pour but de permettre des benchmarks sur des ensemble de données plus grands que ceux initialement disponibles. En partant d'un ensemble d'objets, obéissant à une distribution de probabilité inconnu, on fait un certain nombre de mesures statistiques de grandeurs prédéterminées (par exemple, le nombre d'occurrences de certaines configurations, la taille moyenne, la variance de certains paramètres,...) en espérant ainsi caractériser au mieux cette distribution. Puis, on règle un générateur aléatoire pour qu'il génère des objets ayant une distribution proche sur ces mesures. On peut ainsi obtenir efficacement de grands

ensembles de grands objets, de distribution que l'on espère proche de celle de la réalité.

L'esprit du *raffinement de modèle* est un peu différent. On a un système réel que l'on souhaite modéliser, comme par exemple le repliement des molécules. On connait un certain nombre de règles physiques qui les régissent, mais est-ce suffisant pour expliquer leur distribution ? Pour le tester, on construit un générateur aléatoire qui génère selon ces règles et on observe les résultats obtenus. Sont-ils distribués comme dans la réalité ? En général, il ne le seront pas parfaitement. On peut alors étudier les différences entre l'échantillon réel et l'échantillon simulé. Ces différences nous indiquent alors des pistes de corrections. On raffine alors le modèle (ou les règles physiques que l'on cherche à déterminer) et on recommence.

Deuxième partie

Algorithmes de Rémy

Chapitre 4

L'algorithme original de Rémy

4.1 Introduction

Les arbres enracinés planaires, présentés dans le chapitre 2, sont des structures de données centrales en informatique, et ont été très largement étudiées à la fois en mathématiques et et informatique. Dans cette partie, nous nous intéressons à la génération aléatoire d'arbres de Catalan et d'arbres de Motzkin. La génération aléatoire efficace (en temps, en espace et en bits aléatoires) de ces arbres est d'un grand intérêt. En effet ces structures apparaissent dans de nombreux domaines (structures de données, bioinformatique, probabilités,...) et un générateur uniforme efficace permet de réaliser des tests, des conjectures, et des simulations pour des propriétés importantes.

La génération aléatoire d'arbres a une longue histoire. Les méthodes classiques de génération incluent la méthode récursive [NW78, FZC94], la méthode de Boltzmann [DFLS04], des bijections avec des mots [ARS97, Alo94] ou avec des marches [BPS92], et, pour les arbres de Catalan, des processus de Galton-Watson finis et l'algorithme de Rémy [Rem85].

L'algorithme de Rémy est l'algorithme le plus efficace jusqu'à maintenant. Il consiste à faire grossir un arbre depuis une feuille par des transformations locales que nous appellerons greffes. Le principal intérêt de cette méthode pour la génération aléatoire est d'éviter un processus de branchements entre sous-structures de tailles quelconques. C'est en effet sur ces produits que les méthodes classiques de générations aléatoires perdent en efficacité, ou en précision sur la taille de la sortie pour les générateurs de Boltzmann.

Notre première contribution est de réinterpréter la bijection de Rémy en utilisant des spécifications faisant intervenir des opérateurs différentiels, puis de l'améliorer et de l'étendre aux arbres de Motzkin.

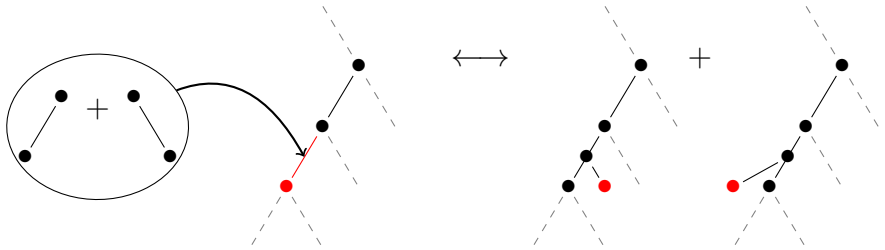


FIGURE 4.1 – La greffe d'un nœud binaire et d'une feuille à un arbre binaire dont un nœud est distingué. Les nœuds distingués apparaissent en rouge.

4.2 L'algorithme de Rémy

L'algorithme de Rémy [Rem85, M̈99] est un algorithme *ad hoc* de génération aléatoire uniforme d'arbres binaires. Il se base sur l'équation récursive :

$$(n+2)c_{2n+3} = 2(2n+1)c_{2n+1}$$

où c_{2n+1} est le nombre d'arbres de Catalan à $2n+1$ nœuds (interne ou feuille),*i.e.*, c_{2n+1} est le n-*ième* nombre de Catalan. [1]

Cette équation récursive est alors interprétée comme une bijection entre le produit cartésien de $\{0,1\}$ et l'ensemble des arbres de Catalan de taille $2n+1$ dont un nœud quelconque (interne ou feuille) est distingué, noté $\mathcal{C}^{\bullet}_{2n+1}$ et l'ensemble des arbres de Catalan de taille $2(n+1)+1$ dont une feuille est distinguée, noté $\mathcal{C}^{(f)}_{2n+1}$.

Remarquons que cette équation récursive a la propriété intéressante d'être une récursion simple, *i.e.*, on a uniquement besoin du terme $n-1$ pour obtenir le terme en n. Cette propriété est très intéressante d'un point de vue énumératif. En ce sens, la bijection de Rémy a récemment été étendue à des classes de cartes combinatoires [Bet14] par une récurrence sur plusieurs variables.

Décrivons maintenant cette bijection. Soient $(e,T) \in \{0,1\} \times \mathcal{C}^{\bullet}_{2n+1}$, d le nœud distingué de T, p son père (si d est racine, $p = \perp$). On ajoute un nœud binaire b et une feuille f en modifiant l'arbre tel que : p est le père de b, si $e = 0$ (*resp.* 1), d est le fils gauche (*resp.* droit), de b et f est le fils droit (*resp.* gauche) de b. On marque f comme distinguée et on oublie la distinction de d. Le reste de l'arbre reste inchangé. On obtient ainsi un arbre de $\mathcal{C}^{(f)}_{2n+1}$. Cette bijection est illustrée dans la figure 4.1. On dit que cette bijection est une *greffe* car elle consiste à l'insertion à un endroit quelconque d'un nombre fini de nœuds.

Il s'agit bien d'une bijection : on peut facilement appliquer la bijection inverse - qui consiste à marquer le nœud frère de la feuille marquée puis à supprimer la feuille et son père en « recollant » les deux arêtes dont il manque une extrémité ainsi obtenues, et à garder un bit d'information 0 ou 1 en fonction de si le feuille marquée était fille droite ou gauche - pour retrouver l'arbre de départ. De même, on peut appliquer cette

1. Ce choix de taille est fait pour garder une cohérence avec ce qui va suivre. Comme on introduit des opérateurs de pointage permettant de choisir n'importe quel nœud d'un arbre, cette définition de la taille est pour nous plus naturelle.

bijection inverse puis la greffe à un arbre de $\mathcal{C}_{2n+1}^{(f)}$ et le retrouver. Une preuve plus détaillée sera donnée en section 5.1.1.

Nous pouvons maintenant donner l'algorithme de Rémy. Dans cet algorithme, on part d'un arbre réduit à une feuille sur lequel on fait des greffes successives jusqu'à obtenir un arbre de la taille souhaitée.

Algorithm 10: Remy(n)

Input: n le nombre de nœuds internes de l'arbre à générer
Output: un arbre choisi uniformément parmi ceux de taille $2n+1$

```
1 T := un arbre réduit à une feuille
2 for i = 1 à n do
3     d := un nœud choisi uniformément parmi les nœuds de T
4     e := 0 ou 1 uniformément
5     if e = 0 then
6         Greffer un nœud interne et une feuille à droite de d
7     else
8         Greffer un nœud interne et une feuille à gauche de d
9 return Γ_x B
```

Cet algorithme génère bien un arbre de taille $2n+1$ uniformément : en effet, on est uniforme à chaque étape puisque tout arbre de taille $2i+1$ a $i+1$ feuilles, on peut donc oublier la feuille distinguée sans biaiser. Une autre façon de la voir est de marquer la feuille distinguée par i lorsqu'elle apparaît à l'étape i. On obtient ainsi un arbre dont les feuilles sont étiquetées, donc chaque forme d'arbre de taille $2n+1$ apparaît $(n+1)!$ fois. C'est sous cette forme que l'algorithme est prouvé dans [Rem85].

Pour la complexité en temps de cet algorithme, nous allons nous placer dans le modèle suivant, utilisé tout du long de cette partie. On considère que les opérations suivantes peuvent être effectuées en temps $O(1)$:

- lire et écrire l'identifiant d'un nœud v ;
- déterminer si un nœud v est un fils gauche ou un fils droit et trouver son père ;
- déterminer si un nœud v est un nœud binaire ou une feuille et trouver ses fils ;

Une structure de données vérifiant ces conditions sera donnée à la section 6.2.1.
De plus, on suppose ici que tirer aléatoirement uniformément un entier entre 1 et m peut être fait un temps constant pour tout m (cette condition sera relâchée dans les chapitres suivants).

Une fois ceci établi, on a la propriété suivante :

Propriété 4.1. *L'algorithme de Rémy a une complexité linéaire en temps et en espace, et utilise $\Theta(n \log n)$ bits aléatoires.*

Démonstration. Les complexités en temps et espace sont évidentes.

Pour les bits aléatoires, pour chaque i entier entre 1 et n, si on veut générer un arbre de taille $2n+1$, on tire un entier entre 1 et $2i+1$ afin de pouvoir choisir uniformément

un nœud de l'arbre. L'entropie d'un tel tirage est $\log_2(2i+1)$. Or,

$$\sum_{i=1}^{n} \log_2(2i+1) = \Theta(n \log n).$$

À chaque étape on choisit également uniformément droite ou gauche, ce qui coute 1 bit aléatoire. On utilise donc bien $\Theta(n \log n)$ bits aléatoires en tout. □

4.3 Spécification holonome

Nous allons maintenant retranscrire la bijection précédente en terme de spécification combinatoire.

Soit $\mathcal{C}$ la classe des arbres de Catalan munie de la fonction taille nombre total de nœuds. Soit $\mathcal{C}^\bullet$ (*resp.* $\mathcal{C}^{(f)}$) la classe des arbres de Catalan munie de la fonction taille nombre total de nœuds dont un nœud quelconque (*resp.* une feuille) est distingué. Le bijection de Rémy donne directement la spécification :

$$\mathcal{C}^{(f)} = \mathcal{Z} + 2\mathcal{Z}^2\mathcal{C}^\bullet.$$

En effet, le plus petit arbre ayant une feuille distinguée est l'arbre réduit à un feuille distinguée, et tous les autres peuvent être obtenus, via la bijection de Rémy, en ajoutant un nœud interne et une feuille à un arbre plus petit. On a ici une justification du choix de notre fonction taille : en choisissant le nombre total de nœuds, on peut ainsi distinguer n'importe quel nœud de l'arbre par un opérateur point.

Ceci nous permet en effet de rester en une seule variable lorsque l'on établit :

$$2\mathcal{C}^{(f)} \sim \mathcal{C} + \mathcal{C}^\bullet.$$

Ceci découle directement du fait qu'un arbre (sans nœud distingué) de taille $2n+1$ a $n+1$ feuilles, donc apparaît $n+1$ fois dans $\mathcal{C}^{(f)}$, donc apparaît $2n+2$ fois dans $2\mathcal{C}^{(f)}$ (la classe des arbres ayant une feuille distinguée, où chaque arbre avec une feuille distinguée apparaît avec multiplicité 2). Or, $2n+2$ est également le nombre total de nœud plus un, ce qui correspond à compter chaque arbre une fois pour chaque nœud qu'il contient, *i.e.*, à compter chaque arbre une fois pour chacun de ses nœuds, plus une fois, ce qui correspond à un arbre sans nœud distingué. Une bijection explicite peut être obtenue en donnant une correspondance explicite entre feuilles et nœuds binaires, par exemple en prenant l'ordre du parcours en profondeur d'abord. Une bijection algorithmiquement efficace sera donnée dans le chapitre suivant.

On obtient donc la spécification :

$$\mathcal{C} + \mathcal{C}^\bullet = 2\mathcal{Z} + 4\mathcal{Z}^2\mathcal{C}^\bullet$$

pour les arbres de Catalan. Cette spécification a la particularité d'être *holonome* : l'équation correspondante est une équation différentielle linéaire à coefficients polynomiaux.

Remarque. En une variable, les termes « *holonome* » et « *D-fini* » sont synonymes.

Les équations holonomes jouent un rôle important en calcul formel : elles permettent, entre autres, de calculer rapidement les premiers termes de la série génératrice de la classe (car elles correspondent à des équations récursives de profondeurs bornées), et il existe de nombreux outils analytiques, souvent moins puissants que ceux pour les classes algébriques. Toute classe algébrique (*i.e.*, pour laquelle il existe une spécification algébrique) est également holonome.

Mais l'aspect qui nous intéresse ici est surtout celui de la génération aléatoire. En effet, pour la méthode récursive ce sont les produits entre classes infinies qui font perdre en efficacité. Dans le cas des générateurs de Boltzmann, ce sont ces mêmes produits qui donnent une taille de sortie difficilement maîtrisable. Nous allons contourner ces difficultés en utilisant une équation holonome. Attention cependant, si ces méthodes sont appliquées telles quelles sur des spécifications holonomes on n'obtient pas des générateurs efficaces : dans le cas de la méthode récursive il faut faire beaucoup d'arithmétique pour recalculer les coefficients des séries pointées et dans le cas de la méthode de Boltzmann il faut utiliser un opérateur de dépointage [BRS12, Rou12] demandant d'évaluer les séries en un grand nombre de points. Nous avons étudié une adaptation d'un générateur de Boltzmann [BBJ13], difficilement implémentable en pratique.

Le chapitre 5 présente une amélioration de l'algorithme de Rémy permettant d'utiliser un nombre de bits aléatoires quasi-optimal, c'est-à-dire, pour générer un arbre de taille n uniformément on utilise $\log_2 C_n + o(\log_2 C_n)$ bits aléatoires tout en conservant une complexité en temps linéaire en moyenne. Cette amélioration est basée sur la spécification donnée ci-dessus, accompagnée de la correspondance efficace entre feuilles et nœuds binaires.

Le chapitre 6 étend ensuite une version de cet algorithme aux arbres de Motzkin. Cette extension repose sur une spécification holonome des arbres de Motzkin, dont on décrit la bijection dans ce chapitre. Ici, le fait de passer par des spécifications permet, via le calcul formel [SZ94], d'obtenir automatiquement une équation sur les séries. On doit ensuite réinterpréter bijectivement cette équation, et l'utiliser pour obtenir un algorithme de génération aléatoire efficace. Cet algorithme utilise toujours un temps linéaire et un nombre de bits aléatoires proportionnel à l'entropie, mais cette fois-ci la constante multiplicative est supérieure à 1. Autrement dit, l'algorithme utilise $O(\log_2 C_n)$ bits aléatoires pour générer uniformément un arbre de taille n.

Chapitre 5

Amélioration de l'algorithme de Rémy

Nous allons dans un premier temps reprendre la spécification des arbres de Catalan déjà brièvement présentée dans le chapitre précédent. Nous l'utiliserons ensuite pour présenter deux algorithmes : un premier qui découle naturellement de la spécification, et un deuxième très proche mais plus efficace. Ce dernier algorithme utilise $2n + \Theta(\log(n)^2)$ bits aléatoires et en temps linéaire en moyenne.

5.1 Spécification des arbres de Catalan

5.1.1 La greffe

Dans cette section nous allons présenter la bijection F pour les arbres binaires :

$$F : 2\mathcal{Z} + 4\mathcal{Z}^2\mathcal{B}^\bullet \to \mathcal{B}^\star$$

où $\mathcal{B}$ est la classe des arbres de Catalan munie de la fonction taille nombre total de nœuds et $\mathcal{B}^\star$ est la classe des arbres de Catalan dont une feuille est distinguée et colorée en rouge ou en bleu, munie de la même fonction taille. Autrement dit,

$$\mathcal{B}^\star = 2\mathcal{B}^{(f)}$$

où $\mathcal{B}^{(f)}$ est la classe des arbres de Catalan dont une feuille est distinguée. Soit T un arbre de $\mathcal{B}^\star$. On dit qu'il est *pointé en couleur* et que sa feuille distinguée est *pointée en rouge* (ou *pointée en bleu*).

Cette bijection, qui est, à un facteur près, celle de Rémy présentée dans la section 4.2, repose sur le principe de greffe. La bijection inverse est donnée en preuve.

Le terme $2\mathcal{Z}$ correspond aux deux arbres réduits à une feuille pointée en bleu ou en rouge. Maintenant, on a 4 manières de faire grossir un arbre de $\mathcal{B}^\bullet$:
Soit T un arbre pointé de $\mathcal{B}^\bullet$. Soit n le nœud pointé, p son père (si n est la racine, $p = \perp$). On a 4 cas possibles, que l'on appelle $F1$, $F2$, $F3$ et $F4$. Dans chacun de ces cas, on crée un nouveau nœud binaire b et une nouvelle feuille ℓ tel que b devient le père de n, p de b et soit :

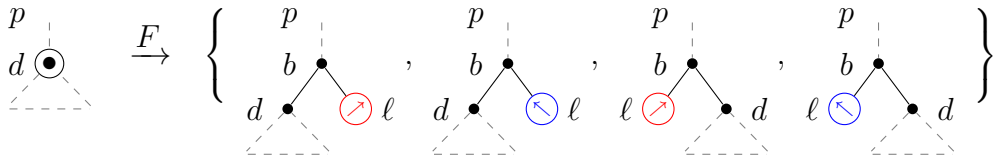


FIGURE 5.1 – La bijection de $\mathcal{B}^{\bullet}_n$ à $\mathcal{B}^{\star}_{n+2}$. Remarquons que le noeud b prend la place de d : il devient la racine si d l'était, le fils gauche (*resp.* droit) de p si d était était le fils gauche (*resp.* droit) de p.

- dans le cas $F1$, ℓ est la fille droite de b et est pointée en rouge,
- dans le cas $F2$, ℓ est la fille droite de b et est pointée en bleu,
- dans le cas $F3$, ℓ est la fille gauche de b et est pointée en rouge,
- dans le cas $F4$, ℓ est la fille gauche de b et est pointée en bleu,

le point d'origine en n est oublié et le reste de l'arbre est inchangé.

Cette bijection est représentée dans la figure 5.1, où un nœud pointé est représenté par ⊙, une feuille pointée en bleu par ↖ et une feuille pointée en rouge par ↗. Ces notations se justifieront dans la section suivante.

Démonstration. Décrivons maintenant la bijection inverse F^{-1} qui, à chaque arbre de $\mathcal{M}^{\star}$ de taille 3 ou plus, associe un unique arbre de $\mathcal{B}^{\bullet}$.[1] Soit ℓ la feuille pointée (en rouge ou en bleu), b son père (comme l'arbre est de taille 3 ou plus, une feuille ne peut pas être la racine), s l'autre fils de b (s peut être un nœud binaire ou une feuille) et p le père de b (avec $p = \perp$ si b est la racine). La bijection F^{-1} consiste à supprimer b, à faire de p le père de s : si b était un fils gauche (resp. droit), s devient un fils gauche (resp. droit) et à pointer le nœud s.

En composant F et F^{-1}, on a $F(F^{-1}(T)) = T$ et $F^{-1}(F(T')) = T'$. □

5.1.2 Le repointage

Nous nous intéressons maintenant à la bijection associée à la spécification :

$$\mathcal{B}^{\star} = \mathcal{B}^{\bullet} + \mathcal{B}\,;$$

Nous cherchons pour cela, à partir d'un arbre T de $\mathcal{B}^{\star}$ pointé en couleur sur une feuille, à obtenir un arbre de $\mathcal{B}$ ayant la même structure d'arbre que T mais où le point a changé (on pointe maintenant un nœud quelconque, sans couleur) ou a disparu. Pour cette raison, on appelle cette opération le *repointage* de T.

Soit v un nœud de T. Le nœud v peut être un fils gauche ou un fils droit. Par convention, on considère que la racine est un fils droit. On donne également un ordre aux ancêtres du nœud pointé v, de v à la racine en suivant l'unique chemin les reliant, incluant v lui-même.

1. En fait, pour avoir une vraie bijection, il faut également garder l'information droite ou gauche et bleu ou rouge. On étend sans difficulté l'opération décrite pour ce faire.

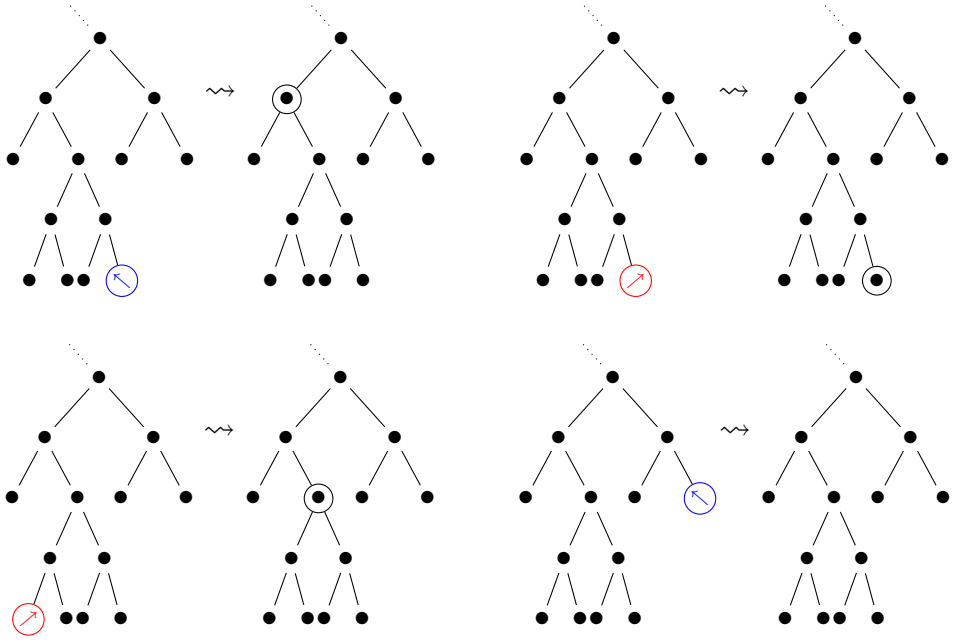


FIGURE 5.2 – Des arbres pointés en couleur de $\mathcal{C}^\star$ et leur arbres pointés associés de $\mathcal{C} + \mathcal{C}^\bullet$.

Le repointage est défini comme suit.
- Si T est pointé en bleu sur une feuille ℓ, soit v son premier ancêtre qui est un fils gauche, ou $v = \perp$ s'il n'existe pas de tel ancêtre.
- Si T est pointé en rouge sur une feuille ℓ, soit v son premier ancêtre qui est un fils droit. Comme la racine est un fils droit, un tel ancêtre existe toujours.

Dans les deux cas, le résultat est l'arbre pointé (T, v) : l'arbre ayant la même structure que T mais pointé sur v, ou l'arbre T sans point si $v = \perp$. Il s'agit donc bien d'un élément de $\mathcal{B}^\bullet + \mathcal{B}$.

Proposition 5.1. *Le repointage décrit ci-dessus est une bijection.*

Démonstration. Soit v un nœud de T ou $v = \perp$. On prouve qu'il existe une unique manière de pointer en couleur une feuille de T tel quel son repointage donne (T, v). On va pour cela distinguer trois cas.
- Si $v = \perp$, la feuille pointée en couleur correspondante ℓ doit être pointée en bleu et ne pas avoir de fils gauche parmi ses ancêtres. On en déduit que ℓ est la feuille la plus à droite de T.
- Si v est un fils gauche, la feuille pointée en couleur correspondante ℓ doit pointée en bleu et être la descendante de v telle que la branche de v à ℓ ne contient pas de fils gauche. Ce qui signifie que ℓ est la descendante la plus à droite de v.
- De manière similaire, si v est un fils droit, la feuille pointée en couleur correspondante ℓ doit être pointée en rouge et être la descendante la plus à gauche de v.

On a donc bien une bijection. □

L'avantage de cette bijection est de pouvoir être réalisée facilement algorithmiquement, comme montré dans la section 6.2.1 du chapitre suivant.

5.2 Génération aléatoire

Tous les algorithmes présentés ici suppose que les opérations suivantes sur un nœud v d'un arbre T peuvent être réalisées en temps $\mathcal{O}(1)$:

– déterminer si v est un fils gauche ou un fils droit et trouver son père ;

– déterminer si v est un nœud binaire ou une feuille et trouver ses enfants ;

De plus, on suppose que tirer aléatoire un entier dans $\{1, \ldots, m\}$ peut être fait en temps $\mathcal{O}(1)$ pour $M = 4$ ou $m = 9$. [2]

Une implémentation pratique respectant ces conditions est présentée dans la section 6.2.1 du chapitre suivant.

5.2.1 Premier algorithme

Nous présentons tout d'abord un algorithme en temps, espace et nombre de bits aléatoires utilisés linéaires. La preuve de cet algorithme sera réutilisée dans le cas des arbres de Motzkin au chapitre suivant. Nous modifierons ensuite légèrement cet algorithme afin d'atteindre une complexité en nombre de bits aléatoires en $2n + \Theta(\log^2 n)$, l'entropie étant $2n - \Theta(\log n)$.

Algorithm 11: Un essai de tirage d'un arbre de Catalan

Input: Une taille n.
Output: Un arbre binaire de taille $2n + 1$ ou **FAIL**

```
1  T := une feuille pointée en rouge ou en bleu (avec probabilité 1/2 chacun)
2  Répéter n fois :
3        (T, v) = repointer T
4        si v = ⊥ return FAIL
5        Choisir l'un des cas suivants :
6        Avec probabiblité 1/4,
7              T := F1(T, v)
8        Avec probabiblité 1/4,
9              T := F2(T, v)
10       Avec probabiblité 1/4,
11             T := F3(T, v)
12       Avec probabiblité 1/4,
13             T := F4(T, v)
14 return T
```

Remarque. À la ligne 11 de l'algorithme 11 on peut substituer « pointée en rouge ou en bleu (avec probabilité 1/2 chacun) » par « pointée en rouge ». En effet, l'algorithme

2. Dans ce chapitre, on a uniquement besoin de $m = 4$.

Algorithm 12: Arbre de Catalan aléatoire

Input: Une taille n.
Output: Un arbre binaire de taille $2n+1$

```
1 T = Un essai de tirage d'un arbre de Catalan (n)
2 while T = Un essai de tirage d'un arbre de Catalan do
3 |   T = Un essai de tirage d'un arbre de Catalan (n)
4 return T
```

échoue à moins que la feuille de laquelle on part soit rouge, donc le changement laisse inchangée la correction de l'algorithme 12.

Théorème 5.2. *L'algorithme 12 renvoie un arbre de Catalan uniformément distribué ayant $2n+1$ nœuds. Sa complexité en temps est linéaire en moyenne.*

Pour prouver le théorème, on commence par établir le lemme suivant.

Lemme 5.3. *Soit T un arbre binaire pointé en couleur sur une feuille à $2n+1$ nœuds. Une passe de l'algorithme 11 atteint T avec probabilité :*

$$\mathbb{P}(T) = \frac{1}{2 \cdot 4^n}.$$

Démonstration. Par récurrence sur n. □

Lemme 5.4. *Supposons que l'algorithme 12 tire un arbre de taille $2n+1$ avec succès. La génération a pris $\mathcal{O}(n)$ opérations.*

Démonstration. Soit T l'arbre généré. Tirer T prend n opérations de greffe et n opérations de repointage. Réaliser une greffe a une complexité $\mathcal{O}(1)$; cependant le repointage coûte $\mathcal{O}(d)$, où d est la distance parcourue par le point durant l'opération (*i.e.*, la distance entre la feuille pointée en couleur et le nouveau point). On prouve que le point parcourt en tout une distance $\mathcal{O}(n)$ lors de la génération de l'arbre.

Considérons la profondeur du point dans l'arbre en train d'être généré. Chaque greffe ajoute un à la profondeur alors que les repointages réduisent la profondeur. Comme la profondeur est positive ou nulle, le point parcourt au plus une distance n lors des repointages successif[3]. D'où le résultat. □

Preuve du théorème 5.2. Par le lemme 5.3, tous les arbres pointés en couleur à $2n+1$ sont tiré avec la même probabilité. Comme tout arbre a $2n+2$ points en couleur possibles, chaque arbre a également la même probabilité d'être tiré. Ceci prouve l'uniformité.

Pour prouver la complexité, on rappelle tout d'abord que le nombre d'arbres binaires à $2n+1$ nœuds est le nombre de Catalan C_n, qui est asymptotiquement équivalent à $4^n n^{-3/2}$. Par le lemme 5.3, comme chaque arbre a $2n+2$ points en couleur

3. Il s'agit d'une analyse de complexité amortie [Tar85, AHU74, CLRC94].

possibles, la probabilité pour une passe de l'algorithme 11 d'atteindre la taille $2n+1$ est :

$$\frac{(2n+2)C_n}{2 \cdot 4^n} \approx n^{-1/2}.$$

Ce qui signifie qu'il y a, en moyenne, $\mathcal{O}(\sqrt{n})$ passes qui échouent.

Nous cherchons maintenant l'espérance du coût d'une passe qui échoue. Pour cela, on calcule la probabilité d'échouer à la taille $2k+1$ avec $k < n$. Par la proposition 5.1, il n'existe qu'un point coloré pour chaque arbre de taille $2k+1$ qui fait échouer l'algorithme à l'étape suivante. D'après le lemme 5.3, la probabilité d'échouer à la taille $2k+1$ est alors :

$$\frac{C_k}{2 \cdot 4^k} \approx k^{-3/2}.$$

Par le lemme 5.4, l'espérance du coût d'une passe dans l'algorithme qui échoue est donc :

$$\sum_{k=0}^{n-1} \mathcal{O}(k \cdot k^{-3/2}) = \mathcal{O}(\sqrt{n}).$$

En rassemblant tous ces éléments, on montre que générer un arbre à $2n+1$ nœuds implique en moyenne $\mathcal{O}(\sqrt{n})$ passes dans l'algorithme 11, chacune coûtant $\mathcal{O}(\sqrt{n})$ en moyenne. On en déduit que le coût total est $\mathcal{O}(n)$.

□

5.2.2 Algorithme quasi-optimal

L'algorithme suivant est une version améliorée de l'algorithme 12. Son déroulement est illustré par la figure 5.3

Théorème 5.5. *L'algorithme 13 tire uniformément un arbre de taille $2n+1$ en complexité en temps et espace $O(n)$ en moyenne. Dans le pire des cas, la complexité en temps est $O(n^2 \log(n))$. De plus, soit B_n le nombre de bits aléatoires utilisés par l'algorithme ; on a, en moyenne :*

$$B_n = 2n + \frac{\log(n)^2}{4 \log 2} + \mathrm{o}(\log(n)^2).$$

Ce résultat signifie que le nombre de bits aléatoires nécessaires est très proche de l'entropie $2n$: le nombre de bits aléatoires en excès est de l'ordre de $(\log n)^2$. L'algorithme est *quasi-optimal* en nombre de bits aléatoire utilisés : ce nombre est équivalent à l'entropie, le nombre de bits aléatoires supplémentaires étant négligeable.

Démonstration. La correction de l'algorithme se démontre facilement en constatant que, à chaque étape i, l'arbre T est distribué uniformément parmi les arbres pointés en couleur de taille $2i+1$.

La complexité dans le pire des cas est également facile à montrer. Il faut tout d'abord remarquer que le pire des cas correspond à générer le peigne droit en ajoutant

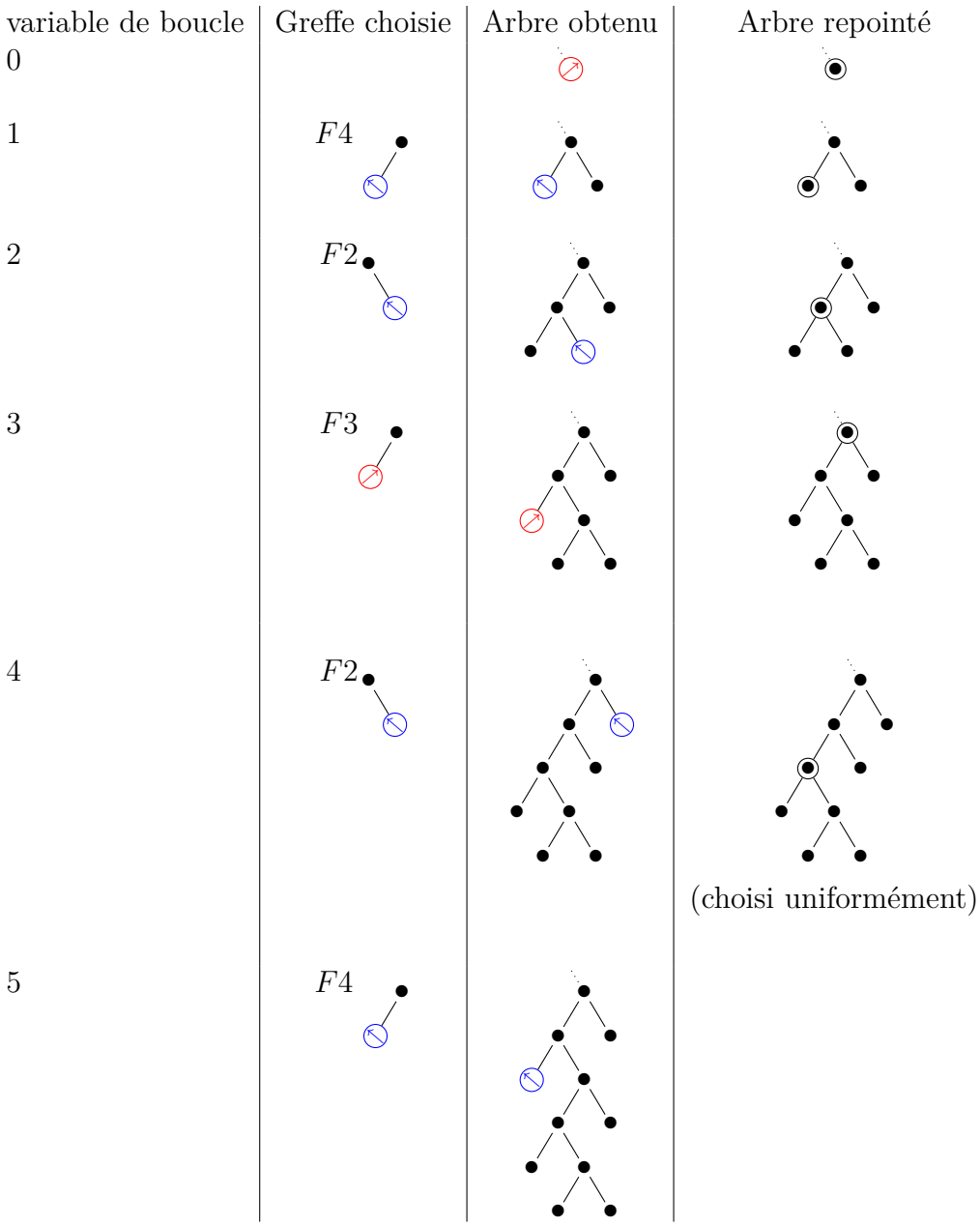


FIGURE 5.3 – Un exemple d'exécution de l'algorithme 13 générant un arbre de taille $2 \times 5 + 1 = 11$.

Algorithm 13: Générateur efficace d'arbres de Catalan

Input: Une taille n.
Output: Un arbre binaire de taille $2n+1$

```
1  T := une feuille pointée en rouge
2  Répéter n fois :
3        (T, v) = repointer T
4        si v = ⊥
5              v := un nœud de T choisi uniformément.
6        Choisir l'un des cas suivants :
7        Avec probabilité 1/4,
8              T := F1(T, v)
9        Avec probabilité 1/4,
10             T := F2(T, v)
11       Avec probabilité 1/4,
12             T := F3(T, v)
13       Avec probabilité 1/4,
14             T := F4(T, v)
15 return T
```

à chaque étape une feuille à l'extrême droite, pointé en bleu. Dans ce cas, le point parcourt $O(i)$ nœuds à l'étape i avant d'être choisi uniformément. La complexité en temps est donc $O(i \log i)$ pour l'étape i, ce qui mène à une complexité totale $O(n^2 \log(n))$.

Pour montrer la complexité en moyenne, on remarque tout d'abord que la boucle utilise 2 bits aléatoires pour choisir entre les 4 opérations de greffe possible. On a donc en tout $2n$ bits aléatoires de cette manière. Maintenant, regardons ce qu'il se passe lors de l'étape i. Comme l'arbre T est uniformément distribué et comme, pour tout arbre non-pointé, un seul point en couleur fait que v est égal à $\perp$, la probabilité de devoir repointer uniformément est $1/(2i+2)$. De plus, comme il y a $2i+1$ choix possibles pour le nouveau nœud v choisi uniformément, le coût en bits aléatoires est $\lfloor \log_2(2i+1) \rfloor$. En sommant sur toutes les étapes, le nombre moyen de bits aléatoires utilisés par les repointages uniformes est :

$$\sum_{i=0}^{n-1} \frac{\lfloor \log_2(2i+1) \rfloor}{2i+2} \sim \int_0^n \frac{\log(2t+1)}{(2t+2)\log 2} dt \sim \frac{\log^2 n}{4 \log 2}.$$

On obtient ainsi l'équivalence annoncée.

Enfin, on prouve que la complexité en temps est linéaire. Tout d'abord, on constate que tirer uniformément un nœud parmi $2i+1$ prend un temps en $O(\log_2 i)$. La seule opération qui pourrait maintenant contredire la complexité linéaire est le repointage. Comme dans la preuve du lemme 5.4, on considère la profondeur du point dans l'arbre T en train d'être généré. Cette fois-ci, la profondeur n'augmente pas seulement lors des greffes mais également lors des repointages uniformes.

À l'étape i, il y a une probabilité $1/(2i+2)$ de repointer uniformément. Après cette opération, comme l'arbre T est aléatoire uniforme, le point aura une profondeur

en $O(\sqrt{i})$ en moyenne. D'où, l'augmentation de la profondeur due aux repointages aléatoires est de l'ordre de :

$$\sum_{i=0}^{n-1} \frac{\sqrt{i}}{2i+2} \sim \mathcal{O}(\sqrt{n}).$$

Comme la profondeur du nœud pointé est positive ou nulle, le point ne peut pas parcourir une distance supérieure à $n + \mathcal{O}(\sqrt{n})$ lors de la génération de l'arbre, ce qui prouve que la génération prend un temps linéaire en moyenne. □

Pour finir, remarquons que l'on obtient par le même algorithme un générateur aléatoire uniforme quasi-optimal pour les arbres binaires pointés. En effet, à chaque étape le point est réparti uniformément sur les les sommets de l'arbre en train d'être construit. Il suffit donc d'effectuer un repointage après la dernière greffe pour obtenir un arbre pointé uniforme. Dans ce cas, le nombre de bits est encore plus proche de l'entropie (celle-ci étant $2n + \log_2 n$).

Chapitre 6

Extension aux arbres de Motzkin

Dans ce chapitre, nous présentons plus généralement les spécifications holonomes et en particulier celle pour les arbres de Motzkin. Nous en dérivons un générateur aléatoire linéaire en temps et en nombre de bits aléatoires. Les bijections et méthodes sont similaires à celles du chapitre précédent mais reposent sur une bijection plus complexe.

6.1 Spécification holonome des arbres de Motzkin

6.1.1 Équation holonome

Afin, de pouvoir utiliser un algorithme type Rémy, il nous faut tout d'abord trouver une spécification holonome pour les arbres de Motzkin, comptés selon leur nombre total de nœuds. Pour cela, nous allons commencer par trouver une équation holonome sur leur série génératrice, que nous interpréterons en temps que spécification dans le chapitre suivant.

Trouver une équation holonome à partir d'une équation algébrique peut être fait automatiquement par le calcul formel. Ici, nous avons par exemple utilisé Maple avec le package `gfun` [SZ94] en partant de l'équation

$$M(z) = z + zM(z) + zM(z)^2.$$

On obtient ainsi, après réorganisation des termes, l'équation holonome :

$$(1-z)M(z) + (z - 2z^2 - 3z^3)M'(z) = 2z.$$

On pose $M^\bullet(z) = zM'(z)$ (basé sur la définition usuelle du point) et $M^\star(z) = M(z) + M^\bullet(z)$. L'équation holonome ci-dessus se réécrit alors :

$$M^\star(z) = 2z + zM^\star(z) + zM^\bullet(z) + 3z^2M^\bullet(z). \tag{6.1}$$

D'un point de vue combinatoire, $\mathcal{M}^\bullet$ est la classe des arbres de Motzkin pointés. On note également $\mathcal{M}^{(\ell)}$ la classe des arbres de Motzkin pointés sur une feuille et par

$\mathcal{M}^{(u)}$ la classe des arbres de Motzkin pointés sur un nœud unaire. On définit $\mathcal{M}^\star$, de série génératrice $M^\star(z)$ (ce qui sera prouvé à la section 6.1.3), par :

$$\mathcal{M}^\star = \mathcal{M}^{(\ell)} + \mathcal{M}^{(\ell)} + \mathcal{M}^{(u)},$$

En d'autres mots, la classe $\mathcal{M}^\star$ est la classe des arbres de Motzkin pointés sur une feuille de deux manière différente (en rouge ou en bleu) ou pointés sur un nœud unaire (que l'on dira *pointés en vert*). Ici encore, on désigne un arbre de $\mathcal{M}^\star$ comme un arbre *pointé en couleur*.

Dans la section 6.1.2, nous donnons une interprétation combinatoire de l'équation (6.1). Dans la section 6.1.3 nous interpretons l'équation $M^\star(z) = M(z) + M^\bullet(z)$ de manière similaire à ce qui a été fait dans la section 5.1.2 du chapitre précédent. On en dérivera un générateur aléatoire « à la Rémy » dans la section 6.2.

6.1.2 La greffe pour les arbres de Motzkin

Cette section est dévouée à la bijection G sur les arbres de Motzkin :

$$G : 2\mathcal{Z} + \mathcal{Z}\mathcal{M}^\star + \mathcal{Z}\mathcal{M}^\bullet + 3\mathcal{Z}^2\mathcal{M}^\bullet \to \mathcal{M}^\star.$$

Cette bijection reprend l'idée de greffe de la bijection de Rémy, mais cette fois-ci la récurrence est sur deux niveaux (les arbres de taille n peuvent être obtenus à partir d'arbres de taille $n-1$ ou $n-2$). Dans un formalisme légèrement différent, cette bijection avait été découverte par Dulucq et Penaud [DP02]. La bijection inverse est donnée en preuve.

Pour plus de clarté, on commence par réécrire l'identité

$$2\mathcal{Z} + \mathcal{Z}\mathcal{M}^\star + \mathcal{Z}\mathcal{M}^\bullet + 3\mathcal{Z}^2\mathcal{M}^\bullet = \mathcal{M}^\star$$

dans le cas où les arbres produits sont de taille n, avec $n \geq 3$, ce qui donne :

$$G_n : \mathcal{Z}\mathcal{M}^\star_{n-1} + \mathcal{Z}\mathcal{M}^\bullet_{n-1} + 3\mathcal{Z}^2\mathcal{M}^\bullet_{n-2} \to \mathcal{M}^\star_n.$$

Maintenant, nous décomposons l'opération G_n en fonction de son ensemble de départ.

- $G1_n : \mathcal{M}^\star_{n-1} \to \mathcal{M}^\star_n$
  L'opération effectuée dépend du la couleur du point.
  - Si l'on part d'un arbre pointé en rouge sur une feuille ℓ, on appelle p le père de ℓ. L'opération $G1_n$consiste à ajouter un nouveau nœud unaire comme parent de ℓ et fils (droit, gauche ou unaire en fonction de ce qu'était ℓ) de p. On conserve le point en rouge sur ℓ.
  - Si l'on part d'un arbre pointé en rouge sur une feuille ℓ, on fait une modification similaire et on conserve le point en bleu sur ℓ.
  - Si l'on part d'un arbre pointé en vert sur un nœud unaire u, on note p son père et f son fils. L'opération $G1_n$ consiste à transformer u en noeud binaire de père p, ayant f pour fils gauche et une feuille ℓ nouvellement crée pour fils droit. Ce nouvel arbre est considéré comme pointé en rouge sur ℓ.

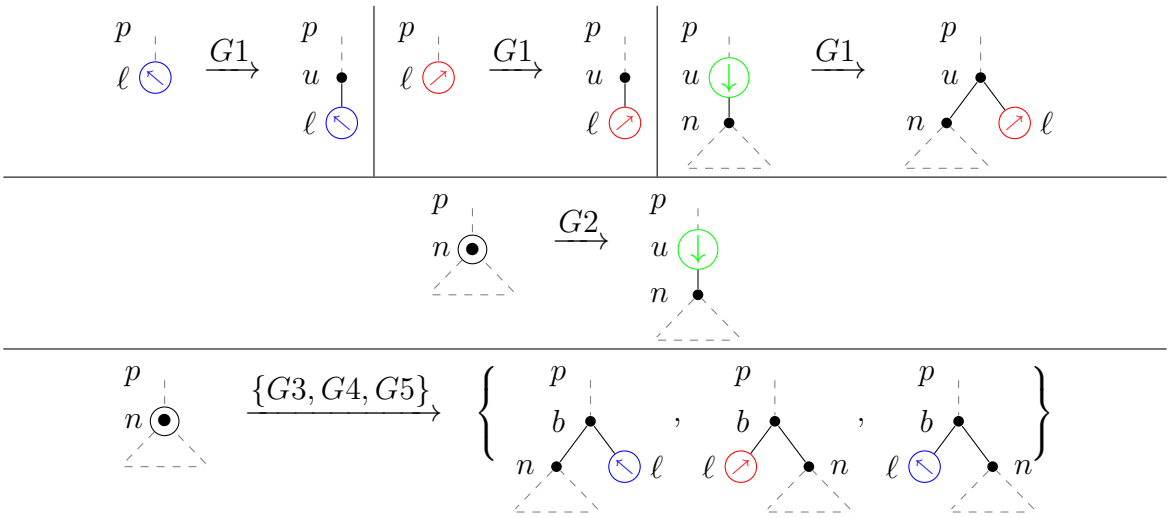


FIGURE 6.1 – La bijection G pour les arbres de Motzkin. Dans le cas $G1$ (au-dessus), le résultat dépend de la couleur du point. Remarquons qu'appliquer $G1$ ou $G2$ augmente la taille de 1 alors qu'appliquer $G3$, $G4$ ou $G5$ augmente la taille de 2.

- $G2_n : \mathcal{M}^{\bullet}_{n-1} \to \mathcal{M}^{\star}_n$
  On part d'un arbre pointé sur un nœud d, de père p, auquel on doit ajouter un nœud. Le nouvel arbre est formé un ajoutant un nœud unaire u qui devient le père de d et le fils (gauche, droit ou unaire selon ce qu'était d) de p. L'arbre est considéré comme pointé en vert sur u.
- $G3_n : \mathcal{M}^{\bullet}_{n-2} \to \mathcal{M}^{\star}_n$
  On part d'un arbre pointé sur un nœud d, de père p auquel on doit ajouter deux nœuds. On crée un nouveau nouveau binaire b et une nouvelle feuille ℓ tel que b devient le père de d, p devient le père de b et ℓ devient le fils droit de b. Le nouvel arbre est considéré pointé en bleu sur ℓ.
- $G4_n : \mathcal{M}^{\bullet}_{n-2} \to \mathcal{M}^{\star}_n$
  On part d'un arbre pointé sur un nœud d, de père p auquel on doit ajouter deux nœuds. On crée un nouveau nouveau binaire b et une nouvelle feuille ℓ tel que b devient le père de d, p devient le père de b et ℓ devient le fils gauche de b. Le nouvel arbre est considéré pointé en rouge sur ℓ.
- $G5_n : \mathcal{M}^{\bullet}_{n-2} \to \mathcal{M}^{\star}_n$
  On part d'un arbre pointé sur un nœud d, de père p auquel on doit ajouter deux nœuds. On crée un nouveau nouveau binaire b et une nouvelle feuille ℓ tel que b devient le père de d, p devient le père de b et ℓ devient le fils gauche de b. Le nouvel arbre est considéré pointé en bleu sur ℓ.

Ces opérateurs sont représentés dans la figure 6.1, où un nœud pointé est toujours représenté par ⦿, une feuille pointée en bleu par ↖, une feuille pointée en rouge par ↗ et un nœud unaire pointé en vert par ↓.

Avant de donner la bijection réciproque en preuve, on peut constater que G_n crée bien (dans l'ordre où les opérations on été présentées) :

- des arbres pointés en rouge sur une feuille fille unique
- des arbres pointés en bleu sur une feuille fille unique
- des arbres pointés en rouge sur une feuille fille droite
- des arbres pointés en vert sur un nœud unaire
- des arbres pointés en bleu sur une feuille fille droite
- des arbres pointés en rouge sur une feuille fille gauche
- des arbres pointés en bleu sur une feuille fille gauche.

Démonstration. De même que pour G, la bijection réciproque G^{-1} est décomposée en fonction du type du point de $T \in \mathcal{M}_n^\star$. Nous décrivons ici explicitement G^{-1}, mais il n'y a rien de compliqué : il suffit de défaire chaque opérations effectuée par G.
Soit $T \in \mathcal{M}_n^\star$, de taille n

- Si T est pointé en rouge sur une feuille fille unique ℓ, son image est dans $\mathcal{M}_{n-1}^\star$. Soit u le nœud unaire père de ℓ et p le père de u. On obtient l'arbre $G^{-1}(T)$ en supprimant u. La feuille ℓ devient alors la fille (gauche, droit ou unique selon ce qu'était u) de p. Le reste de l'arbre est inchangé et l'on conserve la point en rouge sur ℓ.
- Si T est pointé en rouge sur une feuille fille unique ℓ, son image est dans $\mathcal{M}_{n-1}^\star$ et l'opération est alors identique à celle du cas précédent, à la couleur du point près.
- Si T est pointé en rouge sur une feuille fille droite ℓ, son image est dans $\mathcal{M}_{n-1}^\star$. Soient b le nœud binaire père de ℓ, p le père de b et d le fils gauche de b. L'arbre $G^{-1}(T)$ est obtenu en supprimant ℓ, transformant b en un nœud unaire de père p et de fils unique d. L'arbre est maintenant considéré comme pointé en vert sur b.
- Si T est pointé en vert sur un nœud unaire u, son image est dans $\mathcal{M}_{n-1}^\bullet$. Soient p le père de u et d son fils. L'arbre $G^{-1}(T)$ est obtenu en supprimant u, et d devient alors le fils (gauche, droit ou unaire selon ce qu'était u) de p. L'arbre est considéré pointé sur d.
- Si T est pointé en rouge sur une feuille fille droite ℓ, son image est dans $\mathcal{M}_{n-2}^\bullet$. Soient b le nœud binaire père de ℓ, p le père de b et d le fils gauche de b. L'arbre $G^{-1}(T)$ est obtenu en supprimant ℓ et b. Le nœud d devient alors le nouveau fils (gauche, droit ou unique selon ce qu'était b) de p. L'arbre est considéré comme pointé sur d.
- Si T est pointé en rouge sur une feuille fille gauche ℓ, son image est dans $\mathcal{M}_{n-2}^\bullet$. Soient b le nœud binaire père de ℓ, p le père de b et d le fils droit de b. L'arbre $G^{-1}(T)$ est obtenu en supprimant ℓ et b. Le nœud d devient alors le nouveau fils (gauche, droit ou unique selon ce qu'était b) de p. L'arbre est considéré comme pointé sur d.
- Si T est pointé en bleu sur une feuille fille gauche ℓ, son image est dans $\mathcal{M}_{n-2}^\bullet$. Soient b le nœud binaire père de ℓ, p le père de b et d le fils droit de b. L'arbre $G^{-1}(T)$ est obtenu en supprimant ℓ et b. Le nœud d devient alors le nouveau fils (gauche, droit ou unique selon ce qu'était b) de p. L'arbre est considéré comme pointé sur d.

□

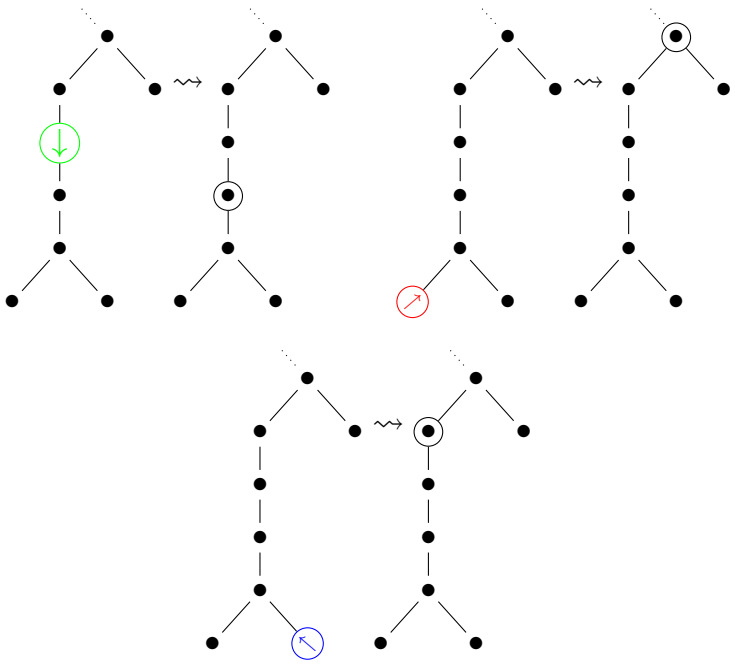


FIGURE 6.2 – Des arbres pointés en couleur de $\mathcal{M}^\star$ et leur arbres pointés associés de $\mathcal{M} + \mathcal{M}^\bullet$.

6.1.3 Le repointage

Nous allons maintenant donner la bijection correspondant à l'identité :

$$\mathcal{M}^\star = \mathcal{M}^\bullet + \mathcal{M}.$$

Cette bijection est une extension de l'opération de repointage présentée à la section 5.1.2 du chapitre précédent pour pouvoir également prendre en compte les nœuds unaire. Lorsqu'on l'applique à un arbre de Catalan (tout arbre de Catalan est un arbre de Motzkin) on a bien exactement la même opération.

Nous partons donc d'un arbre pointé en couleur T de $\mathcal{M}^\star$ et nous allons obtenir un arbre pointé ou pas de même forme.

Par convention, on considère que la racine est un fils droit, et on ordonne les ancêtres de chaque nœud depuis celui-ci jusqu'à la racine.

L'opération de repointage pour un arbre de Motzkin est définie comme suit.

- Si T est pointé en bleu sur une feuille ℓ, soit v son premier ancêtre qui est un fils gauche, ou $v = \perp$ si il n'existe pas de tel ancêtre.
- Si T est pointé en rouge sur une feuille ℓ, soit v son premier ancêtre qui est un fils droit. Comme la racine est un fils droit, un tel ancêtre existe toujours.
- Si T est pointé en vert sur un nœud unaire u, soit v son fils.

Dans chacun de trois cas, l'arbre est maintenant considéré pointé sur v, ou non pointé si $v = \perp$. C'est donc bien un élément de $\mathcal{M}^\bullet + \mathcal{M}$.

Proposition 6.1. *Le repointage décrit ci-dessus est une bijection.*

Démonstration. Soit v un nœud de T ou $v = \perp$. On prouve qu'il existe une unique manière de pointer en couleur une feuille de T tel quel son repointage donne (T, v). On va pour cela distinguer quatre cas, les trois premiers étant identiques au cas des arbres de Catalan.

- Si $v = \perp$, la feuille pointée en couleur correspondante ℓ doit être pointée en bleu et ne pas avoir de fils gauche parmi ses ancêtres. On en déduit que ℓ est la feuille la plus à droite de T.
- Si v est un fils gauche, la feuille pointée en couleur correspondante ℓ doit pointée en bleu et être la descendante de v telle que la branche de v à ℓ ne contient pas de fils gauche. Ce qui signifie que ℓ est la descendante la plus à droite de v.
- De manière similaire, si v est un fils droit, la feuille pointée en couleur correspondante ℓ doit être pointée en rouge et être la descendante la plus à gauche de v.
- Si v est un fils unique, le point d'origine est vert et est placé sur son père.

On a donc bien une bijection. □

6.2 Génération aléatoire

Nous allons présenter dans cette section un algorithme de génération aléatoire uniforme d'arbre de Motzkin, de complexités en temps, espace et nombre de bits aléatoires linéaires. On reprend pour cela le modèle de complexité du chapitre précédent, en rappelant que l'on peut tirer un entier aléatoire de $\{1, \ldots, 9\}$ en temps constant.

Théorème 6.2. *L'algorithme 15 renvoie un arbre de Motzkin de taille n uniformément distribué. Sa complexité en temps est linéaire en moyenne.*

La preuve est très similaire à celle du théorème 5.2 du chapitre précédent. Dans la suite, nous passons rapidement sur les parties similaire pour se concentrer sur les différences. On commence par établir deux lemmes, similaire à ceux du cas binaires.

Lemme 6.3. *Soit T un arbre de Motzkin pointé en couleur à n nœuds. Une passe dans l'algorithme 14 atteint T avec probabilité :*

$$\mathbb{P}(T) = \frac{1}{2 \cdot 3^{n-1}}.$$

Démonstration. Par récurrence sur n. □

Lemme 6.4. *Supposons que l'algorithme 15 ait renvoyé un arbre de taille n avec succès. La génération a utilisé $\mathcal{O}(n)$ opérations.*

Démonstration. Comme pour la preuve du lemme 5.4,on doit prouver que, au cours de l'algorithme, le point parcourt une distance en $\mathcal{O}(n)$ lors des repointages. Cette fois encore, nous considérons la profondeur du nœud pointé dans l'arbre T en train d'être généré. Sa profondeur :

- augmente de un à chaque fois qu'une opération $G1$, $G3$, $G4$ ou $G5$ est appliquée ;
- augmente de un à chaque fois que l'on doit repointer un point vert ;

Algorithm 14: Un essai de tirage d'un arbre de Motzkin

Input: Une taille n.
Output: Un arbre unaire-binaire de taille n, $n+1$ ou **FAIL**

T := une feuille pointée en rouge ou en bleu (avec probabilité 1/2 chacun)
While $|\mathrm{T}| < n$ **do :**
  Choisir l'un des cas suivants :
    Avec probabiblité $\frac{1}{3}$,
      T := $G1(T)$
    Avec probabiblité $\frac{1}{3}$,
      (T, v) = repointer T
      si $v = \perp$ **return FAIL**
      T := $G2(T, v)$
    Avec probabiblité $\frac{1}{9}$,
      (T, v) = repointer T
      si $v = \perp$ **return FAIL**
      T := $G3(T, v)$
    Avec probabiblité $\frac{1}{9}$,
      (T, v) = repointer T
      si $v = \perp$ **return FAIL**
      T := $G4(T, v)$
    Avec probabiblité $\frac{1}{9}$,
      (T, v) = repointer T
      si $v = \perp$ **return FAIL**
      T := $G5(T, v)$
return T

Algorithm 15: Arbre de Moztkin aléatoire

Input: Une taille n.
Output: Un arbre unaire-binaire de taille n

T =Un essai de tirage d'un arbre de Motzkin (n)
While T = **FAIL** ou $|T| = n+1$ **do :**
    T =Un essai de tirage d'un arbre de Motzkin (n)
return T

– reste identique si $G2$ is appliqué ;
– décroit lorsque le point voyage lors du repointage.

De plus, un point vert ne peut être produit que par $G2$, ce qui fait que la profondeur du point augmente au plus une fois par itération. Comme la profondeur est positive ou nulle, le point ne se déplace vers le haut qu'au plus n fois.

□

Preuve du théorème 5.2. Par le lemme 6.3, tous les arbres pointés en couleur à n nœuds sont générés avec la même probabilité. Comme chaque arbre a $n+1$ points possibles, l'uniformité est prouvée.

Le nombre d'arbres de Motzkin à n noeuds est le n-ième nombre de Motzkin M_n, qui est de l'ordre de $3^n n^{-3/2}$. En utilisant le lemme 6.3 on prouve, de manière identique à ce que l'on a fait dans la preuve du théorème 5.2, que la probabilité de succes d'un *run* de l'algorithme 14 est :

$$\frac{(n+1)M_n}{2 \cdot 3^{n-1}} \approx n^{-1/2},$$

et qu'il y a donc en moyennes $\mathcal{O}(\sqrt{n})$ passes qui échouent.

Trouvons maintenant le cout moyen d'une passe qui échoue. Si $k < n$, la proposition 5.1 et le lemme 6.3 montrent que la probabilité d'échouer à l'étape k est :

$$\frac{M_k}{2 \cdot 3^{k-1}} \approx k^{-3/2}.$$

De plus, il y a $(n+2)M_{n+1}$ arbres pointés en couleur de taille $n+1$. Par le lemme 6.3, la probabilité d'échouer en atteignant la taille $n+1$ est au plus :

$$\frac{(n+2)M_{n+1}}{2 \cdot 3^{n-1}} \approx n^{-1/2}.$$

Par le lemme 6.4, le coût moyen d'une passe qui échoue est donc :

$$\sum_{k=0}^{n-1} \mathcal{O}(k \cdot k^{-3/2}) + \mathcal{O}(n \cdot n^{-1/2}) = \mathcal{O}(\sqrt{n}).$$

On en déduit que le complexité totale de l'algorithme 15 est $\mathcal{O}(\sqrt{n})$ fois $\mathcal{O}(\sqrt{n})$, soit $\mathcal{O}(n)$. □

Cette fois encore, l'algorithme nous permet de tirer uniformément des arbres pointés en transformant le point une dernière fois avant de renvoyer l'arbre.

6.2.1 Structure de données et implémentation pratique

Pour implémenter nos algorithmes, on représente un arbre comme un tableau de nœuds, chaque nœud contenant quatre valeurs :

– un entier représentant son arité (0, 1 ou 2 si le nœud est une feuille, unaire ou binaire),
– l'indice de son père (avec une valeur distinguée pour la racine),

– deux indices pour ses fils éventuels.

Un nœud a donc besoin de 128 bits pour être stocké si l'arbre une taille inférieure à 2^{32}. [1] Ces informations suffisent à effectuer en temps constants les opérations que nous souhaitions.

Notre implémentation permet de générer un arbre de taille environ 10 millions en environ 5 secondes sur un PC standard. Ce temps augmente significativement si l'on souhaite écrire ou dessiner l'arbre.

6.3 Conclusion et perspectives

Au cours de cette partie nous avons vu des spécifications basées sur des équations holonomes traduisant un processus de greffes (*i.e.*, de transformations locales) plutôt que des spécifications algébriques traduisant des processus de branchements pour les arbres de Catalan et les arbres de Motzkin. Ces spécifications nous mènent vers des générateurs aléatoires uniformes très efficaces, de complexité linéaire à la fois en espace temps et nombre de bits aléatoires utilisés. Comme toute série génératrice algébrique est également holonome, nous espérons pouvoir trouver d'autres spécifications de ce type. À long terme, on peut même imaginer définir toute une famille de classes combinatoire pouvant être générées « à la Rémy », faisant de cet algorithme connu comme *ad hoc* une nouvelle méthode générique de génération aléatoire. La principale difficulté est que l'on doit pour cela trouver une interprétation combinatoire aux équations holonomes. Ceci n'est pas toujours possible, et même pour des classes combinatoires très simples, comme la classe de arbres de Motzkin, ce ne fut pas élémentaire.

De plus, dans le cas des arbres de Catalan, le générateur que l'on obtient a une complexité en nombre de bits aléatoires quasi-optimale. Nous sommes actuellement en train d'étudier d'autres classes pour lesquelles on pourrait obtenir un générateur quasi-optimal. Pour cela, la première extension possible est de considérer les mots équilibrés, c'est-à-dire les mots à deux lettre ayant autant de a que de b. Comme la classe de tels mots est en bijection avec les arbres binaires, on peut directement réutiliser notre générateur, mais écrire un générateur spécifique nous permettra d'étudier des variations autour de ce cas, tels que les mots équilibrés à 2^n lettres.

D'autres classes sont usuellement traitées (pour l'énumération, par exemple) par des équations holonomes telles que les chemins, les méandres ou les permutations évitant certains motifs. Donner un générateur aléatoire sur le principe de greffe pour une de ces classes est déjà un grand défi, et présenterait certainement une avancée significative pour leur génération aléatoire.

Nous allons maintenant brièvement présenter des travaux sur la spécification bivariée d'arbres de Motzkin, n'ayant pas encore abouti à des générateurs aléatoires efficaces.

1. On remarque que la complexité en espace est donc en fait en $O(n \log n)$ si l'on considère vraiment la taille des pointeurs. Comme en pratique l'algorithme doit être implémenté différemment dans le cas où on retire cette borne, que stocker un arbre plus gros est difficile et que les applications potentielles ne le justifient pas nous avons fait le choix de considérer ces pointeurs de taille $O(1)$, car on utilise une taille fixe quelque soit la taille de l'arbre que l'on souhaite généré, dans l'intervalle $[1, 2^{32}]$.

6.3.1 Spécifications pondérées

Afin de pouvoir simuler des distributions non-uniformes, on peut vouloir pondérer des nœuds ou des configurations. Pour cela nous utilisons des spécifications bivariées où la marque $\mathcal{U}$ pourra être ensuite remplacée par un réel bien choisi pour approcher la distribution voulue. Nous présentons ici deux modèles pondérés différents : dans l'un d'entre eux les nœuds unaires sont pondérés alors que dans l'autre ce sont les opérateurs de greffes. Pour le moment, ces spécifications ne mène pas à une génération aléatoire efficace.

Nœuds unaires pondérés

Nous présentons ici comment adapter la bijection de la section 6.1 à la classe $\mathcal{A}$ des arbres unaires-binaires dont les noeuds unaires sont marqués par $\mathcal{U}$. On a :

$$\mathcal{A}^\star = 2\mathcal{Z} + \mathcal{Z}\mathcal{U}\mathcal{A}^\star + \mathcal{Z}\mathcal{U}\mathcal{A}^\bullet + 4\mathcal{Z}^2\mathcal{A}^\bullet - \mathcal{Z}^2\mathcal{U}^2\mathcal{A}^\bullet \tag{6.2}$$

où $\mathcal{A}^\star = \mathcal{A}^\bullet + \mathcal{A}$.

En fait, le terme négatif $-\mathcal{Z}^2\mathcal{U}^2\mathcal{M}^\bullet$ peut être réécrit comme $-\mathcal{Z}\mathcal{U}(\mathcal{Z}\mathcal{U}\mathcal{M}^\bullet)$ qui, en reprenant la bijection pour les arbres de Motzkin, correspond à l'opération créer un nœud unaire pointé en vert puis le transformer en nœud binaire. On peut donc réécrire l'équation 6.2 comme :

$$\mathcal{A}^\star = 2\mathcal{Z} + \mathcal{Z}\mathcal{U}(\mathcal{A}^{(\ell)} + \mathcal{A}^{(\ell)}) + \mathcal{Z}\mathcal{U}\mathcal{A}^\bullet + 4\mathcal{Z}^2\mathcal{A}^\bullet$$

où $\mathcal{A}^{(\ell)}$ est la classe des arbres unaire-binaire pondérés pointés en bleu sur une feuille et $\mathcal{A}^{(\ell)}$ la classe des arbres unaire-binaire pondérés pointés en rouge sur une feuille.

Nous nous basons sur cette équation et décrivons maintenant la bijection

$$H : 2\mathcal{Z} + \mathcal{Z}\mathcal{U}(\mathcal{A}^{(\ell)} + \mathcal{A}^{(\ell)}) + \mathcal{Z}\mathcal{U}\mathcal{A}^\bullet + 4\mathcal{Z}^2\mathcal{A}^\bullet \to \mathcal{A}^\star,$$

très similaire à la bijection G.

- $H1_{n,k} : \mathcal{A}^{(\ell)}_{n-1,k-1} \to \mathcal{A}^\star_{n,k}$ On part d'un arbre pointé en rouge sur une feuille ℓ. L'opération $H1$ est équivalente à l'opération $G1$ sur un arbre pointé en rouge.
- $H2_{n,k} : \mathcal{A}^{(\ell)}_{n-1,k-1} \to \mathcal{A}^\star_{n,k}$ On part d'un arbre pointé en bleu sur une feuille ℓ. L'opération $H2$ est équivalente à l'opération $G1$ sur un arbre pointé en bleu.
- $H3_{n,k} : \mathcal{A}^\bullet_{n-1,k-1} \to \mathcal{A}^\star_n$ L'opération $H3_{n,k}$, équivalente à $G2$ consiste à ajouter un noeud unaire pointé en vert au-dessus du noeud pointé.
- $H4_{n,k} : \mathcal{A}^\bullet_{n-2,k} \to \mathcal{A}^\star_{n,k}$ Cette opération est similaire à $G3$. On part d'un arbre pointé sur un nœud d, de père p et on ajoute un nœud binaire et une feuille à droite de l'endroit pointé. L'arbre est maintenant considéré pointé en bleu sur la nouvelle feuille.
- $H5_{n,k} : \mathcal{A}^\bullet_{n-2,k} \to \mathcal{A}^\star_{n,k}$ Cette opération est similaire à $G4$. On part d'un arbre pointé sur un nœud d, de père p et on ajoute un nœud binaire et une feuille à gauche de l'endroit pointé. L'arbre est maintenant considéré pointé en rouge sur la nouvelle feuille.

- $H6_{n,k} : \mathcal{A}^{\bullet}_{n-2,k} \to \mathcal{A}^{\star}_{n,k}$ Cette opération est similaire à $G5$. On part d'un arbre pointé sur un nœud d, de père p et on ajoute un nœud binaire et une feuille à gauche de l'endroit pointé. L'arbre est maintenant considéré pointé en bleu sur la nouvelle feuille.
- $H7_{n,k} : \mathcal{A}^{\bullet}_{n-2,k} \to \mathcal{A}^{\star}_{n,k}$ On part d'un arbre pointé sur un nœud d, de père p. On crée un nouveau noeud binaire b et une nouvelle feuille ℓ tel que b devient le père de d ; p devient le père de b et ℓ est le fils droit de b. Le nouvel arbre est considéré pointé en rouge sur ℓ.

Cette spécification nous permet immédiatement d'obtenir un générateur aléatoire, similaire à l'algorithme 14 en ajoutant un rejet lorsque l'on cherche à greffer sur un nœud unaire pointé en vert, et en pondérant les probabilités en fonction des coefficients des termes correspondants. Cependant, cet algorithme a une complexité exponentielle en moyenne, et est donc moins efficace que les algorithmes classiques, comme par exemple la méthode de Boltzmann bivariée [BP10], rejet anticipé [BPS92] où méthodes via mots de Łukasiewicz [BDM14]. Une de nos perspective est bien sur de chercher à adapter ce générateur pour le rendre efficace.

Arbres de Ford

Les arbres de Ford [For06, PW09] correspondent à des processus probabilistes, pouvant être décrit comme des processus où à chaque étape on effectue une greffe de Rémy dans un arbre binaire mais où chaque nœud binaire à probabilité proportionnelle à α d'être choisi et chaque feuille à $1-\alpha$. Nous considérons ici un modèle très légèrement différent où lorsque l'on fait grossir un arbre de taille $2n+1$ chaque nœud binaire à probabilité proportionnelle à αn d'être choisi et chaque feuille à $(1-\alpha)(n+1)$.

On peut obtenir une spécification différentielle pour les arbres de Ford finis en modifiant légèrement celle correspondant à la bijection de Rémy :

$$\mathcal{U}\frac{\partial \mathcal{F}}{\partial \mathcal{U}} = \mathcal{Z}\mathcal{U} + \alpha\mathcal{V}^2\mathcal{Z}^2\mathcal{U}\frac{\partial \mathcal{F}}{\partial \mathcal{V}} + (1-\alpha)\mathcal{V}\mathcal{Z}^2\mathcal{U}^2\frac{\partial \mathcal{F}}{\partial \mathcal{U}}$$

où $\mathcal{F}$ est la classe des arbres de Ford où les nœuds binaires sont marqués par $\mathcal{U}$ et α est une constante.

Cette spécification devrait permettre l'analyse asymptotique de propriétés de ces arbres. Des travaux en cours devraient notamment permettre de montrer par la combinatoire analytique que leur hauteur moyenne est en $O(n^{\alpha})$.

Troisième partie

λ-termes

Chapitre 7

Introduction : λ-termes

Le λ-calcul a été inventé dans les années 30 par Church et Kleene [Chu36, Kle35a, Kle35b] et a aujourd'hui un rôle fondamental en informatique théorique. Il s'agit d'un langage reposant sur l'ensemble des λ-termes et constitué de règles de manipulation. Cette construction donne un système très proche des arbres de preuves (voir par exemple [LBB$^+$13, Partie 2]). Le λ-calcul se révèle être un outil très puissant pour décrire et prouver des programmes, analyser des langages de programmation, et faire des preuves formelles en logique. Toute la programmation linéaire, comme par exemple LISP, est fondée dessus.

Pourtant, peu de propriétés statistiques sont aujourd'hui connues sur ces objets. Nous nous intéressons ici à l'énumération et la génération aléatoire de λ-termes. Cette étude rentre dans le cadre du champ de recherche récent sur la logique quantitative, *i.e.*, l'évaluation de différents paramètres caractérisants des systèmes de logique [FGGZ07, GK09, MTZ00]. L'énumération des λ-termes en général est un problème extrêmement complexe qui a été posé par Ph. Flajolet en 2006. Seules des bornes larges sont connues (*cf.* la discussion autour de ce problème dans [BGG11] et pour un problème similaire dans [GL12, fin de la Sec. 3]). C'est pourquoi nous avons opté dans cette thèse pour une simplification du modèle. Les classes restreintes de λ-termes que l'on a abordées sont à la fois plus abordables d'un point de vue analytique et elles ont néanmoins un sens du point de vue du λ-calcul.

Dans cette thèse, nous nous concentrons uniquement sur le langage des λ-termes indépendamment des règles de calcul associées. En particulier, les termes pourront ne pas être typables, et les termes équivalents (au sens par exemple de la $\beta-$reduction) sont considérés comme différents. L'objectif de cette analyse quantitative est de pouvoir, par la suite, étudier des densités de classes ayant certaines propriétés, mais aussi de créer une boîte à outils potentiellement réexploitable sur des classes plus complexes (par exemple les lambda-termes typables).

Nous abordons, dans la suite de ce chapitre, les définitions de classes de λ-termes ainsi que des spécifications bivariées pour celles-ci. Nous présentons dans le chapitre 8 une bijection entre certaines classes de cartes combinatoires et des sous-classes de λ-termes : les λ-termes linéaires et affines, définis en section 7.4. Nous en déduisons leur énumération mais également des générateurs aléatoires uniformes efficaces. Dans le chapitre 9 nous nous intéressons à une extension des λ-termes linéaires et affines, pour

lesquels nous donnons des spécifications reposant sur un principe de greffe. Ce même principe nous permet également d'obtenir une spécification des λ-termes clos. Dans un cas, celui des λ-termes multi-linéaires, on est alors capable d'obtenir le comportement asymptotique de leur séquence d'énumération. Nous donnons également de nouvelles bornes pour le comportement asymptotique du nombre de λ-termes.

7.1 Définition

L'ensemble des *λ-termes* est défini comme :
- pour toute variable x, x est un λ-terme, on dit alors que c'est la *variable libre* x
- si L_1 et L_2 sont des λ-termes, $L_1\ L_2$ est un lambda terme, on appelle ce terme l'*application* de L_1 à L_2,
- si L est un λ-terme, pour toute variable x, $\lambda x.L$ est un lambda terme, on appelle ce terme l'*abstraction* de x dans L : toute les occurrences de la variable libre x dans L deviennent liées (et ne sont donc plus libres).

La taille d'un λ-terme est son nombre total de variables (libres ou liées), d'abstractions et d'applications.

On considère les λ-termes à renommage des variables liées près. Par exemple, comme chaque variable lie seulement les variables libres de même nom du sous-terme auquel il s'applique, les termes $\lambda y.(\lambda x.x * \lambda z.y)$, $\lambda y.(\lambda x.x * \lambda x.y)$ et $\lambda x.(\lambda y.y * \lambda z.x)$ sont identiques.

Un λ-terme est dit *clos* s'il ne contient pas de variable libre. Par exemple, $(\lambda x.(x * x) * \lambda y.y)$ est un λ-terme clos alors que $(\lambda x.(x * z) * \lambda y.y)$ n'en est pas un.

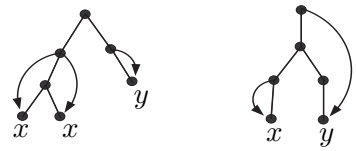
Remarque. Ici, on s'intéresse à l'ensemble de tous les λ-termes, normalisables ou pas, indépendamment de la sémantique.

Des modèles avec une notion de taille différente (les variables ne comptent pas) ont été étudiés dans [DGK$^+$10, GL12]. Dans [DGK$^+$10] des bornes supérieure et inférieure sur le nombre de λ-termes de taille donnée ont été trouvées et des questions comme celle de la proportion de termes typables ont été discutées. L'article [GL12] aborde la question de l'énumération par une représentation des termes utilisant les indices de De Bruijn. Les auteurs y obtiennent une relation de récurrence pour les termes avec ou sans contraintes sur le nombre de variables libres et évoquent également la question de la génération aléatoire des termes, ce qui leur permet une étude expérimentale de différentes propriétés telles que la typabilité et des caractéristiques de taille.

7.2 Arbre syntaxique

A chaque lambda-terme on associe un *arbre syntaxique* : l'arbre syntaxique d'un λ-terme est un arbre unaire-binaire planaire enraciné où :
- Une feuille représente une variable, on lui associe une étiquette correspondant au nom de la variable,
- un nœud binaire représente l'application du terme correspondant à son fils gauche à celui correspondant à son fils droit,

– un nœud unaire est associé au nom d'une variable, et représente l'abstraction de cette variable dans le terme correspondant à son arbre fils.



$(\lambda x.(x * x) * \lambda y.y)$ $\lambda y.(\lambda x.x * \lambda z.y)$

FIGURE 7.1 – Deux arbres syntaxiques et leur λ-termes correspondant. Notez que les étiquettes sur les nœuds peuvent être oubliées puisque $(\lambda x.(x * x) * \lambda y.y)$ et $(\lambda a.(a * a) * \lambda b.b)$ représent le même terme.

Avec cette représentation, on peut remplacer le nom des variables liées par un lien entre un nœud unaire et les feuilles liées par l'abstraction. Ces liens lient toujours des feuilles du sous-arbre du nœud unaire.

FIGURE 7.2 – Les mêmes arbres syntaxiques que dans figure 7.1.

7.3 Spécification bivariée

En suivant la définition précédente qui construit un λ-terme par la racine de manière non-ambigüe on obtient la spécification bivariée suivante :

$$\Lambda(\mathcal{Z},\mathcal{U}) = \mathcal{U}\mathcal{Z} + \mathcal{Z}\Lambda(\mathcal{Z},\mathcal{U}) \times \Lambda(\mathcal{Z},\mathcal{U}) + \mathcal{Z}\Lambda(\mathcal{Z}, 1+\mathcal{U})$$

où $\Lambda(\mathcal{Z},\mathcal{U})$ est la série bivariée des λ-termes comptés selon leur taille et le nombre de variables libres. Il faut noter que l'on omet ici le nom des variables liées, mais aussi ceux des variables libres. Aussi les termes $z(\lambda x.x)$ et $y(\lambda x.x)$ sont représentés de la même façon. Pour obtenir de "vrais" λ-termes à k variables libres ils faudrait k-colorer les $\mathcal{U}$, ce qui reviendrait à faire une substitution $\mathcal{U} \to \mathcal{U}_1 + \mathcal{U}_2 + ... + U_k$.

La classe $\Lambda(\mathcal{Z}, 0)$ est la classe des λ-termes clos : l'omission du nom des variables est ici non ambigüe car il n'y a pas de variable libre. Les termes clos sont spécialement intéressants car tous les termes sémantiquement corrects sont clos. Pour faire un parallèle avec les programmes, une variable libre se rapproche d'une variable non définie, ce qui n'est pas correct sémantiquement. Dans la suite, nous nous intéressons uniquement aux termes clos.

On peut, par traduction directe, obtenir des équations sur les séries génératrices des λ-termes généraux et clos à partir de ces spécifications.

7.4 λ-termes linéaires et affines

7.4.1 Définition des λ-termes linéaires

Définition 7.1. Un *λ-terme linéaire* est un λ-terme clos où chaque abstraction lie exactement une variable.

Par exemple $\lambda x.(\lambda z.(xz))$ est un λ-terme linéaire alors que $\lambda x.(x\lambda z.(xz))$ n'en est pas un.

Une spécification bivariée des λ-termes linéaires avec variables libres, où $\mathcal{Z}$ compte la taille et $\mathcal{U}$ les variables libres, est :

$$\mathcal{L}(\mathcal{Z},\mathcal{U}) = \mathcal{U}\mathcal{Z} + \mathcal{Z}\mathcal{L}(\mathcal{Z},\mathcal{U}) \times \mathcal{L}(\mathcal{Z},\mathcal{U}) + \mathcal{Z}\frac{\partial \mathcal{L}}{\partial \mathcal{U}}(\mathcal{Z},\mathcal{U}); \tag{7.1}$$

puis la classe des λ-termes linéaires est $\mathcal{L}(\mathcal{Z}, 0)$. En effet, un λ-terme linéaire pouvant contenir des variables libres est soit :

- réduit à une feuille, ce qui correspond au terme $\mathcal{U}\mathcal{Z}$,
- l'application d'un λ-terme linéaire pouvant contenir des variables libre à un autre, autrement dit un arbre de racine binaire, ce qui correspond au terme $\mathcal{Z}\mathcal{L}(\mathcal{Z},\mathcal{U}) \times \mathcal{L}(\mathcal{Z},\mathcal{U})$,
- l'abstraction d'une variable apparaissant exactement une fois dans un λ-terme linéaire contenant des variables libres : cela revient à choisir une marque $\mathcal{U}$ et à la supprimer, la variable n'étant plus libre, en faisant à la place apparaitre un lien de cette feuille au nœud unaire. On obtient donc le terme $\mathcal{Z}\frac{\partial \mathcal{L}}{\partial \mathcal{U}}(\mathcal{Z},\mathcal{U})$.

On remarque qu'un λ-terme linéaire est forcément de taille $3n + 2$: dans l'arbre syntaxique, il y a une feuille de plus que de nœuds binaires (puisque c'est un arbre unaire-binaire) et autant de nœuds unaires que de feuilles (puisque chaque nœud unaire lie une feuille est chaque feuille est liée).

Ces λ-termes correspondent, via l'isomorphisme de Curry-Howard (voir par exemple [LBB+13, Partie 2]) aux termes du système de logique *BCI* : les preuves de logique linéaire obtenues à partir de axiomes B, C et I [Hin93]. Ces termes sont très étudiés en logique combinatoire [II65b, II65a, IT79].

7.4.2 Suite de Wright-Louchard-Takács

À partir de la spécification précédente, on peut calculer les premiers coefficients de la séries génératrice, *i.e.*, le nombre de λ-termes linéaires de taille $2, 5, 8, 11, 14, 17, 20...$ est $1, 5, 60, 1105, 27120, 828250....$ Il s'agit de la séquence de Wright-Louchard-Takács apparraisant dans les articles [FL01, VM10, FPV98] dans différents cadres : l'énumération asymptotique de graphes connexes par excès, les moments des longueurs de chemins et d'inversions dans les arbres, les moments de l'aire sous une excursion (discrète ou continue), l'énumération de digrammes trivalents...

On peut obtenir directement l'asymptotique du nombre de λ-termes linéaires en résolvant l'équation différentielle de Ricatti 7.1, qui a une solution explicite :

$$T(x) = \frac{1}{2x} - \frac{\mathrm{Ai}'(1/4x^2)}{\mathrm{Ai}(1/4x^2)}$$

où Ai est la fonction d'Airy. Dans le chapitre 8 nous retrouverons le comportement asymptotique des λ-termes linéaires par une méthode bijective, qui permet également d'en tirer un générateur aléatoire efficace et s'étend au λ-termes affines.

7.4.3 Définition des λ-termes affines

Définition 7.2. Un λ-*terme affine* est un λ-terme clos où chaque abstraction lie au plus une variable.

Par exemple $\lambda x.(\lambda z.(xz))$ et $\lambda x.\lambda y.(\lambda z.(xz))$ sont des λ-termes affines.

Une spécification bivariée des λ-termes affines, où $\mathcal{Z}$ compte la taille et $\mathcal{U}$ les variables libres, est :

$$\mathcal{A}(\mathcal{Z},\mathcal{U}) = \mathcal{U}\mathcal{Z} + \mathcal{Z}\mathcal{A}(\mathcal{Z},\mathcal{U}) \times \mathcal{A}(\mathcal{Z},\mathcal{U}) + \mathcal{Z}\frac{\partial \mathcal{A}}{\partial \mathcal{U}}(\mathcal{Z},\mathcal{U}) + \mathcal{Z}\mathcal{A}(\mathcal{Z},\mathcal{U})$$

puis la classe des λ-termes affines est $\mathcal{A}(\mathcal{Z}, 0)$.
Cette spécification se trouve de manière similaire à celle des λ-termes linéaires.

Comme pour les λ-termes linéaires, on a ici aussi une correspondance avec un système de logique, le système *BCK*.

Remarque. La classe des λ-termes affines peut être vue comme la classe des λ-termes linéaires où chaque nœud (*i.e.*, atome) est remplacé par une séquence non vide de nœuds, correspondant à la chaîne de nœuds unaires non lié le précédent. On a donc :

$$\mathcal{A}(\mathcal{Z}) = \mathcal{L}(\mathcal{Z} \times \textsc{Seq}(\mathcal{Z})).$$

Chapitre 8

Méthode bijective pour λ-termes linéaires et affines

Dans ce chapitre nous présentons une bijection permettant d'énumérer et de générer aléatoirement les λ-termes linéaires et affines. Ces résultats ont été publiés dans [BGJ10].

8.1 Introduction

Les arbres syntaxiques des λ-termes , où l'on a dessiné explicitement les liens comme des arêtes, peuvent être vus comme des cartes combinatoires enracinées, *i.e.*, des graphes plongés dans un plan (ou une autre surface orientable), dont une demi-arête est pointée.

Le résultat principal de ce chapitre est de montrer qu'après une transformation bijective de la carte associée à un λ-terme linéaire, on obtient une classe de cartes facile à décrire. Cela permet d'avoir une forme explicite et l'asymptotique des coefficients, ainsi qu'un générateur aléatoire efficace pour les λ-termes linéaires. On peut en fait étendre ce résultat à un ensemble de classes de cartes combinatoires et à un ensemble de classes d'arbres enrichis, qui comprend les λ-termes affines. Cette bijection repose sur un parcours en profondeur déterministe de la carte combinatoire, qui permet de faire apparaître la structure d'arbre enrichi.

Pour les λ-termes linéaires on retombe ainsi sur la séquence de Wright-Louchard-Takács, apparaissant par exemple dans [FPV98, VM10]. Par la vision en arbres enrichis, cela correspond également à la solution d'équations différentielles non linéaires. On peut ensuite utiliser cette correspondance pour générer aléatoirement des λ-termes linéaires ou affines en temps linéaire en la taille du λ-terme généré.

8.2 Cartes combinatoires

Les *Cartes combinatoires* ont été introduites dans les années 60, pour étudier en particulier les graphes planaires [Edm60, CFGN10]. Elle ont depuis été intensément

étudiées pour leur propriétés propres, notamment en genre donné. Ici, nous ne nous restreignons pas aux cartes planaires et nous n'étudions pas le genre des cartes obtenues.

8.2.1 Définitions

Une carte est un graphe connexe (ou les boucles et multi-arêtes sont autorisées), plongé dans une surface 2-orientable vu à homotopie (déformation continue de la surface) près. Les cartes non-étiquetées non-enracinées sont assez compliquées à étudier dû à leur nombreuses symétries, en particulier leur analyse asymptotique a été le sujet de plusieurs articles [BC78, BFSS01]. Ici, nous nous plaçons dans le cadre beaucoup plus simple des cartes enracinées. Dans la vision graphes, une carte étiquetée est une carte où chaque demi-arête porte un numéro entre 1 et $2n$ où n est le nombre d'arêtes, chaque numéro n'apparaissant qu'une seule fois, *i.e.*, une arête $e = (u, v)$ porte deux étiquettes, une du coté de u et une du coté de v.

La définition des cartes étiquetées peut être formulée ainsi :

Définition 8.1. Une *carte étiquetée* est un triplet (E, σ, α) tel que :
- E est un ensemble fini de demi-arêtes (distinguées les unes des autres),
- σ est une permutation de E,
- α est une involution sans point fixe de E.
- (σ, α) engendre un sous-groupe transitif du groupe des permutations.

Le lien entre la vision graphe et la définition précédente est le suivant : un cycle de σ correspond à un sommet (représenté par ses demi-arêtes adjacentes dans leur ordre d'incidence) et un cycle de α correspond à une arête (représenté par l'ensemble de ses deux demi-arêtes). La dernière condition correspond à la connexité (ou peut atteindre toute demi-arête depuis n'importe quel demi-arête). Un exemple est donné figure 8.1.

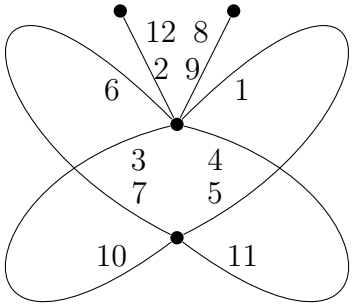


FIGURE 8.1 – Une représentation de la carte $([|1..12|], \sigma, \alpha)$ à dergré contraints dans $\{1, 2, 4, 5\}$ où $\sigma = ((1, 9, 2, 6, 3, 4)(5, 7, 10, 11)(8)(12))$ et $\alpha = ((1, 5)(2, 12)(3, 10)(4, 11)(6, 7)(8, 9))$.

Remarque. Dans la suite, on utilisera $E = \{1, 2, ..2n\}$.

Dans ce chapitre nous nous concentrons plus particulièrement au cas des *cartes étiquetées à degrés contraints* : soit D un ensemble d'entiers strictement positifs, une carte (E, σ, α) est à degrés contraints à D si tous ses sommets ont un degré appartenant à D. Autrement dit, tous les cycles de σ sont de longueur inclus dans D.

Une *carte enracinée* est une carte dont une demi-arête est distinguée. Autrement dit, une carte enracinée est une carte (comptée par ses demi-arêtes) pointée. Plusieurs définitions équivalentes sont souvent utilisées pour étudier les cartes combinatoires : on peut parler d'un coin distingué (*i.e.*, un espace entre deux arêtes autour d'un sommet), ou d'une arête distinguée orientée.

Un *arête* est l'ensemble de deux demi-arêtes e, e' telles que $\alpha(e) = e'$.

Propriété 8.2. *La classe des cartes enracinées est rigide.*

Une preuve en sera donnée en section 8.3.1. Cette propriété est la raison pour laquelle le modèle de carte enracinée est beaucoup plus simple que celui sans racine : l'enracinement élimine toutes les symétries.

8.2.2 Spécification

On cherche ici à décomposer les cartes comme des objets constructibles, afin de pouvoir y appliquer des méthodes efficaces pour l'analyse asymptotique et pour la génération aléatoire. Un travail similaire, mais sous l'angle de la théorie des espèces, a été fait pour le cas des cartes enracinées à degrés contraints à 3 dans [Vid10, VM10] (alors appelées *triangular rooted diagrams*). On notera $\mathcal{C}_D$ la classe des cartes combinatoires à degrés contraints dans D, où la taille d'une carte est son nombre de demi-arêtes. On cherche ici une équation combinatoire pour $\mathcal{C}_D^\bullet$ la classe des cartes enracinées à degrés contraints dans D.

Le définition 8.1 nous conduit à considérer une permutation et une involution sur le même ensemble. On reconnait ici la définition du produit de Hadamard étiqueté, présenté dans le paragraphe 1.4.2. Rappelons les spécifications des involutions et permutations :

La spécification d'une involution est :

$$\mathcal{I} = \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z}))$$

car c'est tout simplement un ensemble de cycles de longueur 2. La série génératrice exponentielle associée est $\exp(\frac{x^2}{2})$.

De même, une permutation dont les cycles sont à longueur dans D a pour spécification :

$$\mathcal{P}_D = \textsc{Set}(\textsc{Cyc}_D(\mathcal{Z}))$$

dont la série associée est $\exp(\sum_{k\in D} \frac{x^k}{k})$. On remarque immédiatement que l'existence d'une forme close pour cette série dépend de l'ensemble D. En particulier, si D est fini, la série admet une forme close.

On souhaite maintenant faire un produit de Hadamard exponentiel de ces deux classes, mais on remarque qu'ainsi on peut obtenir des graphes plongés non connexes (avec par exemple la permutation (12)(34) et l'involution (12)(34)). On obtient donc en fait un ensemble de cartes. D'où :

$$\textsc{Set}(\mathcal{C}_D) = \mathcal{I} \odot \mathcal{P} = \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z})) \odot \textsc{Set}(\textsc{Cyc}_D(\mathcal{Z})). \tag{8.1}$$

Corollaire 8.3. *La classe $\mathcal{C}_D^\bullet$ des cartes enracinées à degrés contraints à D satisfait l'équation :*

$$\Big(\textsc{Seq}_D(\mathcal{Z}) \times \textsc{Set}(\textsc{Cyc}_D(\mathcal{Z}))\Big) \odot \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z})) = \mathcal{C}_D^\bullet \times \textsc{Set}(\mathcal{C}_D). \tag{8.2}$$

Démonstration. On applique en pointage au deux parties de l'équation (8.1) :

$$\textsc{Set}(\textsc{Cyc}_K(\mathcal{Z})) \odot \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z})) = \textsc{Set}(\mathcal{C}_K).$$

On vérifie facilement que, pour deux classes combinatoires étiquetées $\mathcal{A}$ et $\mathcal{B}$, $(\mathcal{A} \odot \mathcal{B})^\bullet \simeq \mathcal{A}^\bullet \odot \mathcal{B} \simeq \mathcal{A} \odot \mathcal{B}^\bullet$: comme les deux sous-objets sont étiquetés sur le même ensemble, si on distingue une étiquette dans l'un alors on peut considérer qu'elle l'est dans l'autre aussi. Et donc le terme de gauche devient :

$$(\textsc{Set}(\textsc{Cyc}_K(\mathcal{Z})))^\bullet \odot \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z})).$$

Puis, par les relations :

1. $\mathcal{Z}^\bullet = \mathcal{Z}$
2. $(\textsc{Set}(A))^\bullet = A^\bullet \times \textsc{Set}(A)$
3. $(\textsc{Cyc}_K(A))^\bullet = A^\bullet \times \textsc{Seq}_{K-1}(A)$ with $K - 1 = \{k - 1 | k \in K\}$

on obtient

$$\Big(\mathcal{Z} \times \textsc{Seq}_{K-1}(\mathcal{Z}) \times \textsc{Set}(\textsc{Cyc}_K(\mathcal{Z}))\Big) \odot \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z}))$$

pour le terme de gauche, et

$$\mathcal{C}_K^\bullet \times \textsc{Set}(\mathcal{C}_K)$$

pour le terme de droite. On en déduit le résultat. □

Dans la suite, nous nous intéresserons également aux cartes enracinées où la racine a un degré contraint à un ensemble R différent de la contrainte D des autres degrés. On peut facilement modifier l'équation (8.2) pour prendre en compte de telles cartes :

$$\Big(\textsc{Seq}_R(\mathcal{Z}) \times \textsc{Set}(\textsc{Cyc}_D(\mathcal{Z}))\Big) \odot \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z})) = \mathcal{C}_{R,D}^\bullet \times \textsc{Set}(\mathcal{C}_D). \tag{8.3}$$

où $\mathcal{C}_{R,D}^\bullet$ est la classe combinatoire des cartes enracinées dont la racine a un degré inclus dans R et tous ses autres degrés inclus dans D.

8.3 Parcours déterministe en profondeur d'abord et arbres enrichis

Le parcours en profondeur d'abord (*DFS*, pour Depth-First Search) est un algorithme classique pour explorer des graphes en informatique et en théorie des graphes, voir par exemple [Knu97a], [CLRS09] (ou des cours de licence).

En quelques mots, voici ce que fait l'algorithme :
Il prend un sommet en entrée.
Si le sommet est marqué, il s'arrête.
Il visite le sommet de l'arbre, et le marque.
Il fait un DFS récursivement sur chacun de ses voisins.

On en déduit un arbre couvrant du graphe. L'arbre couvrant ainsi obtenu dépend du choix du premier sommet visité et de l'ordre dans lequel on visite les voisins de chacun des sommets.

Dans le cadre des cartes enracinées, on peut donc aisément construire un parcours déterministe en profondeur d'abord : le premier sommet visité est celui adjacent à la racine, puis on parcours les arêtes dans l'ordre cyclique, à partir de celle d'arrivée (au premier round, la racine). C'est un parcours classique dans les cartes [Tut01].

Dans cette section nous allons décrire comment ce parcours déterministe en profondeur d'abord permet d'obtenir une nouvelle équation symbolique différentielle pour les cartes, que nous rapprocherons dans la section suivante de la définition des λ-termes linéaires et affines.

8.3.1 L'arbre couvrant en profondeur et droite d'abord

Un *arbre couvrant en profondeur d'abord* d'une carte est un arbre formé par un parcours maximal en profondeur d'abord depuis la racine.

Un *arbre couvrant en profondeur et droite d'abord* est l'arbre couvrant en profondeur d'abord où, à chaque fois que l'on a le choix sur la demi-arête suivante on choisit la prochaine dans l'ordre cyclique local induit par les permutations, *i.e.*, pour une carte (E, σ, α) la demi-arête visitée après $e \in E$ est soit $\sigma(e)$ soit $\alpha(e)$ en fonction de la parité du round.

Il est important de remarquer que grâce à l'enracinement de la carte, ce parcours est entièrement déterministe. En particulier cela prouve la rigidité des cartes enracinées.

Démonstration. Depuis une carte on peut obtenir un arbre couvrant enraciné planaire de manière déterministe. Pour ce faire, on visite toutes les demi-arêtes de l'arbre. L'ordre dans lequel on visite les demi-arêtes est déterministe. On peut ainsi les numéroter de 1 à n. Pour chaque permutation de $\{1, 2, ...n\}$ on voit les étiquettes dans un ordre différent. Il y a donc bien $n!$ cartes enracinées étiquetées différentes correspondant à une carte enracinée.

□

Soit T l'arbre couvrant en profondeur et droite d'abord. On peut maintenant partitionner les arêtes de la carte en deux parties : les arêtes appartenant à T, appelées arêtes de l'arbre, et celles n'appartenant pas à T, appelées arêtes liens (voir Fig. 8.2).

Remarque. Soit une carte (E, σ, α), si $e \in T$ alors $\alpha(e) \in T$, *i.e.*, si une demi-arête est dans T alors l'autre moitié de l'arête y est aussi.

Ce parcours en profondeur et droite d'abord permet également de donner une orientation sur chacune des arêtes. Pour les arêtes de l'arbre il s'agit de l'orientation racine vers feuille, pour les liens on pose que l'origine de l'arête est le nœud le plus profond dans l'arbre (celui où on a essayé de prendre l'arête mais où on a vu que l'on retombait sur nos pas).

8.3.2 Foliation des liens

On cherche ici une bijection entre des classes de cartes et des classes d'arbres enrichis. On peut ainsi utiliser la spécification des cartes pour énumérer et générer aléatoirement les arbres associés, ce qui sera fait en sections 8.4 et 8.5. En particulier, les λ-termes linéaires et les λ-termes affines sont des classes d'arbres enrichis.

En partant d'une carte M on lui associe une carte $\tilde{M}$ où l'on a ajouté un nouveau sommet sur chaque arête lien. Dans la carte $\tilde{M}$ à chaque fois que l'on crée 2 arêtes (à partie d'une) une appartient à son arbre couvrant en profondeur et droite d'abord et l'autre pas (voir Fig. 8.2). De cette manière, toute arête lien est précédée (dans l'ordre induit par le parcours en profondeur) d'un sommet de degré 2.

Soit $\mathcal{C}^\bullet_{R,D}$ la classe des cartes enracinées dont la racine a un degré inclus dans R et tous ses autres degrés inclus dans D, et $\mathcal{D}^\bullet_{R,D}$ la classe des cartes enracinées dont la racine a un degré inclus dans R et tous les sommets à l'origine des liens sont de degré 2 et tous les autres degrés appartiennent à D.

Soit $(\mathcal{C}^\bullet_{R,D})_{n,r}$ (resp. $(\mathcal{D}^\bullet_{R,D\cup\{2\}})_{m,r}$) l'ensemble des cartes de $\mathcal{C}^\bullet_{R,D}$ (resp. $\mathcal{D}^\bullet_{R,D}$) ayant n arêtes dont r liens (resp. m arêtes et r liens).

Définition 8.4. Soit $M = (E, \sigma, \alpha) \in \mathcal{C}^\bullet_{R,D}$, avec $|E| = 2n$. Soit T et R respectivement les arêtes de son arbre couvrant en profondeur et droite d'abord et ses liens avec $r = |R|$. On définit une nouvelle carte $\tilde{M} = (\tilde{E}, \tilde{\sigma}, \tilde{\alpha})$ comme suit :

- $\tilde{E} = E \cup F$, F est l'ensemble des nouvelles arêtes (définies juste après)
- Pour chaque arête $\{e, e'\} \in R$ avec $e' = \alpha(e)$,on définit deux demi-arêtes f et f', tel que $\tilde{\sigma}(f) = f'$, $\tilde{\sigma}(f') = f$, $\tilde{\alpha}(e) = f$, $\tilde{\alpha}(f) = e$, $\tilde{\alpha}(e') = f'$, $\tilde{\alpha}(f') = e'$.
- $\forall e \in E, \tilde{\sigma}(e) = \sigma(e)$.
- $\forall (e, e') \in T, \tilde{\alpha}(e) = \alpha(e) = e'$.

On a $\tilde{M} \in (\mathcal{D}^\bullet_{R,D\cup\{2\}})_{n+r,r}$.

La transformation de M à $\tilde{M}$, illustrée dans la figure 8.2n est appelée *foliation des liens*.

On remarque que l'arbre couvrant en profondeur et droite d'abord de $\tilde{M}$ est l'arbre couvrant en profondeur et droite d'abord de M où l'on a ajouté une arête menant à une nouvelle feuille à l'emplacement de départ des liens de M.

Lemme 8.5. *Pour tout n et r, la foliation des liens est une bijection de $(\mathcal{C}^{\bullet}_{R,D})_{n,r}$ à $(\mathcal{D}^{\bullet}_{R,D})_{n+r,r}$.*

Démonstration. La foliation des liens est inversible. Soit $\tilde{M} = (E \cup F, \tilde{\sigma}, \tilde{\alpha})$ Voici une description de la bijection inverse : les demi-arêtes de F sont les demi-arêtes autour des sommets de degré 2 qui précèdent les liens. En contractant ses demi-arêtes on retrouve M (*i.e.*, on peut reconstruire les permutations de manière parfaitement symétriques à celle de la foliation des liens). □

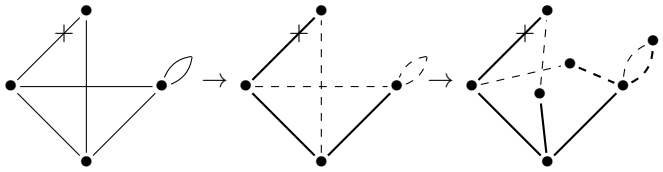


FIGURE 8.2 – De gauche à droite : une carte enracinée, sa partition en arêtes de l'arbre (en gras) et en liens (pointillés), et sa transformation par foliation des liens. La croix marque la racine.

8.3.3 Équation différentielle bivariée

Grâce à la foliation des feuilles, on peut se ramener à étudier uniquement le cas des cartes de $\mathcal{D}^{\bullet}_{R,D}$, la classe des cartes enracinées dont la racine a un degré inclus dans R et tous les sommets à l'origine des liens sont de degré 2 et tous les autres de degré appartenant à D.

On va ici donner une spécification bivariée de ces cartes. C'est une extension de l'équation bivariée des λ-termes affines et linéaires donnée en section 7.4. Pour cela on change ici la fonction taille : la taille d'un arbre enrichi est maintenant son nombre de nœuds (*i.e.*, sommets). Les deux modèles restent équivalents, en effet le nombre de noeuds d'une carte de $(\mathcal{D}^{\bullet}_{R,D})_{n+r,r}$ est $n+1$

On voit maintenant les cartes de $\mathcal{D}^{\bullet}_{R,D}$ comme des arbres enrichis : la structure d'arbre est celle donnée par l'arbre en profondeur et droite d'abord, enrichie de liens. Grâce à la foliation des liens, on sait maintenant que ces liens sont toujours entre des feuilles et un de leurs ancêtres dans l'arbre. Attention cependant : ici, on a un ordre cyclique autour de chaque sommet, ce qui correspond à la planarité de l'arbre, mais cet ordre concerne également les liens.

Comme pour le cas des λ-termes linéaire et affines, vus en section 7.4, on va marquer les feuilles par $\mathcal{U}$ pour pouvoir ensuite les lier par un opérateur différentiel.

Pour plus de simplicité de notation, dans la suite on considère $\mathcal{D}^{\bullet}_{R,D} = \mathcal{D}$ et on suppose R et D donnés.

Soit L_1 l'ensemble des mots de $(a+b)^*$ de longueur appartenant à R et commençant par un a (ou vide si $0 \in R$, ce qui correspond à autoriser la carte réduite à un sommet). Soit L_2 l'ensemble des mots de $(a+b)^*$ de longueur appartenant à $D-1 = \{x-1 | x \in D\}$

et commençant par un a (ou vide si $1 \in D$). Notons que comme une carte doit être connexe si $0 \in D$ on peut considérer $D \setminus 0$ sans que cela ne change rien.

Théorème 8.6. *Une spécification de $\mathcal{D}$ est $\mathcal{D}(\mathcal{Z}) = \mathcal{D}(\mathcal{Z},\mathcal{U})$ où*

$$\mathcal{D}(\mathcal{Z},\mathcal{U}) = \sum_{w\in L_1} \mathcal{Z} \times \mathcal{E}^w \tag{8.4}$$

$$\mathcal{E}(\mathcal{Z},\mathcal{U}) = \mathcal{Z}\mathcal{U} + \sum_{w\in L_2} \mathcal{Z} \times \mathcal{E}^w \tag{8.5}$$

$$\text{où } \mathcal{E}^w = \begin{cases} 1 & \text{si } w = \varepsilon\,; \\ \mathcal{E}(\mathcal{Z},\mathcal{U}) \times \mathcal{D}^{w'} & \text{si } w = w'a\,; \\ \frac{\partial \mathcal{D}^{w'}}{\partial \mathcal{U}}(\mathcal{Z},\mathcal{U}) & \text{si } w = w'b. \end{cases}$$

Comme d'habitude, ces équations formelles sur les classes combinatoires se traduisent en équations (ici différentielles) sur la série génératrice. Dans le cas général ces équations n'ont a priori pas de solution simple.

Démonstration. Soit M une carte de $\mathcal{D}$ et T son arbre couvrant en profondeur et droite d'abord. La première (dans l'ordre cyclique, après l'arête père) arête a d'un nœud interne n de T est toujours dans T. En effet, lorsque l'on visite le nœud n pour la première fois dans le parcours en profondeur, aucun des nœuds des sous-arbres de n n'a encore été visité. Comme un lien lie une feuille à un de ses ancêtres la première arête ne peut pas être un lien car n n'est pas une feuille et sinon il devrait lier une feuille d'un sous-arbre de n, donc encore non visité au moment où n est visité, et dans ce cas on ajoute forcément a à T. Les autres arêtes peuvent être de n'importe quel type (soit elle lie un nœud non encore visité, soit une feuille d'un des sous-arbres précédents dans a). Si une arête mène vers un nouveau sous-arbre, cela correspond à multiplier par $\mathcal{D}$ dans la spécification. Si une arête lie une feuille d'un des sous-arbres précédents, cela correspond à dériver les sous arbres précédant par $\mathcal{U}$ (une feuille ne peut être liée qu'une seule fois).

□

Exemple. Le tableau de la figure 8.3 montre les différents types de nœuds possible pour la classe $\mathcal{D}_4 = \mathcal{D}_{\{4\},\{5\}}$ où chaque nœud interne a 4 arêtes sortantes. L'équation 8.4 donne ici :

$$\begin{aligned}\mathcal{D}_4 &= \mathcal{Z}\Big(\mathcal{U} + \mathcal{D}_4^4 + \frac{\partial \mathcal{D}_4^3}{\partial \mathcal{U}} + \frac{\partial \mathcal{D}_4^2}{\partial \mathcal{U}}\mathcal{D}_4 + \frac{\partial \mathcal{D}_4}{\partial \mathcal{U}}\mathcal{D}_4^2 + \frac{\partial^2 \mathcal{D}_4^2}{\partial \mathcal{U}^2} + \frac{\partial \frac{\partial \mathcal{D}_4}{\partial \mathcal{U}}\mathcal{D}_4}{\partial \mathcal{U}} + \frac{\partial^2 \mathcal{D}_4}{\partial \mathcal{U}^2}\mathcal{D}_4 + \frac{\partial^3 \mathcal{D}_4}{\partial \mathcal{U}^3}\Big) \\ &= \mathcal{Z}\Big(\mathcal{U} + \mathcal{D}_4^4 + 6\mathcal{D}_4^2\frac{\partial \mathcal{D}_4}{\partial \mathcal{U}} + 4\mathcal{D}_4\frac{\partial^2 \mathcal{D}_4}{\partial \mathcal{U}^2} + 3\Big(\frac{\partial \mathcal{D}_4}{\partial \mathcal{U}}\Big)^2 + \frac{\partial^3 \mathcal{D}_4}{\partial \mathcal{U}^3}\Big)\end{aligned}$$

8.4 Asymptotique des λ-termes linéaires et affines

8.4.1 Énumération asymptotique des λ-termes linéaires

D'après la section précédente et la section 7.4, on retrouve la spécification bivariée des λ-termes linéaires avec l'équation (8.4) pour $R = \{2\}$ et $D = \{3\}$:

Nœud à la racine	Terme associé	Définition
•	$\mathcal{Z}\mathcal{U}$	Feuille libre.
	$\mathcal{D}_4^4$	Nœud ayant 4 fils dans T.
	$\mathcal{Z}\frac{\partial \mathcal{D}_4^3}{\partial \mathcal{U}}$	Nœud ternaire ayant un lien en 4-ème position, liant une feuille d'un quelconque de ses fils.
	$\mathcal{Z}\frac{\partial \mathcal{D}_4^2}{\partial \mathcal{U}}\mathcal{D}_4$	Nœud ternaire ayant un lien en 3-ème position liant une feuille d'un de ses deux premiers fils.
	$\mathcal{Z}\frac{\partial \mathcal{D}_4}{\partial \mathcal{U}}\mathcal{D}_4^2$	Nœud ternaire ayant un lien en 4-ème position, liant une feuille de son premier fils.
	$\mathcal{Z}\frac{\partial^2 \mathcal{D}_4^2}{\partial \mathcal{U}^2}$	Nœud binaire ayant 2 liens en 3-ème et 4-ème positions liants chacun un feuille quelconque de ses fils.
	$\mathcal{Z}\frac{\partial(\frac{\partial \mathcal{D}_4}{\partial \mathcal{U}}\mathcal{D}_4)}{\partial \mathcal{U}}$	Nœud binaire ayant 2 liens en 2-ème et 4-ème positions liants 2 feuilles, resp. dans le premier fils et dans un fils quelconque.
	$\mathcal{Z}\frac{\partial^2 \mathcal{D}_4}{\partial \mathcal{U}^2}\mathcal{D}_4$	Nœud binaire ayant 2 liens en 2-ème et 3-ème positions liants chacun une feuille de son premier fils.
	$\mathcal{Z}\frac{\partial^3 \mathcal{D}_4}{\partial \mathcal{U}^3}$	Noeud unaire ayant trois liens liants des feuilles de son fils.

FIGURE 8.3 – Les différents types de nœuds possibles pour un arbre de $\mathcal{D}_{\{4\},\{5\}}$

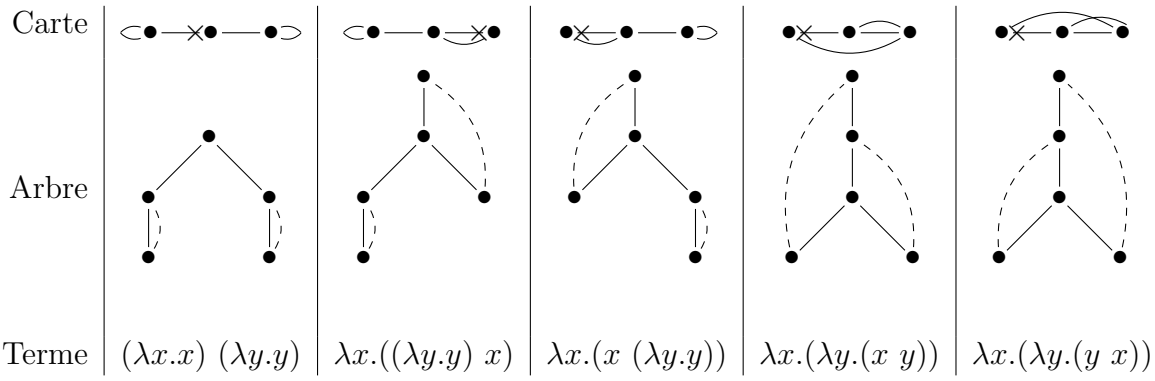


FIGURE 8.4 – Les cartes pointées de taille 8 de $\mathcal{C}^{\bullet}_{2,3}$ et les λ-termes linéaires correspondants de taille 5

- les nœuds binaires ont un père, un fils gauche et un fils droit (sauf la racine). On a bien un sommet de degré 3 avec un ordre induit par l'orientation des arêtes de l'arbre.
- les nœuds unaires ont un fils un père et un lien (sauf la racine). La position du lien n'est pas déterminée, on peut donc fixer canoniquement qu'ils arrivent après l'arête fils. On a bien un sommet de degré 3 avec un ordre induit par l'orientation des arêtes de l'arbre.
- la racine est soit un nœud unaire à un fils et un lien, soit est un nœud binaire est a deux fils. Dans les deux cas le sommet est de degré deux.

(les nœuds unaires ont un fils et un lien, les nœuds binaires deux fils donc tous ont degré 3 sauf la racine).

On en déduit directement le théorème suivant :

Théorème 8.7. *L'ensemble des λ-termes linéaires de taille $3k+2$ est en bijection avec la classe des cartes enracinées non-étiquetées avec degré de la racine 2 et tout les autres sommets de degré 3 de taille $6k+2$.*

Pour déterminer le comportement asymptotique de λ-termes linéaires, on va se baser sur l'équation symbolique donnée par la vision carte (équation 8.3) pour obtenir le théorème suivant :

Théorème 8.8. *On a :*

$$C^{\bullet}_{2,3}(x) := x^3 \cdot \frac{d}{dx} \ln(e^{x^3/3} \odot e^{x^2/2}).$$

où $\mathcal{C}^{\bullet}_{2-3}$ est la classe des cartes enracinée dont la racine est de degré 2 et tous les autres de degré 3.

Démonstration. Soit $\mathcal{C}_3$ la carte des cartes (connexes) dont tous les sommets ont degré 3. Par la spécification (8.1), on obtient :

$$e^{C_3(x)} = e^{x^3/3} \odot e^{x^2/2},$$

ce qui donne $C_3(x) = \ln(e^{x^3/3} \odot e^{x^2/2})$. On peut ensuite pointer (dériver puis multiplier par x) cette équation pour obtenir les cartes pointées, ce qui donne :

$$C_3^\bullet(x) = x\frac{d}{dx}\ln(e^{x^3/3} \odot e^{x^2/2}).$$

On remarque ensuite que l'opération d'ajouter une racine de degré 2 à coté de la demi-arête pointée est une opération bijective qui rajoute 2 demi-arêtes. On obtient ainsi le théorème. □

Des calculs élémentaires donnent :

$$e^{x^3/3} \odot e^{x^2/2} = \sum_{n\geq 0} \frac{(6n)!}{(3n)!(2n)!2^{3n}3^{2n}} x^{6n}.$$

On applique ensuite la formule de Stirling, ce qui donne :

$$[x^{6n}](e^{x^3/3} \odot e^{x^2/2}) \sim \frac{(6n/e)^n}{\sqrt{2\pi n}}.$$

Lemme 8.9. *On a*

$$[x^{6n}](e^{x^3/3} \odot e^{x^2/2}) \sim [x^{6n}]\ln(e^{x^3/3} \odot e^{x^2/2}).$$

Démonstration. Le point central est le théorème suivant, dû à Bender [Ben74] :
Soient $A(x) = \sum_{n>0} a_n x^n$ *et* $F(x,y) = \sum_{i,k} f_{i,k} x^i y^k$ *satisfaisant :*

i $a_{n-1} = o(a_n)$;

ii il existe $r > 0$ *tel que* $\sum_{k=r}^{n-r} |a_k a_{n-k}| = O(a_{n-r})$;

iii $F(x,y)$ *est analytique en* $(0,0)$.

Définissons les deux séries $B(x) = F(x, A(x))$ *et* $D(x) = F_y(x, A(x))$ *où* $F_y(x,y) = \frac{\partial}{\partial y}F(x,y)$. *Alors,*

$$[x^n]B(x) = \sum_{k=0}^{r-1} [x^k]D(x)a_{n-k} + O(a_{n-r}).$$

Dans notre cas, on choisit $F(x,y) = \ln(1+y)$ (remarquons qu'il n'est pas nécessaire d'avoir une série bivariée) et $A(x) = e^{x^3/3} \odot e^{x^2/2} - 1$. Les conditions i et iii sont facilement vérifiées. Pour prouver ii, on observe que pour $r = 6$, $\sum_{k=6}^{n-6} |a_k a_{n-k}| = O(a_{n-6})$ à cause de la croissance surexponentielle des coefficients a_{6n}. Il suffit ensuite d'appliquer le théorème de Bender pour obtenir le lemme. □

Théorème 8.10. *Pour* $n = 2 \mod 3$, *le nombre de λ-termes linéaires de taille n suit l'équivalent asymptotique quand n tend vers l'infini :*

$$BCI_n \sim \frac{\sqrt[3]{2}\sqrt{6}}{2n^{1/6}\sqrt{\pi}} \left(\frac{2n}{e}\right)^{n/3}.$$

Démonstration. On sait que les coefficients $[x^n]T(x)$ sont non nuls uniquement pour $n = 2 \mod 3$, et que

$$[x^{3k+2}]T(x) = [x^{6k}]C(x) \sim \frac{6\sqrt{k}(6k/e)^k}{\sqrt{2\pi}},$$

où $T(x)$ est la série génératrice ordinaire de la classe des λ-termes linéaires, ayant pour fonction taille le nombre de nœuds de leur arbre syntaxique, *i.e.*, la longueur des λ-termes. Puis, depuis ce qui précède, le comportement asymptotique des coefficients est évident. □

8.4.2 Énumération asymptotique des λ-termes affines

Comme pour le cas des λ-termes linéaires, on peut trouver une spécification alternative pour les λ-termes affines à partir des cartes dont la racine a son degré dans $R = \{1, 2\}$ et tous les autres sommets dans $D = \{2, 3\}$. Un λ-terme affine de taille $k + 2$ correspond à une carte de taille $2k$. Pour cela, on fixe également que les liens arrivent canoniquement après le fils d'un nœud unaire.
On peut également, pour des raisons pratiques dans la suite, considérer d'un λ-terme affine est le produit cartésien d'une séquence de nœuds unaires initiaux qui ne lient aucune feuille, et d'un λ-terme affine dont la racine est de degré 2. Autrement dit, la classe des λ-termes affines est en bijection avec le produit cartésien d'une séquence d'atomes et d'une carte enracinée dont la racine a degré 2 et tous les autres nœuds degrés 2 ou 3.

Comme dans la section précédente, on retrouve l'équation suivante pour $Y(x)$, la série génératrice des cartes enracinées dont les degrés sont tous 2 ou 3 :

$$Y(x) := x \cdot \frac{d}{dx} \ln(e^{x^3/3+x^2/2} \odot e^{x^2/2}).$$

Il faut maintenant chercher l'équivalent asymptotique des coefficients de la série $e^{x^2/2+x^3/3}$. Cette question rentre dans le cadre de la méthode du point col [FS09a, Partie 8]. Par des calculs fastidieux mais classiques on obtient :

$$[x^n]e^{x^2/2+x^3/3} \sim \frac{1}{\sqrt{6\pi\, n}} \left(\frac{e}{n}\right)^{1/3n} e^{1/18-1/6\, n^{1/3}+1/2\, n^{2/3}}.$$

Puis, on peut de nouveau appliquer le théorème de Bender d'où ressort :

$$[x^n] \ln(e^{x^2/2+x^3/3} \odot e^{x^2/2}) \sim n^{n/6} \frac{1}{\sqrt{3\pi\, n}} e^{1/18-1/6\, n+1/2\, n^{2/3}-1/6\, n^{1/3}}.$$

La bijection avec le produit cartésien d'une séquence d'atome et d'une carte enracinée dont la racine a degré 2 et tous les autres nœuds degrés 2 ou 3 fournit l'OGF $S(x)$ pour la classe des λ-termes affines. Cette série génératrice vérifie :

$$S(x) = \frac{1}{1-x}(x^2 + x^2 Y(\sqrt{x}))$$

où $Y(x) := x \cdot \frac{d}{dx} \ln(e^{x^3/3+x^2/2} \odot e^{x^2/2})$.

Par une analyse de singularité (le pôle à $x = 1$ de $1/(1 - x)$), on obtient :

Théorème 8.11. *Le nombre de λ-termes affines de taille n admet l'équivalent asymptotique suivant quand n tend vers l'infini :*

$$BCK_n \sim [x^n]S(x) \sim \frac{e^{1/18}\sqrt[3]{2}}{\sqrt{6\pi}\, n^{1/6}} \left(\frac{2n}{e}\right)^{n/3} e^{1/2(2n)^{2/3}-1/6(2n)^{1/3}}.$$

Corollaire 8.12. *Le ratio du nombre l_n de λ-terme linéaires de taille n par rapport au nombre a_n de λ-termes affines de taille n a une croissance sous-exponentielle : pour $n = 2 \mod 3$,*

$$\frac{l_n}{a_n} \sim 3\, e^{-2^{-1/3}n^{2/3}+1/6\ \sqrt[3]{2}n^{1/3}-1/18}.$$

Démonstration. Le résultat découle directement des théorème 8.10 et 8.11 pour $n = 3k + 2$. □

8.5 Génération aléatoire

8.5.1 Générateur de λ-termes linéaires

On peut obtenir un générateur aléatoire de λ-termes linéaires directement depuis la spécification :

$$\Big(\mathcal{Z}^2 \times \textsc{Set}(\textsc{Cyc}_3(\mathcal{Z}))\Big) \odot \textsc{Set}(\textsc{Cyc}_2(\mathcal{Z})) = \textsc{Set}^\bullet(\mathcal{C})).$$

Plus précisément, il suffit de générer deux permutations de même taille, une pointée dont tous les cycles sont de longueur 3, sauf le cycle pointé de longueur 2, et l'autre dont tous les cycles sont de longueur 2. On en déduit la carte sous-jacente si les permutations engendrent un sous-groupe transitif du groupe de permutations, puis on y applique la foliation des liens pour obtenir un λ-terme linéaire. La dernière étape étant bijective, pour obtenir un λ-terme linéaire uniforme de taille $3k + 2$ il suffit de générer uniformément deux permutations (de longueur de cycles 3 et 2) engendrant un sous-groupe transitif du groupe de permutations.

Une manière simple d'obtenir uniformément une permutation de longueur ℓk dont tous les cycles ont longueur ℓ est de générer aléatoirement une permutation quelconque écrite en ligne (par exemple le Knuth shuffle décrit dans la section 3.2.1), puis de regrouper les termes par ℓ dans l'ordre de lecture pour créer les cycles.

Par exemple, tirer la permutation quelconque 296734185 donne l'ensemble de 3-cycles (296)(734)(185).

Démonstration. Pour tirer une structure étiquetée de taille n, on peut lancer le générateur de Boltzmann pour obtenir la structure sous jacente, puis étiqueter les atomes. Il existe une seule structure de taille ℓk dans $\textsc{Set}(\textsc{Cyc}(\mathcal{Z}))$: celle qui regroupe les atomes par groupes de ℓ. □

De même, pour obtenir une permutation pointée composée d'un couple puis d'un ensemble de 3-cycles, il suffit de lire les 2 premiers nombres comme un couple puis le reste de la permutation comme un ensemble de 3 cycles.

Algorithm 16: $\Gamma_{lineaire}$

Input: Un entier n.
Output: Un λ-terme linéaire de taille $3n + 2$.

```
Tirer deux permutations σ et α de taille 6n + 2.
Transformer σ en produit d'un couple et d'une permutation dont tous les cycles
  ont longueur 3. Le premier nombre de σ est la demi-arête pointée.
Transformer α en involution sans point fixe.
if (α, β) engendre un sous-groupe transitif then
    Transformer la carte en λ-terme par foliation des liens.
    Oublier les étiquettes.
    return le λ-terme linéaire
else
    Relancer Γ_lineaire.
```

Exemple. Un λ-terme linéaire de taille 5.

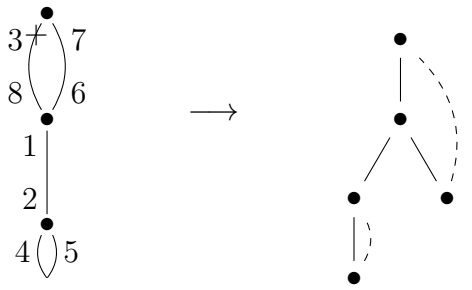


FIGURE 8.5 – Une manière de tirer le λ-terme $\lambda x.((\lambda y.y)x)$

Tirer les permutations $(3, 7, 1, 6, 8, 4, 2, 5)$ et $(8, 3, 6, 7, 4, 5, 1, 2)$ donne :
- $\sigma = (3, 7, (1, 8, 6)(4, 2, 5))$ les demi-arêtes 3 et 7 n'appartenant à aucun cycle correspondent au sommet la racine, 3 étant la demi-arête pointée,
- $\alpha = ((8, 3)(6, 7)(4, 5)(1, 2))$.

Ce qui donne l'élément de $\mathcal{D}$ dans la Figure 1.

Proposition 8.13. *L'algorithme 16 est un générateur aléatoire uniforme de λ-termes linéaires, de complexité moyenne linéaire en la taille du λ-terme.*

Démonstration. L'uniformité découle directement de la rigidité des cartes, et de la bijection avec les λ-termes.

La génération des permutations se fait en temps linéaire, ainsi que le parcours en profondeur (et donc la transformation via la bijection). Par le lemme 8.9, on sait que la carte est connexe avec une probabilité tendant vers 1 quand la taille tend vers

l'infini, donc le nombre moyen de rejets est constant. L'algorithme est donc linéaire en moyenne. □

Remarque. Si l'on utilise la composante pointée plutôt que de rejeter quand la carte n'est pas connexe, on obtient un générateur uniforme en taille approchée.

8.5.2 Générateur aléatoire d'arbres enrichis

Soient $\mathcal{D}$ la classe des cartes enracinées dont la racine a son degré dans R et tous les autres sommets un degré compris dans D. Comme les degrés des nœuds ne sont plus fixés, on ne peut pas faire aussi simple que dans le cas des λ-termes linéaires : en effet, on ne peut pas regrouper les nombre d'une permutation aléatoire uniforme en cycles de longueur déterminée.

On va donc utiliser des générateurs de Boltzmann pour générer un objet σ de

$$\textsc{Seq}_R \times \textsc{Set}(\textsc{Cyc}_D\, \mathcal{Z}).$$

On obtient ainsi une permutation pointée vérifiant les propriétés souhaitées, en taille approchée n. Si n est impair on rejette, sinon on génère une involution sans point fixe de taille n, que l'on combine avec σ pour obtenir une carte de $\mathcal{D}$. Il ne reste qu'à appliquer la bijection et à oublier les étiquettes pour obtenir un arbre enrichi dont la racine a degré dans R et tous les autres nœuds internes dans D.

Algorithm 17: $\Gamma_{arbreenrichi}$

Input: Un paramètre x, une taille minimale n_m, une taille maximale n_M, l'ensemble R des degrés autorisés pour la racine et D pour les autres sommets.

Output: Un arbre enrichi

```
Tirer un objet σ dans Seq_R(Z) × Set(Cyc_D(Z)) par un générateur de
Boltzmann de paramètre x.
if |σ| ≡ 1 mod 2 or |σ| < n_m or |σ| > n_M then
    Restart Γ_arbreenrichi
else
    Tirer un permutation α de taille |σ|.
Transformer σ en permutation sans point fixe.
if la carte ({1, ..., |α|}, α, β) est connexe then
    Transformer la carte en arbre par la bijection
    Oublier les étiquettes.
    return l'arbre.
else
    Recommencer Γ_arbreenrichi.
```

Théorème 8.14. *L'algorithme 17 retourne un arbre de $\mathcal{D}$ de taille appartenant à $[(1-\varepsilon)n, (1+\varepsilon)n]$, pour $\varepsilon > 0$ donné, en temps linéaire en n en moyenne. Tous les arbres de même taille sont tirés avec même probabilité.*

Démonstration. La probabilité qu'un générateur de Boltzmann de paramètre x tire un objet de taille paire dans $\textsc{Seq}_R(\mathcal{Z}) \times \textsc{Set}(\textsc{Cyc}_D(\mathcal{Z}))$ est

$$\gamma(x) = \frac{1}{2}(1 + \frac{f(-x)}{f(x)}),$$

où

$$f(x) = \sum_{k\in R} x^k \cdot \exp(\sum_{k\in D} x^k).$$

Quand x tend vers $+\infty$, $\gamma(x)$ admet une limite dans $[1/2, 1]$. L'espérance du nombre de rejet dû à la parité est donc constant. Après cette phase de rejet, le tirage d'un objet dans $\textsc{Seq}_R(\mathcal{Z}) \times \textsc{Set}(\textsc{Cyc}_D(\mathcal{Z}))$ ayant une taille dans $[(1-\varepsilon)n, (1+\varepsilon)n]$ a une complexité linéaire en moyenne (ce résultat est prouvé dans [DFLS04]). La dernière phase de rejet pour la connexité se fait également en temps constant en moyenne, par le théorème de Bender.

□

Proposition 8.15. *En particulier, l'algorithme 17 appelé avec $R = \{1, 2\}$ et $D = \{2, 3\}$ donne un générateur aléatoire uniforme en taille approché pour les λ-termes affines.*

8.6 Perspectives

Cette bijection ne peut a priori pas s'étendre facilement à d'autres classes de λ-termes. Cependant, elle met en évidence un lien entre des équations de Ricatti et d'autres équations holonomes. Notre perspective principale, en plus des questions liées au λ-termes, est donc d'étudier les équations de Ricatti pour lesquels ont peut obtenir l'asymptotique par des méthodes similaires à celles développées ici : définir une équation holonome proche et en déduire le comportement asymptotique par des méthodes classiques de combinatoire analytique puis utiliser le théorème de Bender pour en déduire l'asymptotique de la solution de l'équation de Ricatti.

Chapitre 9

Spécification monovariée des λ-termes

9.1 Introduction

Le premier but est ici d'obtenir des spécifications monovariées pour différentes classes de λ-termes clos. Les spécifications que l'on obtient font intervenir des opérateurs de branchements, des opérateurs différentiels (pointage) et des substitutions. On y retrouve un principe de greffe similaire à celui de la partie II. Si on peut traduire automatiquement ces spécifications en séries, les outils classiques d'analyse asymptotique ne suffisent pas à obtenir des équivalents pour les λ-termes généraux, la classe n'étant pas holonome.

Plus précisément, nous nous intéressons aux classes que nous appelleront $BCI(p)$ et $BCK(p)$: la classe des $BCI(p)$ (*resp.* $BCK(p)$) est la classe des λ-termes où chaque nœud unaire lie exactement (*resp.* au plus) p feuilles. Pour $p = 1$, on retrouve les λ-termes linéaires et affines.

Nous commencerons, en section 9.3, par préciser nos définitions et notations, ainsi que par quelques remarques immédiates. Dans la section 9.3 nous donnerons des spécifications monovariées des $BCI(p)$, $BCK(p)$ et des λ-termes généraux, et décrirons les bijections associées. Puis, dans la section 9.4, nous en déduirons l'asymptotique du nombre de termes $BCI(p)$ de taille donnée. La section 9.5 est dédiée à obtenir des bornes supérieure et inférieure sur le nombre λ_n de λ-termes de taille n, permettant d'avoir le terme principal de l'asymptotique de $\log \lambda_n$. De plus, nous déduisons une relation de récurrence permettant le calcul efficace de λ_n.

9.2 Notations et faits élémentaires

Définition 9.1. Les classes des termes $BCI(p)$ et des termes $BCK(p)$ sont définies ainsi :

- un λ-terme est $BCI(p)$ si chaque nœud unaire a exactement p pointeurs, c'est-à-dire lie exactement p feuilles. Ces termes sont également appelés *λ-termes p-linéaires*.

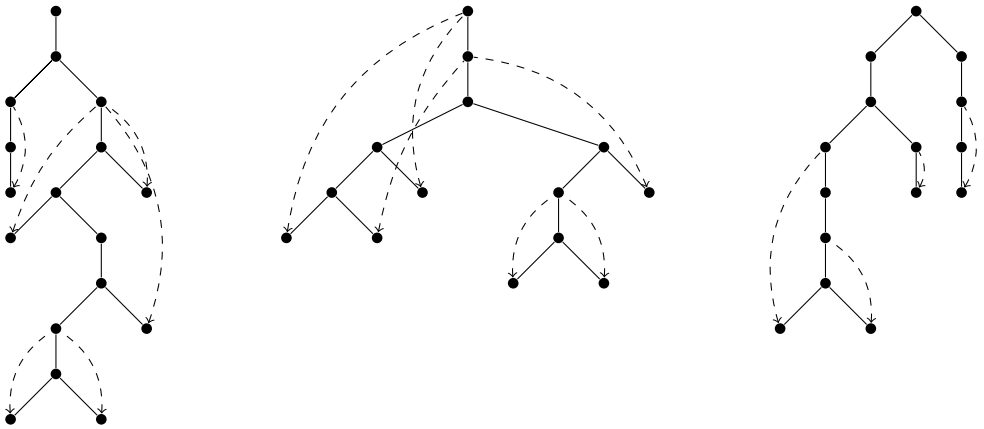


FIGURE 9.1 – De gauche à droite : un λ-terme clos de taille 17, un $BCI(2)$ de taille 14 et un $BCK(1)$ de taille 15.

– un λ-terme est $BCK(p)$ si chaque nœud unaire lie au plus p feuilles. Ces termes sont également appelés *λ-termes p-affines*.

La taille d'un terme $BCI(p)$ ou d'un terme $BCK(p)$ est son nombre total de nœuds (binaires, unaires et feuilles).

Fait 4. Les plus petits termes de $BCI(p)$ ont un nœud unaire à la racine et p feuilles. Il y a p liens liant la racine à chacune des feuilles. On voit immédiatement que si on retire la racine et tous ses pointeurs, on a un arbre de Catalan à p feuilles. La taille d'un tel terme est donc $2p$, et leur nombre

$$C_{p-1} = \binom{2p-2}{p-1}\frac{1}{p}.$$

Fait 5. Un terme $BCI(p)$ à j nœuds unaires a pj feuilles et $pj-1$ nœuds binaires, sa taille est donc $(2p+1)j-1$.

La classe des $BCI(p)$ potentiellement non-clos, admet la spécification bivariée

$$\mathcal{T}(\mathcal{Z},\mathcal{U}) = \mathcal{Z}\mathcal{U} + (\mathcal{Z}\times\mathcal{T}(\mathcal{Z},\mathcal{U})\times\mathcal{T}(\mathcal{Z},\mathcal{U})) + (\mathcal{Z}\times\frac{\partial^p\mathcal{T}(\mathcal{Z},\mathcal{U})}{\partial\mathcal{U}^p}), \qquad (9.1)$$

où $\mathcal{T}(\mathcal{Z},\mathcal{U})$ est la classe des $BCI(p)$ non-clos (*i.e.*, pouvant contenir de feuilles libres) comptés selon leur taille et leur nombre de feuilles libres. On retrouve ici une description bivariée similaire à celle des λ-termes linéaires présentée en section 7.3. Les $BCI(p)$ correspondent ensuite à $\mathcal{T}(\mathcal{Z},0)$.
On peut obtenir une spécification similaire pour les $BCK(p)$, mais il faudra sommer sur tous les ordres de dérivation possibles.

Dans la suite, nous définissons de nouvelles spécifications pour les $BCI(p)$, les λ-termes clos et les $BCK(p)$. Ces spécifications reposeront sur un principe de greffe.

9.3 Spécifications de différentes classes de λ-termes

Nous utilisons ici les séries génératrices comme moyen d'énumération des λ-termes.

Soit $g_n = g_n^{(p)}$ le nombre de λ-termes p-linéaires de taille n et $G_p(z)$ la série génératrice correspondante. Par le fait 5, on a :

$$G_p(z) = \sum_{j\geq 1} g_{j(2p+1)-1} z^{j(2p+1)-1}.$$

De manière similaire, on définit

$$F_p(z) = \sum_{n\geq 1} f_n z^n$$

et

$$\Lambda(z) = \sum_{n\geq 1} \lambda_n z^n$$

où $f_n = f_n^{(p)}$ est le nombre de termes $BCK(p)$ de taille n et λ_n le nombre de λ-termes clos de taille n.

L'étape suivante est d'établir des équations fonctionnelles pour ces séries génératrices. Ceci peut être fait en donnant une spécification des classes impliquées par la méthode symbolique. Nous allons commencer par traiter le cas des $BCI(p)$.

9.3.1 λ-termes multi-linéaires

Nous avons montré dans le chapitre précédent que $G_1(z)$ satisfait l'équation

$$G_1(z) = z^2 + zG_1(z)^2 + \Delta_1 G_1(z),$$

où l'opérateur différentiel Δ_1 est $2z^4 D$ et D symbolise l'opérateur différentiel ordinaire par rapport à z.

Proposition 1. La série génératrice des termes $BCI(p)$ satisfait l'équation différentielle

$$G_p(z) = C_{p-1} z^{2p} + zG_p(z)^2 + \Delta_p G_p(z) \tag{9.2}$$

où

$$\Delta_p = \sum_{l=1}^{p} \frac{\alpha_{l,p}}{l!} z^{l+2p+1} D^l \tag{9.3}$$

où les constantes $\alpha_{l,p}$ sont définies par

$$\alpha_{l,p} = \sum_{\sum_i s_i = l;\ \sum_i i s_i = p} \binom{l}{s_1, \ldots, s_p} \prod_{m=1}^{p} \binom{2m}{m}^{s_m}. \tag{9.4}$$

Démonstration. On peut spécifier la classe des $BCI(p)$, notée $\mathcal{T}$, par :

$$\mathcal{T} = \mathcal{S} + (\mathcal{Z} \times \mathcal{T} \times \mathcal{T}) + (\mathcal{Z} \times \widetilde{\mathcal{T}}), \tag{9.5}$$

où $\mathcal{S}$ est l'ensemble des plus petits termes $BCI(p)$ (*cf.* fait 4) et $\widetilde{\mathcal{T}}$ la classe des $BCI(p)$ non-clos ayant exactement p variables libres (et on considère que le produit cartésien lie ces variables). Cette spécification peut être vue comme la définition suivante. Un terme $BCI(p)$ est soit :

– un des plus petits termes,
– un nœud binaire dont les deux fils sont des $BCI(p)$,
– ou un nœud unaire suivi d'un terme avec exactement p feuilles libres : on lie alors ces feuilles au nœud unaire.

Pour pouvoir complètement spécifier les $BCI(p)$, on doit définir une manière de construire $\widetilde{\mathcal{T}}$. Pour construire un terme $\tilde{t}$ de $\mathcal{Z} \times \widetilde{\mathcal{T}}$ (que l'on appellera un $BCI(p)$ à p variables libres), on part d'un terme $BCI(p)$ auquel on greffe p feuilles, par une méthode rappelant celle de Rémy (*cf.* Partie II). Pour cela, on choisit t un $BCI(p)$ et p nœuds, pas nécessairement distincts, de t. A chaque nœud v on fait correspondre son arête mère, *i.e.*, l'arête reliant v à son père si v n'est pas la racine de t, et v au nœud unaire (de $\tilde{t} \in \mathcal{Z} \times \widetilde{\mathcal{T}}$) nouvellement créé sinon. Ainsi, le choix de p nœuds se traduit comme le choix de p arêtes, où une même arête peut être sélectionnée plusieurs fois. Supposons maintenant que ℓ arêtes distinctes soient sélectionnées, et que pour tout i il y ait s_i arêtes qui ont été sélectionnées i fois.

Si une arête est sélectionnée i fois, on la remplace par un chemin où un arbre binaire est attaché à chaque nœud, soit à gauche soit à droite du chemin, et le nombre total de feuilles de ces arbres binaires est i (voir la figure 9.2 pour une illustration de ce processus). Ce remplacement crée de nouveaux nœuds internes. Au total, les différents remplacements créent donc

$$\sum_{i=1}^{p} i s_i = p$$

nouvelles feuilles et p nouveaux nœuds internes dans t. On en déduit donc que $\tilde{t}$ a exactement $2p+1$ nœuds de plus que t et, clairement, $\tilde{t}$ est un $BCI(p)$.

Inversement, si on a un terme t $BCI(p)$ dont la racine est unaire, alors la retirer (ainsi que ses liens) donne un terme à p feuilles libres. Ces feuilles sont forcément filles de nœuds binaires (puisque chaque nœud unaire est lié à p feuilles de son sous-arbre, avec $p > 0$). Ainsi, les feuilles libres induisent un ensemble de sous-arbres binaires de t maximaux dont toutes les feuilles sont libres.

Comptons maintenant le nombre de façons dont cette opération peut être faite. Chaque arête sélectionnée i fois est en fait remplacée par un séquence d'arbres binaires à droite ou à gauche. La série génératrice associée aux arbres binaires est

$$T(u) = \sum_{n \geq 1} C_{n-1} u^n = \frac{1 - \sqrt{1-4u}}{2}.$$

Puis, le nombre de telles séquences comportant exactement p feuilles est

$$[u^i] \frac{1}{1 - 2T(u)} = [u^i] \frac{1}{\sqrt{1-4u}} = \binom{2i}{i}.$$

Remarquons maintenant que s_i des ℓ arêtes sélectionnées le sont i fois, pour $i = 1, \ldots, p$, le nombre de façons de partitionner les ℓ arêtes en respectant les multiplicités de sélection est

$$\binom{\ell}{s_1, \ldots, s_p}.$$

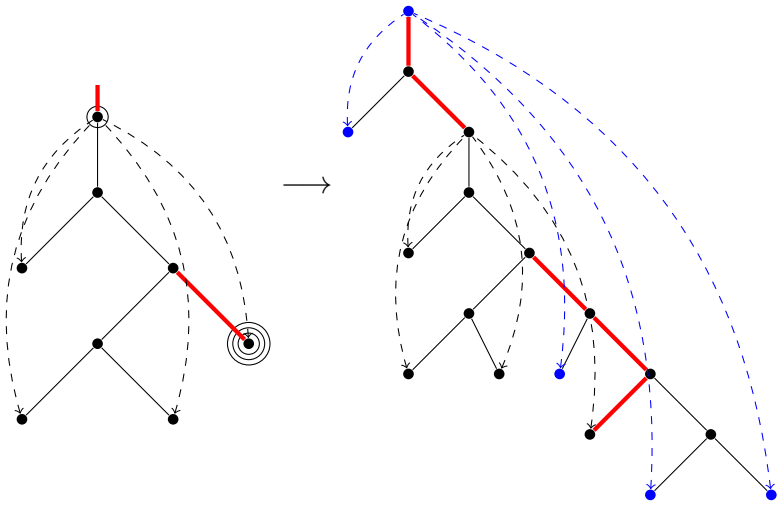


FIGURE 9.2 – A gauche un $BCI(4)$ avec un nœud sélectionné une fois et un autre trois fois : la sélection est représenté par un cercle autour du nœud. Les arêtes sélectionnées correspondantes apparaissent en rouge.

À droite, un des $BCI(4)$ que l'on peut obtenir à partir du terme de gauche. Chaque arête en rouge a été remplacée par un chemin en rouge auquel des nœuds binaires ont été attachés. L'arête racine (en haut du premier terme, qui avait été sélectionnée une fois) a été remplacée par un chemin de longueur deux (ayant donc 3 nœuds) et un arbre de taille 1 à été attaché à gauche du nœud du milieu (notons qu'on ne peut pas attacher d'arbre aux autres nœuds, qui ne sont pas créés par ce remplacement). L'autre arête rouge du terme de gauche a été remplacée par un chemin de longueur 3, auquel 2 arbres sont attachés : un arbre de taille 1 à gauche du nœud nouvellement créé le plus haut et un arbre de taille 3 à droite sur l'autre nœud. Toutes les feuilles nouvellement créées (en bleu) sont liées à la nouvelle racine unaire (en bleu également).

Chacune des s_i arêtes sélectionnée i fois est ensuite remplacée par une des $\binom{2i}{i}$ séquences d'arbres binaires possibles. Il a a donc

$$\prod_{i=1}^{p} \binom{2i}{i}^{s_i}$$

façons de faire le remplacement complet. Enfin, choisir ℓ arêtes distinctes correspond à appliquer l'opérateur $z^l D^l/l!$ au niveau des séries génératrices et les $2p+1$ nœuds nouvellement créés entrainent un facteur z^{2p+1}. □

Proposition 2. Soient $F(z)$ une série formelle (à coefficients réels), $D_u = \partial/\partial u$, la dérivée formelle, et U l'opérateur $G(u) \mapsto G(0)$, $G(u)$ étant une série formelle. Alors

$$\Delta_p F(z) = \frac{z^{2p+1}}{p!} U D_u^p F\left(\frac{z}{\sqrt{1-4u}}\right) = z^{2p+1}[u^p] F\left(\frac{z}{\sqrt{1-4u}}\right).$$

Démonstration. La seconde équation est évidente puisque $UD_u^p/p! = [u^p]$ correspond exactement au théorème de Taylor. Pour prouver la première équation on pose $D_z =$

$\partial/\partial z$ et

$$f(u) := 1/\sqrt{1-4u} = \sum_{i\geq 0} \binom{2i}{i} u^i.$$

Alors, par la formule de Faà di Bruno (voir par exemple [Com74, p. 137]), on obtient

$$\begin{aligned}
\frac{z^{2p+1}}{p!} UD_u^p F(zf(u)) &= \frac{z^{2p+1}}{p!} \sum_{\sum_{i=1}^p i s_i = p} \frac{p!}{s_1! \cdots s_p!} (D^{s_1+\cdots+s_p} F)(zf(0)) \prod_{m=1}^{p} \left(\frac{1}{m!} UD_u^m (zf(u)) \right)^{s_m} \\
&= z^{2p+1} \sum_{\sum_{i=1}^p i s_i = p} \frac{1}{s_1! \cdots s_p!} (D^{s_1+\cdots+s_p} F)(zf(0)) \prod_{m=1}^{p} \left(z \binom{2m}{m} \right)^{s_m} \\
&= z^{2p+1} \sum_{\ell=1}^{p} \frac{1}{\ell!} \binom{l}{s_1, \ldots, s_p} \prod_{m=1}^{p} \binom{2m}{m}^{s_m} z^\ell D^\ell F(z) \\
&= \sum_{\ell=1}^{p} \frac{\alpha_{\ell,p}}{\ell!} z^{\ell+2p+1} D^\ell F(z) = \Delta_p F(z).
\end{aligned}$$

où on substitue $s_1+\cdots+s_p = l$ dans la seconde ligne et on sépare la somme en fonction des valeurs de ℓ et on utilise $f(0) = 1$ dans la troisième ligne.

□

Remarque. De manière plus heuristique, on peut argumenter de la façon suivante. On considère $F(z)$ comme la série d'une structure arborescente où z marque le nombre de nœuds. Alors $F(z/\sqrt{1-4u}\,)$ est la série génératrice où chaque nœud est substitué par un nœud et une séquence (potentiellement vide) d'arbres binaires à gauche ou à droite où le nombre de feuilles apparaissant par substitution est marqué par u. Puis $[u^p]F(z/\sqrt{1-4u}\,)$ est la série génératrice de tels objets où p feuilles apparaissent lors des substitutions. Le terme z^{2p+1} prend en compte l'introduction des $2p+1$ nouveaux nœuds. Ce terme vient de l'énumération des nœuds des arbres binaires venant des substitutions, en ajoutant un nœud de plus pour le nœud connectant ces arbres au chemin et ajoutant une nouvelle racine à la structure. C'est précisément ce que fait Δ_p. Cette construction est plus proche de ce que l'on va maintenant faire en section 9.3.2 pour les λ-termes clos.

Termes multi-linéaires et algorithme de Rémy

Le principe de greffe pour la spécification des λ-termes multi-linéaires commençant par un nœud unaire est très similaire à la greffe de Rémy. En fait, on peut reformuler l'ajout d'un nœud unaire comme nouvelle racine d'un $BCI(p)$ comme :

– créer successivement p feuilles par greffe de Rémy
– lier toutes les feuilles ainsi créées au nouveau nœud unaire.

En effet, lorsque l'on effectue des greffes successives on va pouvoir, en autorisant à repointer sur les nœuds nouvellement créés, faire apparaître des sous-arbres binaires dont toutes les feuilles sont liées à la racine. On en déduit la spécification suivante pour la classe des $BCI(p)$, notée $\mathcal{T}$:

$$\mathcal{T} = \mathcal{Z}\mathcal{C} + \mathcal{Z}\mathcal{T} \times \mathcal{T} + \mathcal{Z} \times \mathcal{R}^p(\mathcal{T})$$

où $\mathcal{C}$ est la classe des arbres de Catalan et $\mathcal{R}(\mathcal{A})$ est un opérateur effectuant une greffe de Rémy tel que $\mathcal{R}(\mathcal{A}) = 2\mathcal{Z}^2\mathcal{A}^\bullet$.

Cette remarque nous permet d'obtenir un générateur aléatoire utilisant la méthode récursive et l'algorithme de Rémy pour les $BCI(p)$, à condition d'avoir pré-calculé tous les premiers termes. En effet, à chaque étape on choisit par méthode récursive si la nouvelle racine est nœud binaire ou unaire. Attention, on fait ces choix en partant de l'arbre les plus gros pour aller vers les plus petits sous-arbres. Il faudra donc stocker un mémoire les choix successifs puis effectuer les modifications au moment de dépiler. Lorsque l'algorithme choisi un nœud binaire, on fait deux appels récursifs pour chacun de ses fils, en ayant déterminé leur tailles respectives. Lorsque l'algorithme choisi un nœud unaire, on fait un appel récursif pour générer un $BCI(p)$ ayant $p-1$ nœuds de moins qu'à l'étape en cours. Lorsque l'on dépile, au moment d'ajouter un nœud unaire, on fait d'abord p greffes de Rémy (en utilisant l'algorithme de Rémy) que l'on lie au nouveau nœud unaire.

On obtient ainsi un algorithme de génération aléatoire uniforme de $BCI(p)$ taille n de complexité en temps $O(n^2)$, en considérant que les calculs arithmétiques et le tirage d'un réel uniforme dans $[0,1]$ peuvent se faire en temps $O(1)$: le choix des tailles des sous-arbres pour un nœud binaire est linéaire et l'ajout d'un noeud unaire de l'ordre de $\log n$. On peut cependant utiliser une technique Boustrophédon pour les nœuds binaires, qui sont en moyenne très déséquilibrés (*cf.* section 9.4). La complexité du tirage devient alors $O(n \log n)$.

9.3.2 λ-termes clos

Soit λ_n le nombre de λ-termes clos et

$$\Lambda(z) = \sum_{n\geq 1} \lambda_n z^n$$

la série génératrice de $\mathcal{L}(\mathcal{Z})$, la classe des λ-termes clos comptés selon leur nombre de nœuds.

Proposition 3. Soit $\mathcal{M}(\mathcal{Z})$ la classe des arbres de Motzkin comptés selon leur nombre de nœuds. L'équation

$$\mathcal{L} = \mathcal{Z}\mathcal{M}(\mathcal{Z}) + \mathcal{Z}\mathcal{L}(\mathcal{Z})^2 + \mathcal{Z}\mathcal{L}(\mathcal{Z}\,\textsc{Seq}(2\mathcal{Z}\mathcal{M})) \tag{9.6}$$

est une spécification des λ-terme clos. Cette spécification se traduit automatiquement comme une équation sur la série génératrice

$$\Lambda(z) = zM(z) + z\Lambda(z)^2 + z\Lambda\left(\frac{z}{1-2zM(z)}\right). \tag{9.7}$$

Cette spécification est illustrée en figure 9.3 pour le sens constructif et en figure 9.4 pour le sens destructif.

Démonstration. Tout d'abord, remarquons que les plus petites structures correspondent aux arbres de Motzkin possédant une racine unaire supplémentaire liant toutes les

feuilles, ce qui correspond au terme $\mathcal{Z}\mathcal{M}(\mathcal{Z})$. Le terme $\mathcal{Z}\mathcal{L}(\mathcal{Z})^2$ correspond à une racine binaire. Il reste maintenant à étudier le terme $\mathcal{Z}\mathcal{L}(\mathcal{Z}\,\textsc{Seq}(2\mathcal{Z}\mathcal{M}))$, correspondant à ajouter une racine unaire.

Comme dans le cas des $BCI(p)$ on remplace ici des arêtes par des suites d'arbres dont les feuilles sont liées au nouveau nœud unaire. Ici encore, on va associer à chaque arête (y compris celle menant à la nouvelle racine) le nœud à son extrémité la plus profonde. De cette manière il suffit de remplacer chaque sommet par un sommet (car on ne veut pas le supprimer) et un chemin ayant des arbres pendants à droite ou à gauche, sans contrainte sur le nombre de feuilles crées. Par contre, on peut ici avoir des nœuds unaires qui ne sont pas liés à des feuilles, on autorise pour cela les nœuds unaires dans ces arbres pendants. Il y a donc deux types de nœuds unaires : ceux obtenus lors de l'ajout de nouvelles feuilles et ceux obtenus lors de l'ajout d'une racine unaire si toutes les arêtes ont été substituées par des séquences vides. On peut faire la différence sans difficulté en regardant si le nœud est dans un sous-arbre dont toutes les feuilles sont liées au même nœud unaire. On a donc bien une spécification des λ-termes clos.

□

On peut éviter cette discussion sur les nœuds unaires venant des arbres substitués ou d'une racine créée sans substitution grâce à la proposition suivante.

Proposition 4. Soit $C(z) = (1 - \sqrt{1-4z^2}\,)/2$ la série génératrice associée aux arbres binaires avec une racine supplémentaire, comptés selon leur nombre total de nœuds. Soit $\tilde{\Lambda}(z)$ la série génératrice solution de

$$\tilde{\Lambda}(z) = C(z) + z\tilde{\Lambda}(z)^2 + z\tilde{\Lambda}\left(\frac{z}{1-2C(z)}\right) - z\tilde{\Lambda}(z). \tag{9.8}$$

Alors, on a :

$$\Lambda(z) = \tilde{\Lambda}(\frac{z}{1-z}).$$

Démonstration. Tout d'abord, remarquons que $\tilde{\Lambda}(z)$ est la série génératrice des λ-termes clos où chaque nœud unaire lie au moins une feuille. En effet, en procède de la même manière que dans la preuve de la proposition 1, mais sans limiter le nombre de nœuds pointés (ce qui simplifie la preuve) on arrive à l'équation

$$\tilde{\Lambda}(z) = \sum_{p\geq 1} C_{p-1} z^{2p} + z\tilde{\Lambda}(z)^2 + \mathcal{D}\tilde{\Lambda}(z)$$

où $\mathcal{D} = \sum_{p\geq 1} \Delta_p$ pour les λ-termes clos où chaque nœud unaire lie au moins une feuille. Maintenant, en appliquant la proposition 2 on obtient (9.8).

Il ne reste ensuite qu'à insérer des nœuds unaires non liés sur chaque arête de l'arbre, ou au dessus de la racine. On substitue pour cela chaque nœud d'un λ-terme sans nœud unaire non lié par une séquence non vide d'atomes, représentant le nœud remplacé et la suite des nœuds unaires qui le précède.

□

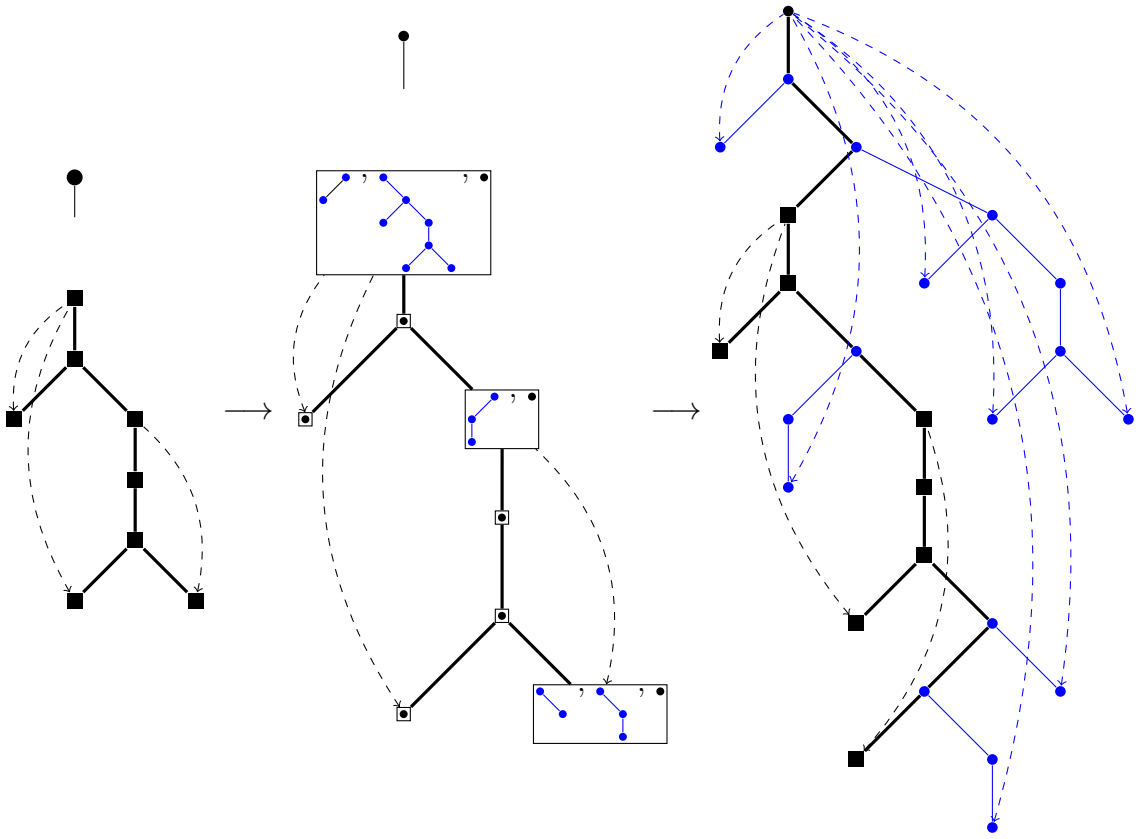


FIGURE 9.3 – Une étape de la construction d'un λ-terme. À gauche un λ-terme de taille 8 auquel on cherche à ajouter un nœud unaire à la racine. Au centre, chaque nœud contient ce par quoi il va être substitué : une séquence de chemins à laquelle des arbres de Motzkin sont attachés à droite ou à gauche, suivi du nœud d'origine. À droite le λ-terme de taille 26 obtenu. Les sommets, arêtes et liens nouvellement créés par substitution apparaissent en bleu. Les sommets d'origine sont rectangulaires.

9.3.3 λ-termes multi-affines

L'obtention de l'équation différentielle de la série génératrice pour les termes $BCK(p)$ est un peu plus complexe que pour les $BCI(p)$. On remarque que l'opérateur différentiel Δ_p correspond à créer p pointeurs de la racine vers certaines feuilles. On peut être tenté de remplacer Δ_p dans l'équation (9.2) par une somme de Δ_ℓ pour prendre en compte moins de p pointeurs. Mais la formule obtenue ainsi n'est pas entièrement correcte.

Proposition 5. Soit $F_p(z)$ la série génératrice associée aux $BCK(p)$. Alors,

$$F_p(z) = Y(z/(1-z))$$

où $Y(s)$ est l'unique série entière

$$Y(z) = \sum_{n\geq 0} Y_n z^n$$

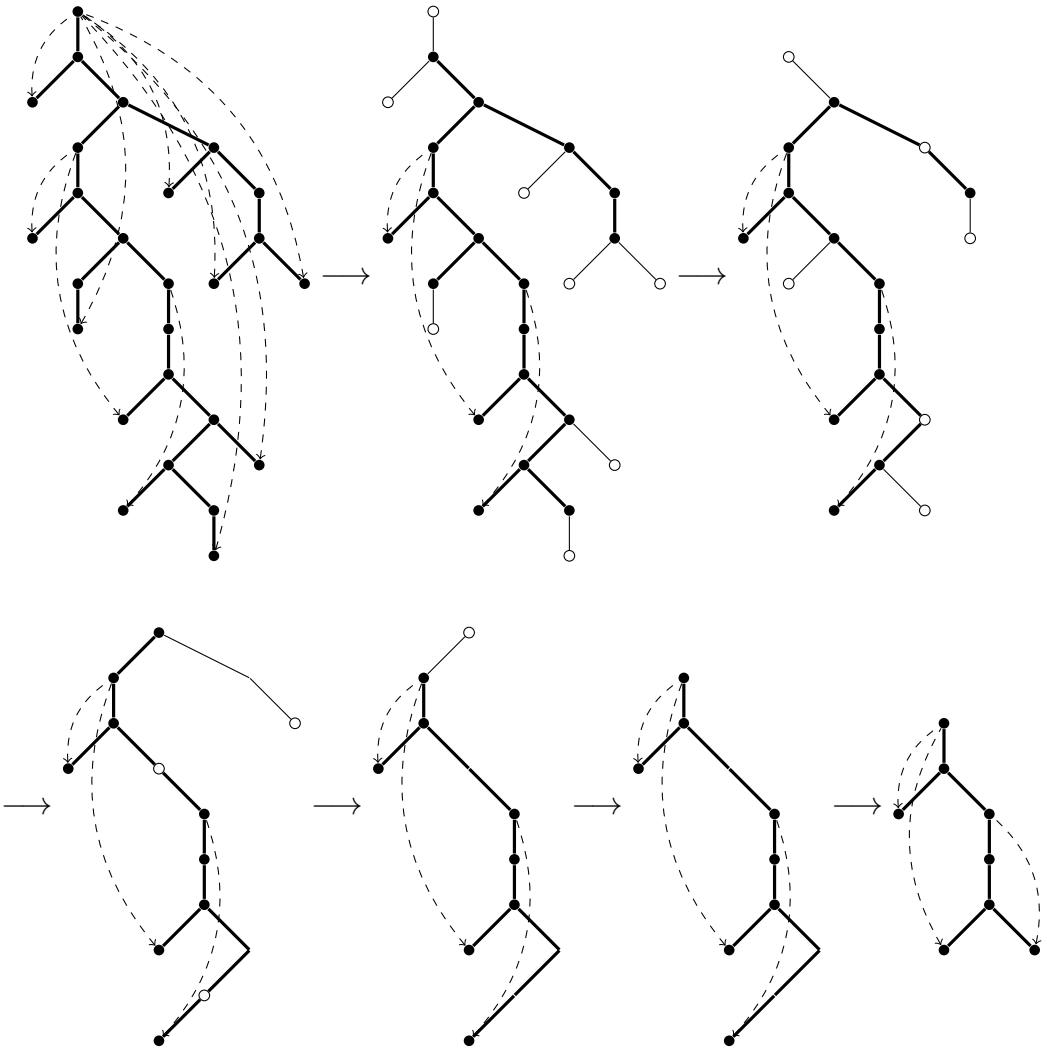


FIGURE 9.4 – Comment retrouver le λ-terme d'origine après une extension ajoutant un nœud unaire à la racine ? Tout d'abord, on colorie en blanc le nœud unaire racine et toutes les feuilles qui y était liées. Ensuite, on supprime les nœuds blancs (en raccordant les arêtes incidentes entre elles le cas échéant) et on colorie leurs voisins incomplets (nœuds binaires n'ayant pas deux fils, nœuds unaire sans fils) en blanc. On continue recursivement jusqu'à avoir un terme bien formé.

à coefficients positifs qui vérifie

$$Y(z) = \sum_{\ell=1}^{p} C_{\ell-1} z^{2\ell} + zY(z)^2 + z\left(\sum_{\ell=1}^{p} \Delta_\ell\right) Y(z). \tag{9.9}$$

Démonstration. Les $BCK(p)$ minimaux (en un certain sens) sont les arbres binaires ayant à au plus p feuilles surmontés d'un nœud unaire liant toutes les feuilles. C'est ce que l'on trouve dans le premier terme à droite de l'équation (9.9).

Remarquons qu'un noeud unaire peut également ne pas avoir de lien. Un noeud unaire sans pointeur qui n'est pas à la racine ne peut pas être directement généré par une équation très similaire à (9.5). On commence donc par construire les termes où chaque nœud unaire a au moins un lien. Des arguments similaires à ceux développés pour les $BCI(p)$ mènent directement à (9.9). Enfin, on remplace chaque arête par un chemin, ce qui permet de créer des nœuds unaires n'importe où. Cette opération correspond exactement à la substitution $z \to z/(1-z)$. □

Comme dans le cas des λ-termes clos, on peut également commencer avec les arbres de Motzkin ayant une racine unaire supplémentaire liant toutes les feuilles comme structures minimales. Le terme ayant un noeud unaire comme racine peut être généré de la manière suivante. On fixe ℓ le nombre de liens de la racine et on va maintenant sélectionner différentes arêtes, comme pour les $BCI(p)$. Mais plutôt que de substituer les arêtes sélectionnées par des séquences d'arbres binaires à droite ou à gauche, nous allons ici utiliser des séquences d'arbres de Motzkin à droite ou à gauche, avec un nœud supplémentaire les liant au chemin. On fait cette substitution de telle manière que l'ensemble de nouveaux arbres créés aient en tout ℓ feuilles. En utilisant le fait que sélectionner des arêtes correspond, au niveau de la série génératrice, à appliquer un opérateur différentiel on obtient une ainsi une équation différentielle pour $F_p(z)$.

Proposition 6. Soit $M(z,u)$ la série génératrice des arbres de Motzkin où z marque la taille (*i.e.*, le nombre total de noeuds) et u marque le nombre de feuilles. Cette fonction est l'unique série entière solution de $M(z,u) = uz + zM(z,u) + zM(z,u)^2$, c'est-à-dire

$$M(z,u) = \frac{1 - z - \sqrt{(1-z)^2 - 4uz^2}}{2z}. \tag{9.10}$$

Alors, $F_p(z)$ est la solution de

$$F_p(z) = z[u^p]\frac{M(z,u)}{1-u} + zF_p(z)^2 + z[u^p]\frac{1}{1-u}\, F_p\left(\frac{z}{1-2zM(z,u)}\right). \tag{9.11}$$

Démonstration. Ceci est une conséquence directe de la remarque ci-dessus et de la proposition 2. □

9.4 Nombre asymptotique de termes $BCI(p)$

Rappelons que

$$G_p(z) = \sum_{n\geq 1} g_{n(2p+1)-1} z^{n(2p+1)-1}$$

est la série génératrice des $BCI(p)$. La fonction $G_p(z)$ satisfait l'équation fonctionnelle (9.2) qui implique l'opérateur différentiel Δ_p donné par (9.3). Notre but est maintenant d'en déduire une équation de récurrence pour les coefficients de $G_p(z)$.

Proposition 7. Les coefficients $g_{n(2p+1)-1}$ satisfont la relation de récurrence

$$g_{n(2p+1)-1} = \sum_{l=1}^{n-1} g_{l(2p+1)-1} g_{(n-1-l)(2p+1)-1} + Q_p(n-1) g_{(n-1)(2p+1)-1}, \text{ for } n \geq 2, \quad (9.12)$$

avec la condition initiale $g_{(2p+1)-1} = C_{p-1}$ et où

$$Q_p(n) = \sum_{m=1}^{p} \alpha_{m,p} \binom{n(2p+1)-1}{m}. \quad (9.13)$$

Démonstration. Clairement, le premier terme à droite de l'équation (9.2) n'affecte que le cas $n = 1$, le terme quadratique est un produit de Cauchy et Δ_p est une combinaison linéaire de puissance de l'opérateur différentiel ordinaire, qui agit sur les coefficients de la série entière en faisant un décalage et une multiplication par $Q_p(n-1)$. □

Lemme 9.2. *Les polynômes $Q_p(n)$ peuvent être représentés plus explicitement comme*

$$Q_p(n) = 4^p \binom{\left(p+\frac{1}{2}\right) n + p - \frac{3}{2}}{p}.$$

Démonstration. Posons $f(u) = 1/\sqrt{1-4u}$. Il est facile de voir que $\alpha_{m,p} = [u^p](f(u)-1)^m$ et que le coefficient à droite vaut zéro si $m > p$. Ainsi, on obtient

$$\begin{aligned} Q_p(n) &= \sum_{m=1}^{p} \binom{(2p+1)n-1}{m} \alpha_{m,p} \\ &= [u^p] \sum_{m \geq 1} \binom{(2p+1)n-1}{m} (f(u)-1)^m = [u^p] f(u)^{(2p+1)n-1} \\ &= 4^p \binom{\left(p+\frac{1}{2}\right) n + p - \frac{3}{2}}{p}. \end{aligned}$$

□

La clef de l'analyse asymptotique est la linéarisation de l'équation différentielle, qui est possible car les coefficients de $G_p(z)$ croissent rapidement. On commence par énoncer un résultat auxiliaire pour les séquences croissant rapidement et qui dit que dans leur produit de Cauchy seuls les termes extrêmes seront importants :

Lemme 9.3. *Soient $n_0 \in \mathbb{N}$ et $A(z) = \sum_{n \geq n_0} a_n z^n$ une série entière à coefficients positifs (à partir d'un indice n_0). Supposons qu'il existe $\sigma \geq 1$ avec $a_{n+1}/a_n = \Omega(n^\sigma)$ quand $n \to \infty$. Alors $[z^n]A(z)^2 = 2a_{n_0}a_{n-n_0}(1 + O(n^{-\sigma}))$ quand $n \to \infty$. Pour obtenir le terme de second ordre, on prend les deux termes suivants, et ainsi de suite.*

Démonstration. Soit $q(n) = a_{n+1}/a_n$; alors $1/q(n) = O(n^{-\sigma})$. Supposons que n soit impair. Alors, le coefficient de z^n dans $A(z)^2$ est

$$\sum_{l=n_0}^{n-n_0} a_l a_{n-l} = 2a_{n_0}a_{n-n_0} + 2\sum_{l=1}^{\lfloor n/2\rfloor - n_0} a_{n_0+j}a_{n-n_0-j}$$
$$= 2a_{n_0}a_{n-n_0}\left(1 + \sum_{l=1}^{\lfloor n/2\rfloor - n_0} \frac{q(n_0)q(n_0+1)\cdots q(n_0+l-1)}{q(n-n_0-1)q(n-n_0-2)...q(n-n_0-l)}\right).$$

Dans le cas où n est pair, on doit soustraire $\mathbf{1}_{\{n/2\in\mathbb{N}\}}a_{n/2}^2$ au terme de droite.

Le premier terme de la somme dans la dernière ligne est

$$q(n_0)/q(n-n_0-1) = O((n-n_0-1)^{-\sigma}) = O(n^{-\sigma})$$

(rappelons que n_0 est une constante). Les termes suivants sont d'ordre $O(n^{-2\sigma})$ et il n'y en a pas plus de $\lfloor n/2\rfloor$. Donc la somme est d'ordre $O(n^{1-2\sigma}) = O(n^{-\sigma})$. D'où

$$[z^n]A^2(z) = 2a_{n_0}a_{n-n_0}(1+O(n^{-\sigma})) \sim [z^n]2a_{n_0}z^{n_0}A(z). \qquad \square$$

Nous pouvons maintenant déduire des bornes pour les coefficients de $G_p(z)$.

Lemme 9.4. *On définit $\phi_n = g_{n(2p+1)-1}$, $(n \geq 1)$. Alors on a $\phi_{n+1}/\phi_n = \Omega(n^p)$ quand $n \to \infty$.*

Démonstration. Par (9.12) on a $\phi_1 = C_{p-1}$ et, pour $n \geq 2$,

$$\phi_n = \sum_{l=1}^{n-1} \phi_l \phi_{n-1-l} + Q_p(n-1)\phi_{n-1}. \tag{9.14}$$

D'où $\phi_n \geq Q_p(n-1)\,\phi_{n-1}$. Par 9.2 il est clair que $Q_p(n)$ est polynomial en n de terme dominant $\frac{2^p(2p+1)^p}{p!}n^p$, ce qui implique le résultat. $\square$

Corollaire 1. Pour $p \geq 1$, on a l'équivalent asymptotique :

$$\sum_{m+l=n-1} \phi_m\phi_l \sim 2\phi_1\phi_{n-1}(1+O(1/n^p)).$$

Remarque. L'intuition derrière les considérations ci-dessus est celle qui suit. De notre étude de $BCI(1)$ et depuis les bornes obtenues (bien que dans un modèle différent) dans [DGK+10], on sait déjà que le comportement asymptotique du nombre de λ-termes diffère beaucoup du nombre d'arbres : la grande différence entre le nombre de λ-termes d'une taille donnée par rapport au nombre d'arbres de Motzkin, qui sont la structure arborescente sous-jacente aux λ-termes, vient du fait que l'on a un très grand nombre de façons de lier les feuilles aux noeuds unaires. Donc, le rôle du terme G_p^2, qui correspond à la structure purement binaire, est asymptotiquement négligeable comparé au terme différentiel qui prend en compte la liaison des feuilles.

Remarque. L'équation différentielle exacte pour $G_p(z)$ est (9.2) mais les arguments de la remarque 9.4 montrent que l'on peut travailler avec l'équation linéarisée [1]

$$L_p(z) = C_{p-1}z^{2p} + \Delta_p L_p(z). \tag{9.15}$$

L'équation linéarisée a également une interprétation combinatoire. En effet, elle compte le nombre de structure $\mathcal{S}$ définies comme suit. Les structures les plus petites de $\mathcal{S}$ sont exactement les plus petits $BCI(p)$, *i.e.*, un nœud unaire suivi par un arbre binaire à $2p-1$ nœuds (et toutes les feuilles sont liées au nœud unaire). Tous les termes de $\mathcal{S}$ ont une racine unaire. Pour construire leur terme plus gros, on ajoute une nouvelle racine unaire et on étend le sous-terme en dessous en utilisant le même principe de sélection et substitution que pour les $BCI(p)$ de racine unaire. Ainsi, ces termes peuvent avoir des nœuds binaires, mais jamais en tant que racine. Plus précisément, il n'y a pas de sous-arbre enraciné sur le fils d'un nœud binaire qui donne une structure bien définie (*i.e.*, dont toutes les feuilles sont liées à des nœuds de ce sous-arbre).

Lemme 9.5. *Pour $p \geq 1$, la suite $(\phi_n)_{n\geq 1}$ satisfait*

$$2\phi_1\phi_{n-1} \leq \sum_{l=1}^{n-1} \phi_l\phi_{n-1-l} \leq 2\phi_1\phi_{n-1} + (n-3)\phi_2\phi_{n-2}.$$

Démonstration. La borne inférieure est claire : on se contente de garder le premier et le dernier terme. Posons $q(n) = \phi_{n+1}/\phi_n$. Pour prouver la borne supérieure, on remarque que $(\phi_n)_{n\geq 1}$ est croissante et que pour tout $1 \leq i \leq \lceil (n-3)/2 \rceil$ on a

$$\phi_{2+i}\phi_{n-2-i} = \phi_2\phi_{n-2}\frac{q(2)q(3)\cdots q(1+i)}{q(n-2)q(n-3)\cdots q(n-1-i)} \geq \phi_2\phi_{n-2}. \qquad \square$$

Nous nous intéressons maintenant à l'équation linéarisée (9.15).

Théorème 9.6. *Posons $\ell_{p,n} = [z^n]L_p(z)$ où L_p est donné par* (9.15)*. Alors, pour p fixé et $n \to \infty$,*

$$\ell_{p,n} \sim B_p\beta_p^{n-1}n^{\gamma_p}(n-1)!^p$$

où

$$B_p = C_{p-1}\prod_{j=1}^{p}\frac{1}{\Gamma\left(1+\frac{2(p-k)-1}{2p+1}\right)} \tag{9.16}$$

$$= C_{p-1}\exp\left(-\frac{2p+1}{2}\int_1^2 \log(\Gamma(x))\,dx\right)\left(1+O\left(\frac{1}{p}\right)\right), \qquad \textit{quand } p\to\infty, \tag{9.17}$$

$$\approx C_{p-1}(1.0844375142\ldots...)^{(2p+1)/2}\left(1+O\left(\frac{1}{p}\right)\right)$$

et

$$\beta_p = \frac{(4p+2)^p}{p!}, \qquad \gamma_p = \frac{p(p-2)}{2p+1}. \tag{9.18}$$

1. Ce n'est pas une linéarisation dans le sens strict du terme puisqu'on ne remplace pas le terme quadratique par un terme linéaire mais on l'omet simplement.

p	a_p	A_p
2	1.048668...	0.981017...
3	1.0046726194...	2.19232485...
4	1.0006911656...	6.17349476...
5	1.0001221936...	19.2515312...

TABLE 9.1 – Les premières valeurs de a_p.

Démonstration. L'équation (9.15) implique $\ell_{p,2p} = C_{p-1}$ et $\ell_{p,n} = Q_p(n-1)\ell_{p,n-2p-1}$ pour $n > 2p$. D'où

$$\begin{aligned}
\ell_{p,(2p+1)n-1} &= C_{p-1} \prod_{j=1}^{n-1} Q_p(j) \\
&= C_{p-1} \left(\frac{(4p+2)^p}{p!} \right)^{n-1} \prod_{k=1}^{p} \frac{\Gamma\left(n + \frac{2(p-k)-1}{2p+1}\right)}{\Gamma\left(1 + \frac{2(p-k)-1}{2p+1}\right)} \\
&= C_{p-1} \beta_p^{n-1} (n-1)!^p \prod_{j=1}^{n-1} \prod_{k=1}^{p} \left(1 + \frac{2(p-k)-1}{2p+1} \cdot \frac{1}{j}\right) \qquad (9.19)
\end{aligned}$$

Enfin, on remarque que, quand $n \to \infty$,

$$C_{p-1} \prod_{j=1}^{n-1} \prod_{k=1}^{p} \left(1 + \frac{2(p-k)-1}{2p+1} \cdot \frac{1}{j}\right) \sim B_p n_p^{\gamma};$$

ce qui permet de conclure. La forme asymptotique (9.17) peut être obtenues par la formule d'Euler-McLaurin. □

Théorème 9.7. *Pour $p \geq 2$, le nombre de $BCI(p)$ de taille $(2p+1)n-1$ est asymptotiquement*

$$A_p \, \beta_p^{n-1} n^{\gamma_p} (n-1)!^p$$

où β_p et γ_p sont les mêmes que dans (9.18)*, $A_p = a_p B_p$ avec B_p défini comme dans* (9.16) *et $a_p = 1 + O(1/(pe^p))$, quand $p \to \infty$.*

Remarque. Les premières valeurs des constante a_p et A_p sont données dans le tableau 9.1.

Remarque. En appliquant la formule de Stirling on obtient la forme alternative

$$\bar{A}_p \bar{\beta}_p^{n-1} n^{\bar{\gamma}_p} n^{np}$$

où

$$\bar{\beta}_p = \frac{\beta_p}{e^p}, \qquad \bar{\gamma}_p = \frac{-5p}{4p+2}$$

et $\bar{A}_p = (2\pi/e^2)^{p/2} A_p$.

Démonstration. A partir de l'équation de récurrence (9.14) pour ϕ_n, on obtient

$$\begin{aligned}\phi_n &= \phi_{n-1}Q_p(n-1) + \sum_{l=1}^{n-1} \phi_l \phi_{n-1-l} \\ &= \phi_{n-1}\left(Q_p(n-1) + \Gamma_{n-1}\right),\end{aligned}$$

avec

$$\Gamma_{n-1} = \sum_{1 \le l \le n-1} \frac{\phi_l \phi_{n-1-l}}{\phi_{n-1}}$$

et $Q_p(n)$ tels que définis dans (9.13). D'où

$$\phi_n = \phi_1 \prod_{j=1}^{n-1} (Q_p(j) + \Gamma_j) = K_p(n)\phi_1 \prod_{j=1}^{n-1} Q_p(j)$$

où

$$K_p(n) = \prod_{j=1}^{n-1} \left(1 + \frac{\Gamma_j}{Q_p(j)}\right).$$

Pour $p \ge 2$ on a $Q(n) = \Omega(n^p)$ et, de plus, le lemme 9.5 donne $\Gamma_{n-1} = 2\phi_1 + O(1/n^{p-1}) = 2C_{p-1} + O(1/n^{p-1})$. Donc la séquence $(K_p(n))_{n\ge 1}$ est convergente et on obtient

$$\phi_n = a_p C_{p-1} \left(\prod_{j=1}^{n-1} Q_p(j)\right) \left(1 + O\left(\frac{1}{n}\right)\right)$$

où $a_p = C_{p-1} \cdot \lim_{n\to\infty} K_p(n)$. Le produit

$$C_{p-1} \prod_{j=1}^{n-1} Q_p(j)$$

est déjà évalué dans (9.19), ce qui mène vers le comportement asymptotique de l'équation linéarisée donnée dans le théorème 9.6.

La différence entre la linéarisation et ϕ_n est cachée dans la constante a_p. Il nous reste donc à déterminer a_p. On va se restreindre à l'évaluation asymptotique pour $p \to \infty$.

Tout d'abord, remarquons que le lemme 9.2 implique immédiatement l'inégalité

$$Q_p(n) \ge \frac{2^p(2p+1)^p}{p!} n^p. \tag{9.20}$$

Maintenant observons que $\Gamma_1 = 0$ et que par le lemme 9.5 on a $\Gamma_j \le 2\phi_1 + (j-2)\phi_2\phi_{j-1}/\phi_j$. Le quotient dans le dernier terme a déjà été estimé dans la preuve du lemme 9.4 par $\phi_{j-1}/\phi_j \le 1/Q_p(j)$. En utilisant cette estimation ainsi que l'inégalité (9.20) on obtient, pour $j > i$,

$$\Gamma_j \le 2\phi_1 + j\frac{\phi_2 p!}{2^p(2p+1)^p j^p} = 2C_{p-1} + j\frac{\phi_2 p!}{2^p(2p+1)^p j^p}.$$

D'où

$$\begin{aligned} a_p &= \prod_{j\geq 2}\left(1+\frac{\Gamma_j}{Q_p(j)}\right) \\ &\leq \prod_{j\geq 2}\left(1+\frac{C_{p-1}p!}{2^p(2p+1)^p j^p}+\frac{\phi_2 p!}{2^{2p}(2p+1)^{2p}j^{2p-1}}\right) \\ &\leq \prod_{j\geq 2}\left(1+\frac{C_{p-1}p!}{2^p(2p+1)^p j^p}\right)\prod_{j\geq 2}\left(1+\frac{\phi_2 p!}{2^{2p}(2p+1)^{2p}j^{2p-1}}\right). \end{aligned} \tag{9.21}$$

Les deux produits au dessus sont de la forme $\prod_j\left(1+\frac{\varepsilon_p}{j^p}\right)$ avec $\varepsilon\to 0$ quand $p\to\infty$. On peut donc facilement les estimer par

$$\log\prod_j\left(1+\frac{\varepsilon_p}{j^p}\right)=\sum_j\sum_{k\geq 1}\frac{(-1)^{k-1}}{k}\frac{\varepsilon_p^k}{j^{pk}}=\sum_k\frac{(-1)^{k-1}}{k}\varepsilon_p^k\zeta(pk).$$

Comme $\zeta(x)=1+O(2^{-x})$ quand $x\to\infty$ on obtient

$$\prod_j\left(1+\frac{\varepsilon_p}{j^p}\right)=1+O(\varepsilon_p).$$

Maintenant, en se référant à (9.21) on a

$$\frac{C_{p-1}p!}{2^p(2p+1)^p j^p}\sim\frac{1}{pe^p\sqrt{2e}}\qquad\text{et}\qquad\frac{\phi_2 p!}{2^{2p}(2p+1)^{2p}j^{2p-1}}=o(e^{-2p})$$

où on utilise

$$\phi_2=\phi_1 Q_p(1)=C_{p-1}4^p\binom{2p-1}{p}$$

pour la deuxième équivalence. Ceci implique $a_p=1+O(1/(pe^p))$, d'où le résultat. □

9.5 Énumération des λ-termes clos

Nous n'avons pas, pour le moment, déterminé le comportement asymptotique de λ_n. Nous allons ici obtenir des bornes supérieure et inférieure sur λ_n et une équation de récurrence permettant le calcul rapide des premiers termes de λ_n.

9.5.1 Équivalent pour λ_n

Le nombre de $BCI(p)$ donne une première borne inférieure, mais on peut, en utilisant un équivalent plutôt grossier et élémentaire, trouver une meilleure borne.

Théorème 9.8. *Le nombre λ_n de λ-termes clos de taille n satisfait, pour tout $\varepsilon>0$ et pour n assez grand, les inégalités*

$$c_1\left(\frac{4n}{e\log n}\right)^{n/2}\frac{\sqrt{\log n}}{n}\leq\lambda_n\leq c_2\left(\frac{9(1+\varepsilon)n}{e\log n}\right)^{n/2}\frac{(\log n)^{n/(2\log n)}}{n^{3/2}}$$

où c_1, c_2 sont des constantes positives.

Démonstration. On détermine le borne inférieure en comptant certains λ-termes particuliers de taille n. Prenons un arbre binaire à n_f feuilles et attachons à sa racine une séquence de n_u nœuds unaires. Connectons maintenant les feuilles aux nœuds unaires par des liens. Les objets obtenus ainsi sont des λ-termes clos et il y en a $C_{n_f} n_u^{n_f}$. Remarquons que $n_u = n + 1 - 2n_f$. On obtient alors

$$\lambda_n \geq \sum_{n_u=1}^{n-1} C_{n_f} n_u^{(n+1-n_u)/2} \geq C_{\tilde{n}_f} n_{\tilde{u}}^{(n+1-\tilde{n}_u)/2}$$

où $\tilde{n}_u$ et $\tilde{n}_f$ sont les valeurs de n_u et n_f, respectivement, où $n_u^{n_f}$ atteint son maximum. Le maximum est atteint à $\tilde{n}_u = n/W(en)$ où $W(n)$ est la fonction W de Lambert définie implicitement par $W(n)e^{W(n)} = n$. Il est facile de montrer que

$$W(en) = \log n - \log\log n + 1 + O\left(\frac{\log\log n}{\log n}\right).$$

Ce qui implique

$$\frac{n}{\log n} \leq \tilde{n}_u \leq \frac{n}{\log n - \log\log n}. \tag{9.22}$$

D'où on obtient

$$\begin{aligned}
\tilde{n}_u^{\tilde{n}_f} &\geq \left(\frac{n}{\log n}\right)^{(n/2)\cdot(1-1/(\log n-\log\log n))+1/2} \\
&= \left(\frac{n}{\log n}\right)^{n/2} \sqrt{\frac{n}{\log n}} \exp\left(-\frac{n}{2(\log n - \log\log n)}(\log n - \log\log n)\right).
\end{aligned}$$

La borne inférieure vient maintenant de $C_r \sim k_1 4^r / r^{3/2}$ $(r \to \infty)$ où k_1 est une constante positive.

Pour l'équivalent supérieur on construit un ensemble d'objets tel qu'un sous-ensemble strict correspond à tous les λ-termes de taille n. Prenons un arbre de Motzkin et ajoutons des liens tels qu'une feuille est connectée à un nœud unaire quelconque de l'arbre. Clairement, chaque λ-terme peut être généré de cette façon. Mais comme une feuille x peut également être connectée à un nœud qui ne fait pas partie de ses ancêtres, des arbres enrichis ne représentant pas de λ-terme peuvent également être obtenus. On obtient donc la borne supérieure $\lambda_n \leq M_n \max n_u^{n_f}$ où M_n est le nombre d'arbres de Motzkin à n nœuds. Comme ci-dessus, on a $n_u = n/W(en)$. Maintenant, (9.22) implique que pour n assez grand on a

$$\begin{aligned}
n_u^{n_f} &\leq \left(\frac{n}{\log n - \log\log n}\right)^{\frac{n}{2}\left(1-\frac{1}{\log n}\right)} \\
&\leq \left(\frac{(1+\varepsilon)n}{\log n}\right)^{\frac{n}{2}} \left(\frac{n}{\log n}\right)^{-\frac{n}{2\log n}} \\
&= \left(\frac{(1+\varepsilon)n}{e\log n}\right)^{\frac{n}{2}} \exp\left(\frac{n\log\log n}{2\log n}\right)
\end{aligned}$$

où on utilise $\log n/(1+\varepsilon) \le \log n - \log\log n$ pour n assez grand. Enfin, le fait bien connu $M_r \sim k_2 3^r/r^{3/2}$ (quand $r \to \infty$ et pour une constante $k_2 > 0$) permet de conclure la preuve. □

Remarque. Si $\bar{\lambda}_n$ est le nombre de λ-termes clos à n nœuds internes (autrement dit, où les feuilles ne contribuent pas à la taille), alors David *et al.* [DGK+10] ont montré le résultat suivant sur la croissance de la suite :

$$\left(\frac{(4-\epsilon)n}{\log n}\right)^{n-n/\log n} \le \bar{\lambda}_n \le \left(\frac{(12+\epsilon)n}{\log n}\right)^{n-n/3\log n}.$$

Donc la croissance exponentielle est similaire à celle de λ_n, bien que le modèle sous-jacent soit assez différent.

9.5.2 Une relation de récurrence

L'équation (9.7) implique immédiatement que λ_n suit la relation de récurrence

$$\lambda_n = M_{n-1} + \sum_{\ell+q=n-1} \lambda_\ell \lambda_q + \sum_{1\le\ell\le n-1} \delta_{n,\ell}\lambda_\ell \tag{9.23}$$

où $M_n = [z^n]M(z,1)$ est le nombre d'arbres de Motzkin de taille n et

$$\delta_{n,\ell} = [z^{n-1-\ell}]\frac{1}{(1-2zM(z))^\ell} = \sum_{r\ge 0}\binom{\ell-1+r}{\ell-1}\zeta_{n-\ell-1,r}$$

avec $\zeta_{s,r} := [z^s](2zM(z))^r$. Remarquons que $\zeta_{s,r} = 0$ sauf quand $s \ge 2r$ et donc

$$\delta_{n,\ell} = \sum_{r=0}^{\lfloor (n-\ell-1)/2\rfloor}\binom{\ell-1+r}{\ell-1}\zeta_{n-\ell-1,r}.$$

Par inversion de Lagrange on obtient

$$\zeta_{s,r} = 2^r[z^{s-r}]M(z)^r = 2^r\frac{r}{s-r}\sum_{a,b,c:\ b+2c=s-2r}\binom{s-r}{a,b,c}$$

qui donne, après quelques calculs

$$\delta_{n,\ell} = \sum_{t=0}^{\lfloor\frac{n-\ell-1}{2}\rfloor}\sum_{r=0}^{t}\frac{r2^r\binom{\ell-1+r}{r}(n-\ell-2-r)!}{t!\ (t-r)!\ (n-\ell-1-2\,t)!}. \tag{9.24}$$

Maintenant posons

$$b_{n,\ell,t} := \sum_{r=0}^{t}\frac{r2^r\binom{\ell-1+r}{r}(n-\ell-2-r)!}{t!\ (t-r)!\ (n-\ell-1-2\,t)!}.$$

Cette somme se prête à la création télescopique (voir [Zei91]) qui entraine un système de deux récurrences d'ordre un pour la séquence multi-indicée $(b_{n,\ell,t})_{n,\ell,t\geq 0}$:

$$\left(-\ell^2-2nt-2\ell t-\ell-n+n^2\right)b_{n,\ell,t}+\left(2\ell t-2\,n\ell+2\ell^2+4\ell\right)b_{n,\ell+1,t}$$
$$+\left(-4t^2-2t+4nt-4\ell t-n^2+2n\ell-\ell+n-\ell^2\right)b_{n+1,\ell,t}=0$$

et

$$(2n-t-2)\,(n-\ell-2t-2)\,(n-\ell-2t-1)\,b_{\ell,n,t}-t\,(t+1)\,(n-\ell-t-2)\,b_{\ell,n,t+1}$$
$$-(n-\ell-2t-2)(n-\ell-2t-1)\,(n-\ell-2t)\,b_{\ell,n+1,t}=0.$$

où les conditions sont données par la représentation en somme de $b_{n,\ell,t}$.
Ce système peut-être résolu explicitement et on obtient :

$$b_{n,\ell,t}=\frac{2\ell}{t}\cdot\frac{\Gamma\left(n-\ell-2\right)\ {}_2F_1(-t+1,\ell+1;-n+\ell+3;2)}{\Gamma\left(t\right)^2\Gamma\left(n-\ell-2t\right)}$$

Maintenant, en utilisant les mêmes techniques, on obtient également un système de deux récurrences D-finies pour $\delta_{n,\ell}$:

$$(n-\ell)\,(n+1-\ell)\,(n-2\ell-2)\,\delta_{n+2,\ell}-(n-\ell)\left(2n^2-6n\ell-5n+2\ell^2+3\ell+1\right)\delta_{n+1,\ell}$$
$$-(n-1)\left(3n^2-2n\ell+n-\ell^2-9\ell-8\right)\delta_{n,\ell}+20\,(n-1)\,\ell\,(\ell+1)\,\delta_{n,\ell+2}$$
$$+2\,(n-1)\,(5n-9\ell-12)\,\ell\delta_{n,\ell+1}=0$$

et

$$(n-\ell)\,(\ell-n-1)\,\delta_{n+2,\ell}+(n-\ell)\,(2n-\ell)\,\delta_{n+1,\ell}-\ell\,(n-1)\,\delta_{n+1,\ell+1}$$
$$-4\ell\,(n-1)\,\delta_{n,\ell+1}\,(n-1)\,(3n-2\ell+1)\,\delta_{n,\ell}=0.$$

Malheureusement, ce système n'admet pas de solution explicite en terme des fonctions spéciales classiques. Néanmoins, cette récurrence permet le calcul efficace des premières valeurs de $\delta_{n,\ell}$.

9.6 Conclusion et perspectives

La motivation de notre analyse était l'énumération de λ-termes clos. Comme le problème semble difficile, nous avons étudié les sous-classes $BCI(p)$ qui imposent une restriction importante sur le degré de liberté dans le lien des variables par des quantificateurs. On s'attend donc à ce que la classe des $BCI(p)$ soit petite en comparaison de celle des λ-termes clos. Nos résultats vérifient et quantifient ce fait. De plus, ils montrent que la restriction est plus faible que celle qui vise à limiter la hauteur unaire, *i.e.*, le nombre maximum de nœuds unaires sur les chemins de la racine aux feuilles. En effet, si la hauteur unaire est bornée par L, le nombre asymptotique de terme est alors $Cn^{-3/2}\rho^n$ en général ; *i.e.*, le comportement asymptotique est similaire à celui des

arbres de Motzkin. Pour certaines valeurs particulières de L seulement, le comportement asymptotique devient $Cn^{-5/4}\rho^n$ (voir [BGG11]). Ce comportement asymptotique change si l'on remplace la contrainte sur la hauteur unaire par une contrainte sur le nombre de liens qu'un nœud unaire peut avoir (comme dans le cas $BCI(p)$) ou qu'on l'abandonne complètement (tous les λ-termes clos). Ces structures sont donc en effet très différentes des structures arborescentes : leur nombre croît beaucoup plus vite que celui des arbres de Motzkin. Nous concluons donc que l'énumération des $BCI(p)$ n'est pas seulement intéressante pour l'étude de cette classe particulière, mais est aussi bien plus proche du problème d'énumération d'origine que de l'énumération des arbres.

Comme la classe des $BCK(p)$, quand p tend vers l'infini, est précisément la classe des λ-termes clos, on pourrait vouloir aborder le problème du comportement asymptotique de λ_n par le nombre de $BCK(p)$, en faisant tendre p vers l'infini. Pour obtenir une telle limite on a besoin de l'asymptotique précise du nombre de $BCK(p)$. Malheureusement, ce calcul s'avère être beaucoup plus difficile que dans le cas des $BCI(p)$. Le cas $p = 2$ a récemment été traité dans [BG14] et les auteurs y conjecturent la forme de l'asymptotique pour tout p. Pour pouvoir se rapprocher d'avantage de l'asymptotique des λ-termes clos il faut donc maintenant développer des outils de calcul formel puissants pour l'évaluation de cet asymptotique, ou trouver une nouvelle approche.

Quatrième partie

Autres travaux autour de la génération aléatoire

Chapitre 10

Arbres avec occurrences restreintes de motifs

10.1 Introduction

Dans ce chapitre, nous cherchons à spécifier et à générer aléatoirement des arbres en contrôlant le nombre de motifs apparaissant dans ces arbres. La génération aléatoire de plusieurs objets combinatoires selon leur occurrences de motifs (ou évitant certains motifs) a récemment été étudiée (comme les mots [CR02], les permutations [BBP+12, Pie13], les marches [BFPW13], les graphes pour certains mineurs [BMW13], les arbres où les sous-arbres sont les motifs considérés [FMJ09]...) Plusieurs méthodes différentes ont été développées dans ce but. Ici, nous allons utiliser des spécifications pondérées (cf section 1.5).

Les spécifications pondérées permettent de favoriser certaines configurations. Un cas simple et utile est de pondérer un certain type d'atome, sur un niveau de récursion comme par exemple les arbres unaires-binaires dont les nœuds unaires sont pondérés. La spécification bivariée se décrit alors facilement en ajoutant une marque sur les éléments concernés. Mais il est assez courant de vouloir pondérer des motifs, *i.e.*, des configurations qui se forment sur plusieurs niveaux de récursion. Pour cela, il ne suffit plus d'ajouter des marques sur une spécification existante, mais de d'abord trouver une spécification permettant d'isoler les cas formant ces motifs pour pouvoir ensuite les pondérer.

Nous nous intéressons ici aux arbres enracinés planaires (d'arités potentiellement restreintes). Soit T un arbre planaire, un arbre M à f feuilles est un *motif* de T s'il existe des arbres $t_1, t_2, ...t_f$ tels que l'arbre obtenu en substituant chaque feuille i de M par t_i est un sous arbre de T (nous reviendrons sur cette définition dans la sous-section 10.2.1). Pour cela, nous allons écrire un algorithme permettant d'obtenir une spécification sous forme d'un système d'équations. Chaque ligne de ce système correspondra à un début de motif de profondeur donnée. Afin de rendre la grammaire non-ambigüe tout en comptant toutes les occurrences de motifs, nous devons considérer le phénomène d'auto-corrélation des motifs (défini à la sous-section 10.2.3) d'arbres. Une fois cette spécification obtenue, il est alors possible de l'utiliser pour

générer uniformément des arbres de taille donnée ayant un nombre d'occurrences d'un motif fixé ou pour compter le nombre d'occurrences d'un motif dans un arbre. Cet algorithme peut être rapproché de l'automate (de mots) permettant de compter le nombre d'occurrence d'un motif [CR02] : là aussi il faudra calculer l'auto-corrélation pour pouvoir, lorsque l'on rajoute une lettre à la fin d'un mot, passer dans le bon état de l'automate. La principale différence dans les arbres est qu'un arbre peut être continué en chacune de ses feuilles : il faudra alors considérer en même temps l'extension de toutes les feuilles pouvant entrainer une auto-corrélation où les motifs se superposent par endroits.

Dans [CDKK08], les auteurs obtiennent une grammaire pour une notion de motifs similaires à celle que nous utilisons ici dans le cadre des arbres non planaires. Leur méthodes peuvent être adaptées au cas planaire, cependant, la grammaire obtenue est alors au moins aussi grosse que celle que nous obtenons ici. Plus précisément, dans le pire cas, la grammaire est identique (tant que l'arité maximale des nœuds ne croît pas avec la taille du motif) mais pour des cas plus favorables notre algorithme sera meilleur. Nous conjecturons que notre algorithme est polynomial en moyenne en la taille du motif, alors que l'algorithme de [CDKK08] est toujours exponentiel.

Nous commençons, par une première section, par définir les notions fondamentales que nous utilisons dans notre algorithme. La deuxième section est dévouée à la présentation de l'algorithme permettant d'obtenir automatiquement depuis un motif M donné la spécification bivariée des arbres comptés selon leur taille et leur nombre d'occurrences de M. La section 10.4 décrit brièvement comment utiliser cette spécification pour obtenir un générateur aléatoire, et pour la reconnaissance de motifs. Enfin, dans la dernière section, nous explorons des pistes d'extensions.

10.2 Définitions

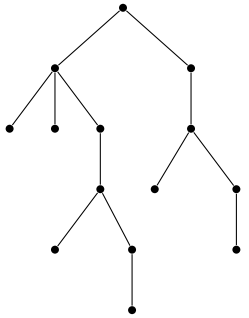
Nous allons commencer par donner diverses définitions sur les arbres enracinés planaires que nous utiliserons ensuite pour définir notre algorithme. Nous considérons ici que la taille d'un arbre est son nombre total de nœuds et que le plus petit arbre est l'arbre réduit à une feuille, de taille 1.

10.2.1 Préfixe, suffixe and motifs

Nous commençons par définir, et illustrer dans la figure 10.1 , les motifs auxquels nous nous intéressons.

Soit T un arbre, un arbre S est un *suffixe* si et seulement s'il existe un nœud n dans T tel que S est le sous-arbre issu de n.
Soit k le nombre de feuilles de T, un arbre P est un *préfixe* si et seulement s'il existe un k-uplet d'arbres $\mathcal{S} = (S_1, \cdots, S_k)$ tel que l'on obtient T en substituant, pour tout $i \in \{1, .., k\}$, la feuille i par l'arbre S_i dans P.

Définition 10.1. Un arbre M est un *motif* de T si et seulement s'il existe un suffixe S de T tel que M est préfixe de S.



Arbre	Occurrences en tant que préfixe	Occurrences en tant que suffixe	Occurrences en tant que motif
	1	2	3
	1	0	1
	0	0	0
	0	0	2
	0	0	1
	0	2	3

FIGURE 10.1 – Exemples de préfixes, suffixes et motifs : pour des arbres donnés, le tableau à droite montre combien de fois ils apparaissent en tant que préfixes, suffixes et motifs dans l'arbre de gauche.

Soit T et M deux arbres, n un nœud de T. On dit que M apparait comme motif enraciné en n si M est un préfixe du sous-arbre enraciné en n.

10.2.2 Troncatures et segments

Définition 10.2. La *troncature* d'un arbre T à profondeur d, notée $tronc_T(d)$, est le plus grand préfixe de T de profondeur d, *i.e.*, l'arbre réduit à ses nœuds de profondeur au plus d.

On remarque qu'il y a hauteur$(T) + 1$ troncatures de T. Une troncature de T est dite *stricte* si et seulement si elle est différente de T.

Une feuille d'un arbre est dit *active* si elle est de profondeur maximale, c'est à dire qu'elle est de même profondeur que l'arbre. L'ensemble de feuilles actives d'un arbre T est noté $active(T)$.

Cette définition va nous servir pour construire des arbres depuis la racine en les faisant croitre simultanément sur toutes leurs feuilles actives. On construit ainsi un arbre par ses troncatures, en y ajoutant des *segments*.

Définition 10.3. Un segment $s \in \mathbb{N}^{\mathbb{N}}$ est une séquence d'arités. Le *segment d'un arbre* T à profondeur $d \geq 1$, noté $segment_T(d)$, est la séquence des arités des nœuds de profondeur d.

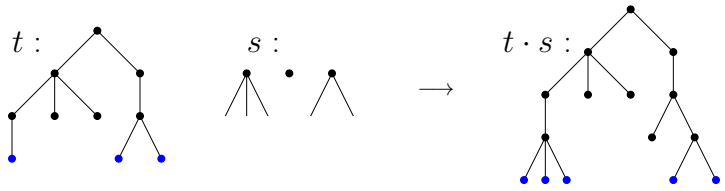


FIGURE 10.2 – De gauche à droite : un arbre t ayant 3 feuilles actives (représentées en bleu), un segment $s = (3, 0, 2)$ compatible avec t (car s est de longueur 3), et l'arbre $t \cdot s$ obtenu par concaténation de s à t.

La longueur de $segment_T(d)$ est $|active(tronc_T(d))|$. La troncature $tronc_T(d)$ peut être obtenu depuis $tronc_T(d-1)$ et $segment_T(d-1)$: c'est la propriété que nous utiliserons pour faire croitre nos arbres. On dit qu'un segment s est *compatible* avec un arbre T si et seulement si $|s| = |active(T)|$. Soit T un arbre et $s = (s_1, ..., s_k)$ un segment compatible avec T. L'arbre T' obtenu en substituant à chaque feuille active i de T par un nœud d'arité s_i dont tous les fils sont des feuilles, noté $T \cdot s$, est appelé la concaténation de s à T.

10.2.3 Corrélations

Afin d'obtenir une grammaire non ambigüe qui compte bien toutes les occurrences d'un motif, nous définition une corrélation dans les arbres. Ceci nous permettra de détecter lorsque deux occurrences (potentielles) du motif se superposent.

Soit S et T deux arbres, l'*arbre couvrant commun*, s'il existe, à S et T est le plus petit arbre U ayant S et T pour préfixes.

Soit deux arbres C et T, l'arbre C est dit *corrélé* à T s'il existe un nœud n de T tel que le sous-arbre issu de n à un arbre couvrant commun avec C. Le nœud n est appelé *point de corrélation* de T pour C. On dit qu'un point de corrélation n *couvre* un nœud a de T s'il existe un préfixe P de C tel que P soit un motif de T enraciné en n dont a est un nœud. Notons immédiatement que la relation de corrélation définie ici n'est pas symétrique, et que toutes les feuilles d'un arbre T sont des points de corrélation pour tout arbre C.

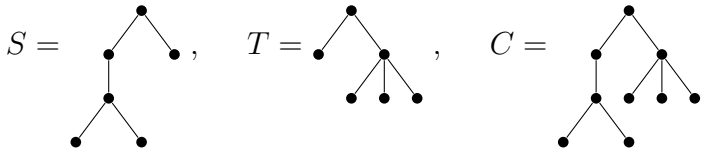


FIGURE 10.3 – De gauche à droite : un arbre S, un arbre T et leur arbre couvrant commun C.

Soit T et C deux arbres, n un point de corrélation de T pour C, la *superposition corrélée* de C en n est l'arbre où l'on a substitué le sous-arbre issu de n par l'arbre couvrant commun entre le sous-arbre issu de n et C. Cela revient à compléter T de manière à ce que C apparaisse comme motif enraciné en n.

Soit T un arbre de profondeur p et $F = tronc_T$ l'ensemble de ses troncatures, soit F_1 un élément de F de profondeur k et n un point de corrélation de F_1 pour T, on considère l'opération suivante : On construit F_2 en faisant une superposition corrélée de T en n dans F_1 puis on ajoute $tronc_k(F_2)$, la troncature à profondeur k de F_2, à F. La clôture de F par cette opération est appelée l'ensemble des *troncatures corrélées* de T. Une illustration de cette définition est donnée en figure 10.4.

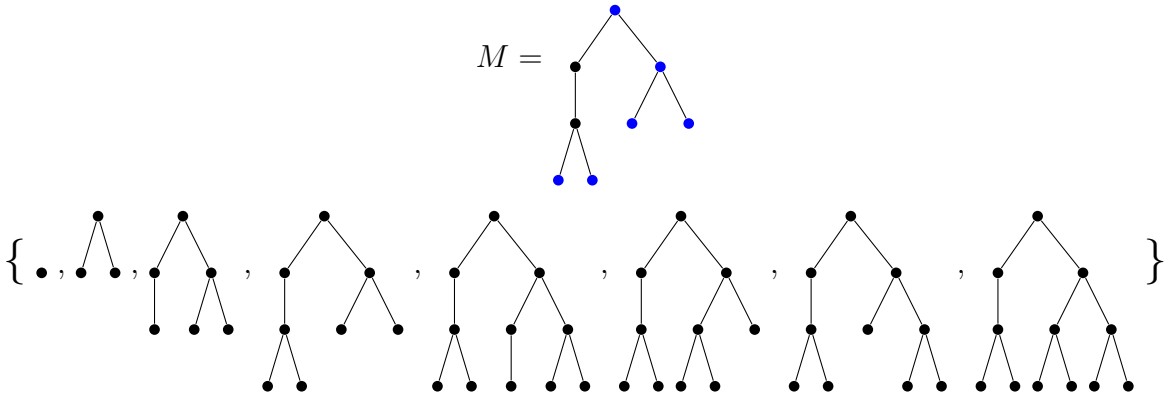


FIGURE 10.4 – Un arbre M dont les points d'auto-corrélation apparaissent en bleu et son ensemble de troncatures corrélées.

Lemme 10.4. *Soit T un arbre de profondeur p, d'arité maximale a_{max} fixée. Il y a au moins un nombre linéaire en p de troncatures corrélées de T et au plus en nombre exponentiel en p.*

Démonstration. Les troncatures de T sont des troncatures corrélées de T, il y en a donc au moins $p+1$. Il n'y a qu'un nombre exponentiel d'arbres de profondeur inférieure à p et d'arité max a_{max}, ce qui permet de conclure. □

Dans la suite, nous utiliserons uniquement un sous-ensemble de ces troncatures corrélées, les troncatures corrélées spéciales (*cf.* définition 10.7). Mais avant de les définir, nous avons besoin d'autres notions.

Définition 10.5. Soient T et C deux arbres, on associe à chaque feuille active ℓ de T un point de corrélation m_ℓ de T pour C, dit *maximum* pour ℓ, tel que m_ℓ couvre ℓ et tous les points de corrélation de T pour C couvrant f appartiennent au sous-arbre issu de n. L'ensemble des *points de corrélation maximaux* de T pour C est l'ensemble des points de corrélation qui sont maximaux pour au moins une feuille active de T.

Les figures 10.5 et 10.6 illustrent cette définition et celle des sous-arbres élagués corrélés (*cf.* définition 10.6).

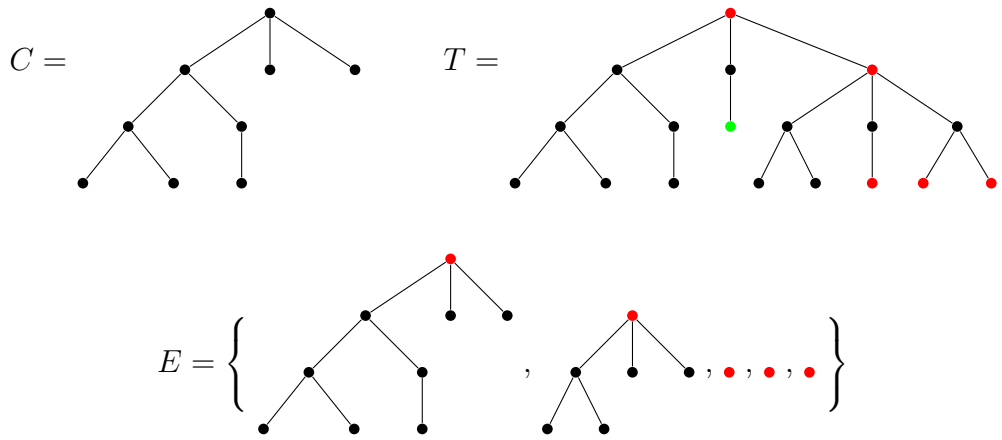


FIGURE 10.5 – Deux arbres C et T, et l'ensemble E des sous-arbres élagués de T corrélés à C. Les 5 nœuds en rouge de l'arbre T forment l'ensemble des points de corrélation maximaux pour C. Remarquons que la feuille verte, qui n'est pas active, n'est couverte par aucun point de corrélation maximum.

On peut trouver ces points en parcourant l'arbre de la racine vers les feuilles. On marque pour cela toutes les feuilles actives de l'arbre et on cherche sur chaque branche de l'arbre le nœud de corrélation le plus haut couvrant des feuilles marquées. On teste pour cela sur chaque nœud de l'arbre, du plus haut au plus bas, s'il est un point de corrélation couvrant une feuille marquée. On ajoute les nœuds ainsi trouvés à l'ensemble de points de corrélation maximaux, on supprime les marques des feuilles couvertes et on répète sur tous les sous arbres des derniers points de corrélation maximaux trouvés contenant encore des feuilles marquées. Ceci peut se faire en temps quadratique par rapport à la taille de T : on considère au plus une fois chaque nœud comme un point de corrélation maximal potentiel, et le test pour savoir si le nœud est un point de corrélation et quelles feuilles sont couvertes est également linéaire (il faut juste comparer toutes les nœuds du sous-arbre à ceux correspondant dans C).

Nous allons maintenant définir les sous-arbres élagués corrélés. Ceux-ci interviendront dans notre algorithme car c'est en considérant les segments agrandissant ces sous-arbres élagués que l'on va pouvoir former des motifs.

Définition 10.6. Soit T et C deux arbres, E_m l'ensemble de points de corrélation maximaux de T pour C. L'ensemble F des sous-arbres élagués de T corrélés à C est un ensemble de motifs de T défini comme suit. Un arbre A appartient à F si et seulement si :

- A est un motif de T enraciné en un point e de E_m,
- tous les nœuds couverts par e sont dans A,
- pour toute feuille active $f \in A$, pour tout $e' \neq e$ de E_m couvrant f, tous les nœuds couverts par e' sont dans A,
- il n'y a pas d'autre nœud dans A.

On remarque immédiatement deux choses : un arbre a au plus autant de sous-arbres élagués corrélés que de points de corrélation maximaux et chaque feuille active

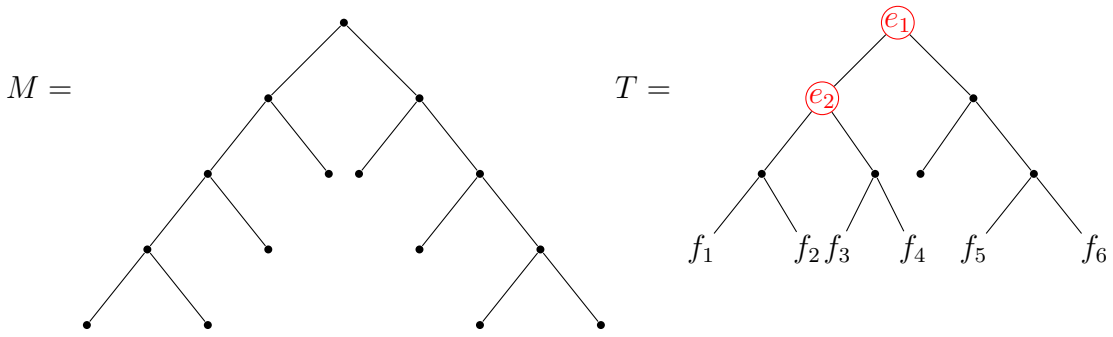


FIGURE 10.6 – Un arbre M, appelé double peigne de profondeur 4, et un arbre T. L'arbre T a 6 feuilles actives, $f_1, ..., f_6$, et deux points de corrélations maximaux e_1 et e_2. Le point e_1 couvre f_1, f_2, f_5 et f_6. Il est maximum pour ces 4 feuilles. Le point e_2 couvre f_1, f_2, f_3 et f_4. Il est maximum pour f_3 et f_4. L'ensemble des sous-arbres élagués de T corrélés à M est le singleton $\{T\}$: en effet, e_1 couvre f_1 et comme f_1 et également couvert par e_2 tous les nœuds couverts par e_1 et e_2 doivent appartenir au même sous-arbre élagué corrélé.

d'un arbre T appartient à exactement un sous-arbre élagué corrélé. On obtient donc ainsi une partition des feuilles actives entre les différents sous-arbres élagués corrélés.

Prenons de l'avance sur la section suivante avec la remarque suivante. Si on complète un arbre T par un segment s et que l'on s'intéresse au début d'occurrences d'un motif M dans T, on observe qu'un tel début de motif ne peut pas recouvrir des feuilles actives de deux sous-arbres élagués de T corrélés à C différents. Il est donc équivalent de considérer la concaténation de différents sous-segments à chacun des sous-arbres élagués corrélés, en ce qui concerne la capacité de compter les occurrences du motif C.

Passons maintenant à la dernière définition nécessaire avant de présenter notre algorithme.

Définition 10.7. Soit M un arbre de profondeur p, une troncature corrélée T de M est dite *spéciale* si et seulement si :

- L'ensemble des sous-arbres élagués de T corrélés à M est le singleton $\{T\}$,
- La profondeur de T est strictement inférieure à M.

Dans de nombreux cas, les seules troncatures corrélées spéciales seront les troncatures strictes. C'est par exemple le cas des arbres M de la figure 10.4, C et T de la figure 10.5 et T de la figure 10.6. Par contre, dans le cas du double peigne (*cf.* figure 10.6) par exemple, il y aura un nombre exponentiel de tel troncature. Un exemple est donné en figure 10.7.

Proposition 10.8. *Soit M un arbre de profondeur p, d'arité maximale a_{max} fixée, il y a au moins p troncatures corrélées spéciales de M et au plus un nombre exponentiel en p.*

Démonstration. Les troncatures strictes sont des troncatures corrélées spéciales et il y en a p (une pour chaque profondeur de 0 à $p-1$). La borne supérieure découle directement du lemme 10.4. □

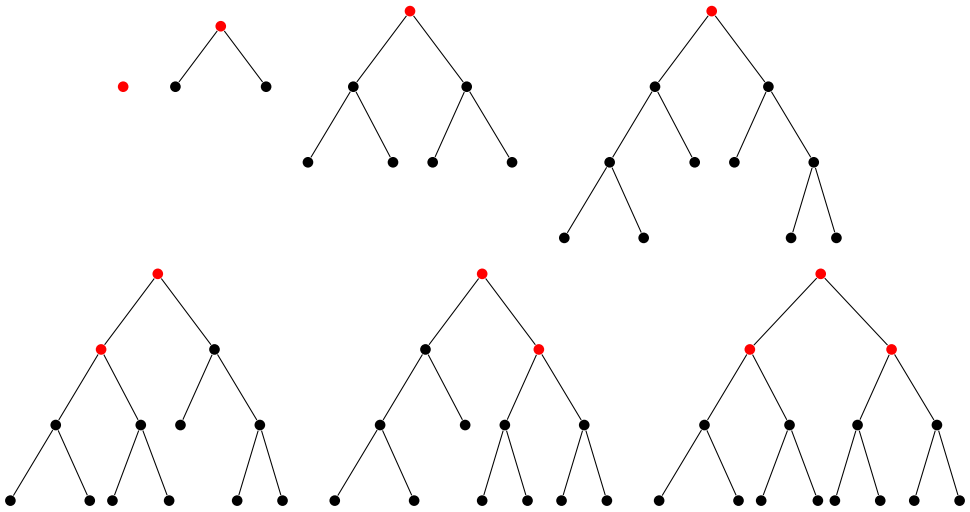


FIGURE 10.7 – Les troncatures corrélées spéciales du double peigne de profondeur 4. Les points de corrélations maximaux sont indiqués en rouge.

10.3 Construction automatique de la spécification

On cherche maintenant à définir un algorithme qui étant donné un motif M construit une spécification bivariée des arbres comptés selon leur nœuds et le nombre d'occurrences de M. Cette spécification sera constituée d'un système d'équations, où l'ensemble des membres de droite est l'ensemble des troncatures corrélées spéciales de M.

Soit M un motif de profondeur p, et soit T une troncature corrélée spéciale de M, on note $\mathcal{A}_T$ la série génératrice bivariée des arbres comptés selon leur nombre de nœuds et leur nombre d'occurrences du motif M ayant T pour préfixe. On cherche à obtenir la spécification de $\mathcal{A}_\bullet$ (où $\bullet$ représente l'arbre réduit à une feuille). Pour cela, on va écrire un système dont les membres de gauche des équations sont les $\mathcal{A}_T$ avec T une troncature corrélée spéciale de M. Les termes de gauche correspondent aux différents cas possibles lorsque l'on fait grandir T par un segment compatible lors de sa dérivation. On doit alors ajouter autant de nœuds que son nombre de feuilles actives, puis chercher les nouveaux états dans lesquels on se trouve. Comme dans le cas des spécifications algébriques pour les arbres, on a un processus branchant menant vers des sous-arbres que l'on va considérer indépendamment. La principale différence est qu'ici ce branchement ne se passe pas à tous les nœuds : si, dans un sous-arbre de T, on a une chance de voir apparaitre une occurrence du motif M, alors tous les nœuds pouvant faire partie de cette occurrence doivent être considérés simultanément lors de la prochaine étape de dérivation. Nous donnons un exemple de telle spécification dans la figure 10.8.

Soit M un motif de profondeur p,l'idée de l'algorithme est de construire cette spécification en produisant successivement chacune des troncatures corrélées spéciales par ajout de segments. Soit E l'ensemble des troncatures corrélées spéciales de M. On va construire E au fur et à mesure de l'algorithme, initialement E contient unique-

ment $\bullet$. On définit un système d'équation [1] contenant initialement la ligne incomplète « $\mathcal{A}_\bullet =$ ». À chaque fois que l'on ajoute une troncature corrélée spéciale T à E, on ajoute la ligne « $\mathcal{A}_T =$ » au système d'équation. Ensuite, on considère successivement chacunes des troncatures corrélées spéciales dans E, de la plus petite à la plus grande. Pour chaque troncature corrélée spéciale C, on note i son nombre de feuilles actives et on va concaténer successivement chaque segment compatible s (on verra plus tard comment gérer le fait que cet ensemble est infini) à C. On obtient ainsi un nouvel arbre T, et on ajoute le terme correspondant dans la spécification de A_C.

- Si T est une troncature corrélée spéciale de M, on ajoute T à E et le terme

$$\mathcal{Z}^i \mathcal{A}_T$$

  à la spécification de $\mathcal{A}_C$.
- Si T est de profondeur strictement inférieure à p et n'est pas une troncature corrélée spéciale de M : T a donc plusieurs sous-arbres élagués corrélés à M. On détermine $E_S = \{S_1, .., S_\ell\}$ l'ensemble des sous-arbres élagués de T corrélés à M. [2]
  On ajoute le terme

$$\mathcal{Z}^i \prod_{j=1}^{\ell} \mathcal{A}_{S_j}$$

  à la spécification de $\mathcal{A}_C$.
- Sinon T est de profondeur p. On va alors distinguer deux cas : celui où un motif apparaît et celui où il n'apparaît pas. Soit $F_1, ..., F_k$ les fils de la racine de T de profondeur $p-1$, *i.e.*, l'ensemble des plus grands sous-arbres contenant des feuilles actives, on détermine $E_S = \{S_1, .., S_\ell\}$ l'union des ensembles des sous-arbres élagués de $F_1, ..., F_k$ corrélés à M.
  - Si la racine de T est un point de corrélation maximum pour M, alors le motif apparaît. On ajoute le terme

$$\mathcal{U}\mathcal{Z}^i \prod_{j=1}^{\ell} \mathcal{A}_{S_j}$$

    à la spécification de $\mathcal{A}_C$.
  - Sinon, on ajoute le terme

$$\mathcal{Z}^i \prod_{j=1}^{\ell} \mathcal{A}_{S_j}$$

    à la spécification de $\mathcal{A}_C$.

Détecter les nœuds de corrélations maximum peut facilement se faire en temps polynomial par rapport à la taille du motif M.

1. Bien sur, ce n'est pas la structure de données à utiliser pour implémenter l'algorithme. Une discussion plus élaborée sur les structures de données sera présentée dans [CDJ]

2. On reconnait ici le même phénomène que pour l'automate reconnaissant $\Sigma^* u$ [CR02] : lorsque l'on ajoute une lettre, il faut trouver le plus grand suffixe du nouveau mot qui est un préfixe de u.

Revenons maintenant à l'ensemble S des segments compatibles pour un arbre T à i feuilles actives. Nous montrons ici comment on peut ne considérer qu'un nombre fini de segments compatibles. Soit e_a l'ensemble des arités apparaissant dans le motif considéré M, et soit $s = (s_1, ..., s_i) \in S$, on défini $\tilde{s} = (\tilde{s}_1, ..., \tilde{s}_i)$ par :

$$\tilde{s}_i = \begin{cases} s_i \text{ si } s_i \in e_a \\ 0 \text{ sinon} \end{cases}$$

Soit j le nombre d'éléments de s n'appartenant pas à e_a,

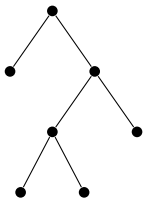
$$\mathcal{B}_j = \left(\sum_{k \notin e_a} \mathcal{A}_\bullet^k \right)^j ,$$

et $\mathcal{C}_{\tilde{s}}$ le terme obtenu dans l'algorithme par l'ajout de $\tilde{s}$ à T, le terme obtenu dans l'algorithme par l'ajout de s à T est $\mathcal{C} \times \mathcal{B}$. On peut donc se restreindre à l'ensemble $\tilde{S}$ des segments compatibles ne contenant que des arités apparaissant dans M en remplaçant $\mathcal{C}_{\tilde{s}}$ par

$$\mathcal{C}_{\tilde{s}} \times \sum_{j=0}^{d} \mathcal{B}_j$$

où d est le nombre de 0 de $\tilde{s}$.

$M =$



$$\mathcal{B} = \mathcal{A}_\bullet + \textsc{Seq}_{\geq 3}(\mathcal{A}_\bullet)$$

$$\begin{cases} \mathcal{A}_\bullet = \mathcal{Z} + \mathcal{Z}\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}\mathcal{B} \\ \mathcal{A}_{\bullet\ \bullet} = \mathcal{Z}^2 + \mathcal{Z}^2\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}^2\mathcal{A}_{\bullet\ \bullet}\mathcal{B} + \mathcal{Z}^2\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}^2\mathcal{B}\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}^2\mathcal{A}_{\bullet\ \bullet}\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}^2\mathcal{B}^2 \\ \mathcal{A}_{\bullet\ \bullet\ \bullet} = \mathcal{Z}^2 + \mathcal{Z}^2\mathcal{U}\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}^2\mathcal{U}\mathcal{A}_{\bullet\ \bullet}\mathcal{B} + \mathcal{Z}^2\mathcal{A}_{\bullet\ \bullet\ \bullet} + \mathcal{B}\mathcal{Z}^2\mathcal{A}_{\bullet\ \bullet\ \bullet} + \mathcal{U}\mathcal{A}_{\bullet\ \bullet}\mathcal{A}_{\bullet\ \bullet} + \mathcal{Z}^2\mathcal{B}^2 \end{cases}$$

FIGURE 10.8 – Un motif M et la spécification des arbres comptés selon leur taille et leur nombre d'occurrences de ce motif. En particulier, le nombre d'arbres de taille n à k occurrences de m est : $\left[\mathcal{Z}^n\mathcal{U}^k\right]\mathcal{A}_\bullet$.

Appelons l'algorithme que nous venons ainsi de définir *algorithme de spécification automatique*.

Théorème 10.9. *La spécification produite par l'algorithme de spécification automatique suit les propriétés suivantes :*

- *Chaque arbre peut être dérivé d'une unique manière.*
- *Le nombre d'arbres de taille n à k occurrences du motif M est*

$$\left[\mathcal{Z}^n\mathcal{U}^k\right]\mathcal{A}_\bullet.$$

Démonstration. La spécification est déterministe : chaque segment n'est considéré qu'une fois pour chaque troncature et la séparation en différents processus partitionne les feuilles actives. Il y a donc au plus une dérivation possible pour chaque arbre. Comme tous les segments sont considérés (via ce qui précède) pour toute troncature, tout arbre peut être dérivé. Comme dériver un arbre revient à dériver ses troncatures et à ne séparer en processus indépendants que lorsqu'une occurrence du motif ne peut pas apparaître entre les deux parties séparées, et que l'on détecte à chaque étape si on a atteint le motif, on compte bien le nombre d'occurrences. □

On remarque ici que l'on suppose que l'on connait une correspondance explicite entre un segment et le terme qu'il produit pour une ligne donnée, afin de pouvoir dériver l'arbre. Ce sera la cas pour la plupart des applications de l'algorithme et il suffit pour cela de stocker cette correspondance lors du calcul des termes.

Proposition 10.10. *Soit M un motif de profondeur p, d'arité maximale a_{max} fixée, la spécification obtenue par l'algorithme de spécification automatique a :*

- *un nombre de lignes au moins linéaire et au plus exponentiel en p.*
- *son nombre de termes apparaissants à droite des équations au plus exponentiel en p.*

Démonstration. Comme il y a une ligne par troncature corrélée spéciale exactement, la première propriété suit directement de la proposition 10.8. Le nombre de termes apparaissants à gauche de l'équation ayant $\mathcal{A}_T$ pour terme droit a

$$\sum_{j=1}^{i}\binom{i}{j}2^i a_{max}^{i-j},$$

termes, où i est le nombre de feuilles actives de T, car un segment prend ses valeurs dans $\{0, ...a_{max}\}$ et les 0 peuvent être remplacés, ou pas, par des arités supérieures à a_{max}. □

La condition d'arité maximale fixée n'est souvent pas restrictive pour les applications. Dans le cas où cette restriction ne serait pas réaliste, une méthode similaire à celle de [CDKK08] adaptée pour les arbres planaires serait probablement préférable à la notre.

Remarque. Nous avons considéré les spécifications d'arbres enracinés planaires généraux mais rien n'empêche de restreindre les arités autorisées. Dans ce cas, on ne considère pas les segments ayant des arités non-autorisées et les résultats restent corrects.

10.4 Applications de l'algorithme

10.4.1 À la génération aléatoire

Soit un motif M. L'algorithme de spécification automatique nous permet d'obtenir une spécification algébrique bivariée pour les arbres comptés selon leur taille et leur nombre d'occurrences du motif M. Cette spécification rentre dans le cadre de la méthode de Boltzmann pour les classes multi-variées [BP10]. Cette méthode permet de tirer des arbres de taille appartenant à $[n(1-\varepsilon), n(1+\varepsilon)]$ ayant approximativement m occurrences du motif (la précision dépend de n). Sa complexité [3] moyenne sera en $\Theta(n)$ (fois la taille de la spécification) car la distribution moyenne du nombre d'occurrences d'un motif dans l'ensemble des arbres de taille fixée est gaussienne [CDKK08].

La méthode récursive peut également être appliquée, mais on doit, pour générer un arbre de taille n à k occurrence de M, pré-calculer les coefficients des différentes séries entrant en jeu, jusqu'à la taille n et k occurrence du motif. La complexité en temps d'un tirage est est $\Theta(n^2)$.

Dans les deux cas, la génération est efficace si la taille du motif est très petite devant celle des arbres que l'on souhaite générer.

Pour pouvoir appliquer ce générateur à la simulation de contexte, il faut tout d'abord extraire les motifs intéressants dans les arbres étudiés. Pour beaucoup de contextes, ce n'est pas un problème évident, et l'on doit recourir à des algorithmes de fouille de données pour l'extraction de ces motifs. L'extraction d'arbres fréquents est un sujet bien connu et abordé par exemple dans [Ari08]. On fixe pour cela un seuil q, et un motif est dit fréquent s'il apparait plus de q fois dans une collection d'objets. Ceci peut être fait en complexité en temps et en espace polynomial, par rapport à la taille de la sortie. On peut détecter des motifs interdit par des méthodes similaires.

10.4.2 À la reconnaissance de motifs

L'algorithme de spécification automatique donne immédiatement un algorithme permettant de compter le nombre d'occurrences d'un motif dans un arbre. En effet avec la spécification il suffit de dériver l'arbre et de compter le nombre d'occurrences de $\mathcal{U}$: cela se fait en temps linéaire par rapport à la taille de l'arbre à dériver fois la taille de la spécification. Là encore, c'est dans le cas où le motif est beaucoup plus petit que l'arbre que notre algorithme sera efficace. Ceci est d'autant plus vrai lorsque l'on veut compter le nombre d'occurrences du même motif dans différents arbres. En effet, ici, comme pour la génération aléatoire le pré-calcul couteux de la spécification ne sera fait qu'une fois. La spécification (de taille potentiellement exponentielle en la taille du motif) devra cependant être stockée en mémoire.

Les autres méthodes linéaires de reconnaissance de motif [HO82] demandent également un pré-calcul exponentiel en la taille du motif.

Dans le cas où les motifs ne sont pas si petits par rapport à la taille de l'arbre, des

3. Le modèle de complexité considéré ici permet de tirer une loi uniforme et de faire des calculs arithmétiques en temps constant.

méthodes comme celles présentées en [CH03], de complexité $O(n \log n)$, ou [SF83], de complexité quadratique dans le pire des cas et linéaire en moyenne, seront préférables.

10.5 Extensions de l'algorithme

Notre algorithme peut facilement être étendu à au moins deux autre contextes : les arbres comptés selon leur nombre d'occurrences de différents motifs et les arbres colorés, *i.e.*, dont les nœuds portent des informations supplémentaires que les motifs doivent respecter. Ces adaptations se transfèrent sans problème à la génération aléatoire et à la reconnaissance de motifs.

10.5.1 Ensemble de motifs

Pour donner une spécification d'une classe d'arbres comptés selon différents motifs, on peut utiliser un type de marque différent pour chaque motif différent. Notons que l'on peut également s'intéresser au nombre total d'occurrence d'un des motif avec une seule marque, mais on peut immédiatement déduire une spécification pour ce modèle d'après le précédent.

Pour cela, on construit l'ensemble des troncatures spéciales de tous les motifs, corrélées à tous les autres motifs, ce qui oblige à considérer les points de corrélations (maximaux) pour chaque couple de motifs.

Les autres modifications nécessaires suivent naturellement.

10.5.2 Arbres colorés

Il est très courant de vouloir utiliser plusieurs type de nœuds de même arité. On utilisera pour ce faire des arbres colorés, *i.e.*, des arbres où chaque nœud possède une couleur déterminant son type. Pour pouvoir s'intéresser aux motifs dans de tels arbres, il faut déjà retoucher les définition de préfixe, suffixe et arbre couvrant commun pour y inclure le fait que deux nœuds à la même positions doivent être de même couleur. La définition des segments est également modifiée : un segment de longueur ℓ est alors un ℓ-uplet de séquences, potentiellement vides de couleurs. Ces couleurs détermine celles des nouvelles feuilles actives crées par concaténation du segment à un arbre. Les longueurs de chaque séquence déterminent les arités des nouveaux nœuds internes.

10.6 Conclusion et perspectives

Nous avons obtenu un algorithme permettant de spécifier automatiquement la classe bivariée des arbres enracinés planaires munie de la fonction taille nombre total de nœuds et du paramètre nombre d'occurrences d'un motif donné. Une fois ce pré-calcul effectué, on en tire un générateur aléatoire linéaire en moyenne, ainsi qu'un algorithme qui compte le nombre d'occurrences d'un motif dans un arbre. Ce dernier algorithme a une complexité linéaire en la taille de l'arbre dans lequel on veut compter

les motifs, et sera donc très efficace dans le cas où le motif est très petit par rapport à la taille de l'arbre.

Il reste néanmoins beaucoup à faire pour une implémentation pratique efficace de l'algorithme de spécification automatique. Tel que présenté ici l'algorithme refait plusieurs fois les mêmes opérations, qui pourraient être factorisées avec un choix de structures de données adaptées.

Ensuite, notre principal objectif est d'établir la complexité de notre algorithme. En effet, on voit facilement que si le motif contient peu d'arités différentes et peu de points de corrélation l'algorithme sera polynomial. Comme ceci semble être le cas pour la plupart des motifs, on conjecture que notre algorithme sera polynomial en moyenne, mais le prouver demande une analyse poussée. De plus, le pire cas n'est quand à lui pas évident à identifier. En effet, si on imagine une famille des motifs où le nombre d'arités différentes apparaissant dans un motif croît avec la taille, notre algorithme pourrait à priori être sur-exponentiel. Pour que cela se produise même avec une implémentation maligne (qui, par exemple, testerait au préalable l'existence de points de corrélation non-triviaux) il faudrait que le nombre de points de corrélation maximaux croisse également et suffisamment vite avec la taille des motifs. L'existence et l'identification d'une telle classe demande une étude combinatoire pointue de ce que doit contenir une telle structure.

Enfin, un autre problème proche de celui étudié ici est la spécification d'arbres étiquetés (par les entiers de 1 à n) à motifs ordonnés : il s'agit alors de considérer qu'un arbre M est motif étiqueté d'un arbre T si et seulement si M est un motif de T enraciné en un nœud n et les étiquettes des nœuds de T apparaissant dans cette occurrence de M ont le même ordre relatif que les étiquettes de M. La question de la spécification d'une telle classe semble être bien plus difficile.

Chapitre 11

Polyominos digitalement convexes

Introduction

En géométrie discrète, un ensemble de cellules du réseau $\mathbb{Z}^2$ constitue un *polyomino digitalement convexe* s'il existe un convexe du plan tel qu'il s'agit exactement de l'ensemble des cellules incluses à l'intérieur. [1]. On considère les polyominos digitalement convexes à translations près. Le *périmètre* est celui de la plus petite boîte rectangulaire qui le contient. Ce périmètre est également celui du polygone formé par le contour des cellules, en admettant un polygone non auto-évitant dans le cas où l'ensemble n'est pas connexe (voir figure 11.1).

La notion de polyominos digitalement convexes apparaît naturellement dans le contexte de l'estimation de courbure et est conséquemment dans le domaine de la reconnaissance de motifs géométriques [KLN08, CMT01].

Brlek *et al.* [Brl] ont décrit une caractérisation des polyominos digitalement convexes en termes de mots codant leur contour. Dans ce chapitre, nous allons reformuler cette caractérisation comme une spécification, que nous utiliserons pour analyser la forme moyenne et donner un générateur aléatoire de polyominos digitalement convexes.

L'algorithme se base sur des générateurs de Boltzmann pour tirer des quarts de polyominos digitalement convexes aléatoirement. La génération de quarts de polyominos se fait en temps linéaire, grâce à un bon paramétrage du paramètre de Boltzmann. Celle des polyominos complets est moins efficace, mais néanmoins réalisable en quelques dizaines de minutes sur un ordinateur de bureau pour des polyominos de périmètre quelques milliers.

La génération aléatoire de ces polyominos suggère l'existence d'une forme limite pour les quarts de polyominos digitalement convexes. Nous identifions et prouvons cette forme limite dans la Section 11.2.

La première section est dévouée à définir les objets d'étude et à retranscrire la caractérisation de Brlek *et al* [Brl] en tant que spécification. Comme annoncé ci-dessus, la section 11.2 se concentre sur l'analyse asymptotique des propriétés des quarts de polyominos digitalement convexes. Finalement, nous présentons en section 11.3 les

1. La définition classique des polyominos requiert que l'ensemble soit connexe, ce qui n'est pas forcément le cas des polyominos digitalement convexes.

générateurs aléatoires pour les quarts de polyominos et les polyominos complets, ainsi que l'analyse de la complexité de la génération.

11.1 Caractérisation des polyominos digitalement convexes

Le but de cette section est de rappeler (sans preuve) la caractérisation mise en évidence par Brlek, Lachaud, Provençal, Reutenauer [Brl] des polyominos digitalement convexes et de la reformuler selon les termes de la méthode symbolique. Cette caractérisation et la spécification associée sont le point de départ de l'analyse et de la génération aléatoire de ceux-ci, développées dans les sections suivantes.

11.1.1 Polyominos digitalement convexes

Rappelons tout d'abord la définition donnée en introduction :

Définition 11.1. Un *polyomino digitalement convexe*, abrégé en *PDC*, est l'ensemble de cellules de $\mathbb{Z}^2$ inclus dans une région convexe bornée du plan.

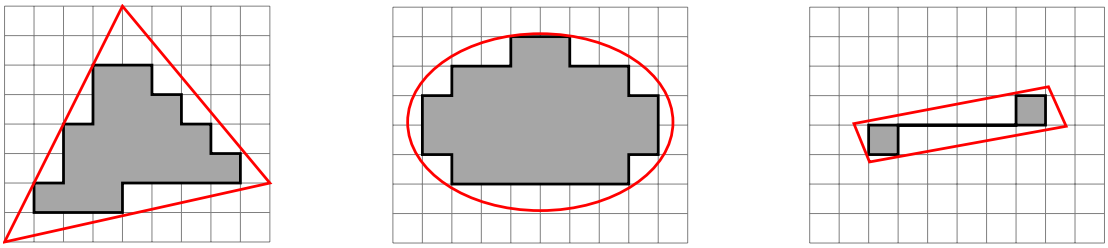


FIGURE 11.1 – Des polyominos digitalement convexes (en gris), de périmètres (de gauche à droite) 24, 26 et 16. Leur contours apparaissent en noir et les convexes du plan dans lesquels ils sont compris en rouge (notons que ceux-ci ne sont pas uniques).

Une première caractérisation découle directement de la définition : un ensemble de cellules P de la grille carrée est un polyomino digitalement convexe si et seulement si toutes les cellules incluses dans l'enveloppe convexe de P sont dans P. Une illustration de cette propriété est donnée en figure 11.2.

Ici, nous caractérisons plutôt un PDC d'après son contour.

Définition 11.2. Le *contour* d'un polyomino P est le chemin fermé sans croisement (mais pas forcément auto-évitant) ni demi-tour du réseau carré qui partitionne le plan en cellules de P et cellules n'appartenant pas à P.
Dans le cas où P n'est pas connexe (et où le chemin ne sera donc pas auto-évitant), le contour est le seul chemin respectant les conditions précédentes et qui reste à l'intérieur de l'enveloppe convexe de P. Cette définition n'a de sens que si P est digitalement convexe.

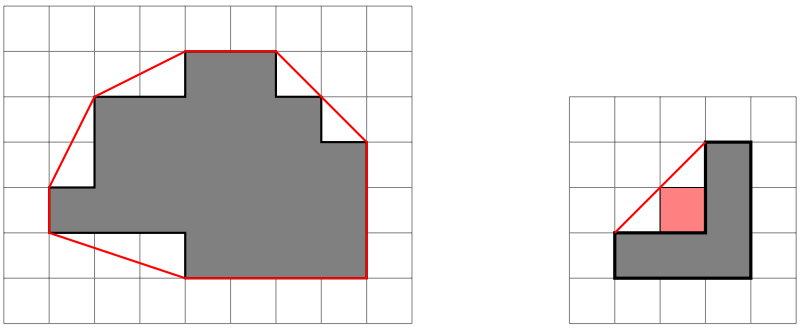


FIGURE 11.2 – Deux polyominos en gris : celui de gauche est digitalement convexe : toutes les cellules à l'intérieur de son enveloppe convexe appartiennent au polyomino. Celui de droite n'est pas digitalement convexe : la cellule en rouge est à l'intérieur de son enveloppe convexe mais n'appartient pas au polyomino.

On définit le *périmètre* d'un PDC P comme la longueur de son contour. Notons que le périmètre de P est égal au périmètre de la plus petite boîte rectangulaire contenant P.

Le contour d'un PDC peut être décomposé en quatre sous-chemins par la décomposition standard de polyominos, que nous rappelons.
La décomposition standard d'un polyomino distingue quatre points extrémaux :

- W, le point le plus bas du côté le plus à gauche,
- N, le point le plus à gauche du côté le plus haut,
- E, le point le plus haut du côté le plus à droite,
- S, le point le plus à droite du côté le plus bas.

Ces définitions sont illustrées dans la figure 11.3.

Le contour d'un PDC est alors constitué du quadruplet de chemins WN, NE, ES, et SW. En tournant ces trois derniers chemins de respectivement un quart, un demi et trois quarts de tour dans le sens trigonométrique on obtient des chemins dirigés qui ne contiennent que des pas nord et est. Ces chemins permettent une nouvelle caractérisation des polyominos digitalement convexes : un polyomino est digitalement convexe si et seulement si, une fois tournés, les chemins WN, NE, ES, et SW sont tous NW-convexes et forment, dans leurs orientations d'origine, un chemin fermé.

Définition 11.3. Un chemin p est *NW-convexe* si et seulement si il commence et finit par un pas nord et qu'il n'y a pas de cellule entre p et son enveloppe convexe supérieure (voir figure 11.3).

La taille d'un mot NW-convexe est sa longueur.

Dans la suite de ce chapitre, nous nous concentrons principalement sur la caractérisation et la génération aléatoire de chemins NW-convexes.

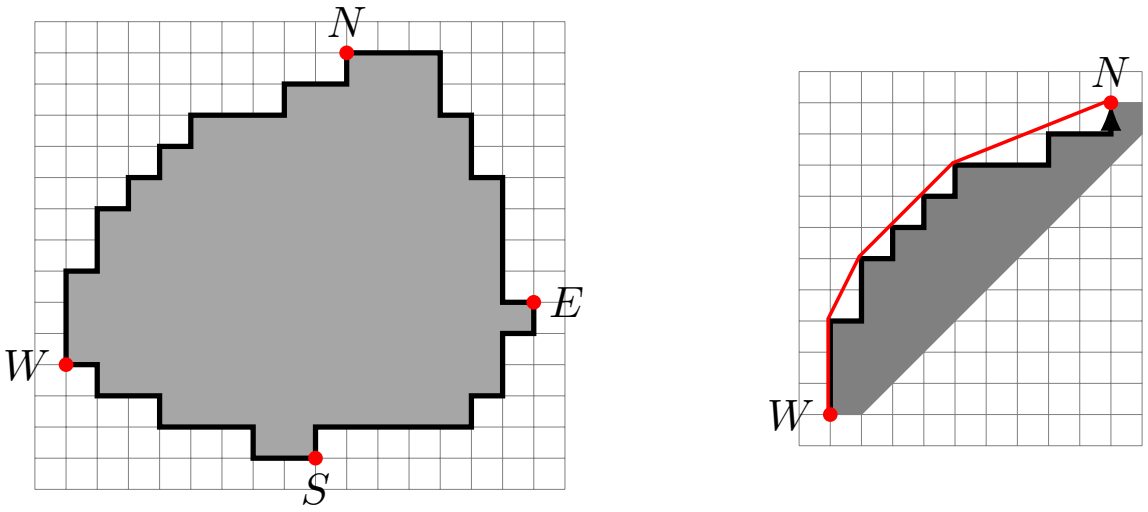


FIGURE 11.3 – Un polyomino digitalement convexes, composé de quatre chemins NW-convexes délimités par les points N, E, S et W. Le chemin NW-convexe allant de W à N est codé par le mot $w = 11101101010100010 01$.

11.1.2 Mots

La caractérisation de [Brl] est basée sur la combinatoire des mots. On s'intéresse ici aux mots sur l'alphabet $\{0, 1\}$. On note $\{0, 1\}^*$ l'ensemble de tous les mots à deux lettres et $\{0, 1\}^+$ l'ensemble de tous les mots à deux lettres d'au moins une lettre. On code les chemins dirigés en notant, depuis le point de départ jusqu'à celui d'arrivée, 0 pour un pas est et 1 pour un pas nord (voir figure 11.3).

L'idée est maintenant de décomposer un chemin NW-convexe par ses contacts avec son enveloppe convexe.

Définition 11.4. Soient p et q deux entiers, avec $p \geq 0$ et $q > 0$. Le *mot de Christoffel* associé à (p, q), ou mot de Christoffel de pente p/q, est le mot qui code le chemin le plus haut allant de $(0, 0)$ à (p, q) tout en restant sous la ligne allant de $(0, 0)$ à (p, q) (*i.e.*, le chemin codant le segment discret $[(0, 0), (p, q)]$).
Un mot de Christoffel est dit *primitif* ssi p et q sont premiers entre eux.

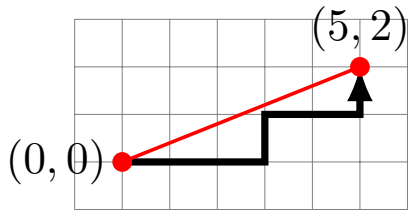


FIGURE 11.4 – En rouge, le mot de Christoffel primitif 0001001, de pente 2/5.

Notons qu'un mot de Christoffel finit toujours par 1.

11.1.3 Spécification des chemins NW-convexes

Notre spécification suivante vient du théorème suivant, tiré de [Brl] :

Théorème 11.5. *Un mot est NW-convexe si et seulement si il se décompose en une séquence de mots de Christoffel primitifs de pentes décroissantes, dont le premier commence par* 1.

Un mot NW-convexe commence forcément par 1 (*i.e.*, un chemin NW-convexe commence forcément par un pas vertical) afin d'éviter les demi-tours pour le contour d'un PDC. En effet, comme tous les mots de Christoffel primitifs terminent par 1, on assure ainsi la compatibilité avec la décomposition standard (commencer et finir dans un coin). Comme les mots de Christoffel primitifs apparaissent dans un ordre canonique (ici décroissant), la classe des chemins NW-convexes peut être identifiée à la classe des multi-ensembles de mots de Christoffel primitifs contenant le mot 1 avec multiplicité au moins 1. On peut s'affranchir de cette dernière condition en considérant plutôt le produit cartésien du singleton $\{1\}$ et de la classe des multi-ensembles de mots de Christoffel primitifs. C'est cette description que nous utilisons comme spécification. Soient $\mathcal{S}$ est la classe des chemins digitalement convexes comptés selon leur longueur, et $\mathcal{CP}$ celle des mots de Christoffel primitifs comptés selon leur longueur. On a donc :

$$\mathcal{S} = \mathcal{Z} \times \textsc{MSet}\,\mathcal{CP}.$$

La série génératrice $CP(z)$ de $\mathcal{CP}$ est :

$$\sum_{n\geq 1} \varphi(n) z^n$$

où φ est l'*indicatrice d'Euler* : $\varphi(n)$ est le nombre d'entier compris entre 1 et n premiers avec n. On en déduit la série $S(z)$ de $\mathcal{S}$:

$$S(z) = \prod_{n=1}^{\infty} (1 - z^n)^{-\varphi(n)}$$

Pour étudier plus précisément la forme limite des chemins NW-convexes, on regarde la spécification à 3 variables $\mathcal{Z}$, $\mathcal{H}$ et $\mathcal{V}$ où $\mathcal{Z}$ marque la taille (*i.e.*, la longueur totale des mots), $\mathcal{H}$ (pour horizontal) le nombre de 0 et $\mathcal{V}$ (pour vertical) le nombre de 1, si bien que le coefficient $[z^n h^i v^j] S(z, h, v)$ est le nombre de chemins NW-convexes commençant en $(0, 0)$ et finissant en (i, j). La série génératrice multivariée est alors définie par :

$$S(z, h, v) = (1 - zv)^{-1} \prod_{n=2}^{\infty} \prod_{p+q=n, p\wedge q=1} (1 - z^n v^p h^q)^{-1}.$$

11.2 Asymptotique et forme limite des chemins NW-convexes

Cette section est dédiée à l'analyse de propriétés asymptotiques des chemins NW-convexes. Le principal objectif est de décrire une forme limite pour les chemins NW-convexes normalisés (*i.e.*, dont l'échelle a été changée pour être contenu dans une

boîte de largeur 1). Cette forme limite est obtenue en trois étapes. On extrait tout d'abord le nombre asymptotique de chemins NW-convexes en utilisant une approche par transformées de Mellin. Pour la seconde étape, on utilise la même méthode pour prouver que le nombre de pas verticaux initiaux est en $O(\sqrt[3]{n})$. Finalement, on utilise des lemmes techniques pour conclure que la forme limite des chemins NW-convexes est $\sqrt{2z} - z$ avec $0 \leq z \leq 1/2$.

Commençons par un bref rappel sur les transformées de Mellin. Pour plus de détails, voir [FS09a, Annexe B.7].

11.2.1 Rappel sur les transformées de Mellin

La transformée de Mellin est une transformée intégrale similaire à la transformée de Laplace depuis laquelle on peut déduire des équivalents asymptotiques des expressions impliquant certains types de produits ou sommes infinis. Étant donnée une fonction continue f définie sur $\mathbb{R}^+$, la *transformée de Mellin* de f est la fonction :

$$\mathcal{M}[f](s) := \int_0^\infty f(t)t^{s-1}\mathrm{d}t. \tag{11.1}$$

Si $f(t) = O(t^{-a})$ quand $t \to 0^+$ et $f(t) = O(t^{-b})$ quand $t \to +\infty$, alors $\mathcal{M}[f](s)$ est une fonction analytique définie sur la *bande fondamentale* (*fundamental strip*) $a < \mathrm{Re}(s) < b$. De plus, $\mathcal{M}[f](s)$ est dans la plupart des cas prolongeable en une fonction méromorphe dans tout le plan complexe.

La première propriété fondamentale de la transformée de Mellin est qu'elle permet de factoriser les sommes harmoniques comme suit :

$$G(t) = \sum_{k\geq 1} a_k f(\mu_k t) \;\Rightarrow\; \mathcal{M}[G](s) = \Big(\sum_{k\geq 1} a_k \mu_k^{-s}\Big)\mathcal{M}[f](s). \tag{11.2}$$

De la même manière que la transformée de Laplace, la transformée de Mellin est presque involutive, la fonction $f(t)$ pouvant être retrouvée depuis $\mathcal{M}[f](s)$ en utilisant la *transformée inverse*

$$f(t) = \int_{c-i\infty}^{c+i\infty} \mathcal{M}[f](s)t^{-s}\mathrm{d}s \quad \text{pour tout } c \in (a, b). \tag{11.3}$$

Avec la transformée inverse et le théorème des résidus, le développement asymptotique de $f(t)$ quand $t \to 0^-$ peut être déduit des pôles de $\mathcal{M}[f](s)$ à gauche du domaine fondamental, le plus à droite de ces pôles donnant le terme dominant du développement asymptotique. Si $\mathcal{M}[f](s)$ décroît très rapidement quand $\mathrm{Im}(s) \to \infty$ (ce qui est le cas dans toutes les séries que nous analyserons dans la suite, puisque $\Gamma(s)$ décroît rapidement et $\zeta(s)$ a une croissance modérée lorsque $\mathrm{Im}(s) \to \infty$), alors on a la règle de transfert suivante : un pôle de $\mathcal{M}[f](s)$ d'ordre $k+1$ ($k \geq 0$),

$$\mathcal{M}[f](s) \underset{s\to\alpha}{\sim} \lambda_\alpha \frac{(-1)^k k!}{(s-\alpha)^{k+1}}$$

donne un terme

$$\lambda_\alpha t^{-\alpha} \ln(t)^k$$

dans le développement de $f(t)$ autour de 0.
En particulier, un pôle simple $\lambda_\alpha/(s-\alpha)$ induit un terme λ_α/t^α.

Déterminons tout d'abord la transformée de Mellin de $\ln(S(e^{-t}))$, ce qui est plus simple que d'obtenir la transformée de Mellin de $S(z)$. Après cela, il est assez simple de calculer le développement de S(z) quand z tend vers 1.

Lemme 11.6. *La transformée de Mellin associée à la série de mots de Christoffel primitifs est*

$$\mathcal{M}[\ln(S(e^{-t}))](s) = \frac{\zeta(s+1)\,\zeta(s-1)\,\Gamma(s)}{\zeta(s)},$$

où $\zeta(z)$ et $\Gamma(z)$ sont respectivement la fonction zeta de Riemann et la fonction Gamma.

Démonstration. La preuve repose sur un schéma exp-log, réécrivant $\ln(S(e^{-t}))$ comme une somme harmonique puis appliquant des propriétés des sommes harmoniques.

En effet, on a :

$$\ln(S(e^{-t})) = \sum_{n=1}^{\infty} -\varphi(n)\ln(1-e^{-tn}).$$

Puis, par développement de $\ln(1-x)$, on obtient :

$$\ln(S(e^{-t})) = \sum_{n=1}^{\infty}\sum_{k=1}^{\infty}\frac{1}{k}\varphi(n)e^{-tkn}.$$

On échange ensuite les sommes :

$$\ln(S(e^{-t})) = \sum_{k=1}^{\infty}\frac{1}{k}B(kt)$$

où $B(t) = \sum_{n=1}^{\infty}\varphi(n)e^{-nt}$.
Avec la propriété harmonique de la transformée de Mellin (équation 11.2), on voit que :

$$\mathcal{M}[\ln(S(e^{-t}))](s) = \zeta(s+1)\mathcal{M}[B](s).$$

On peut maintenant appliquer la propriété harmonique à B et on obtient

$$\mathcal{M}[B](s) = \frac{\zeta(s-1)}{\zeta(s)}\mathcal{M}(e^{-t})(s).$$

Enfin, comme $\mathcal{M}(e^{-t})(s) = \Gamma(s)$, on a :

$$\mathcal{M}[\ln(S(e^{-t}))](s) = \frac{\zeta(s+1)\,\zeta(s-1)\,\Gamma(s)}{\zeta(s)}$$

□

Proposition 11.7. *On a l'équivalence suivante pour $S(z)$ quand z tend vers 1^- :*

$$S(z) \sim \frac{\exp\left(\frac{6z\zeta(3)}{\pi^2(1-z)^2}\right)}{(2\pi(1-z))^{\frac{1}{6}}} \cdot \exp\left(\frac{g(1-z)+\zeta(3)}{2\pi^2} - 2\,\zeta'(-1)\right),$$

où

$$g(t) = \sum_r t^{-r}\Gamma(r)\zeta(r+1)\zeta(r-1)$$

et r parcourt les zéros non triviaux de la fonction zêta de Riemann. Et

$$g(t) = \sum_r Res_{s=r}(F(s)), \tag{11.4}$$

$$F(s) = \frac{t^{-s}\zeta(s+1)\zeta(s-1)\Gamma(s)}{\zeta(s)}, \tag{11.5}$$

où le terme de droite de l'équation (11.4) représente la somme des résidus de $F(s)$ sur les zéros non-triviaux de la fonction zêta de Riemann.

Démonstration. Notre argument est similaire à celui présenté par Brigham [Bri50] et Yang [Yan00] dans l'étude des partitions d'entiers en nombres premiers. Premièrement, on applique la transformée de Mellin inverse à la fonction du lemme 11.6 et on obtient :

$$\ln S(e^{-t}) = \frac{1}{2\pi i}\int_{c_0-i\infty}^{c_0+i\infty} t^{-s}\Gamma(s)\zeta(s+1)\frac{\zeta(s-1)}{\zeta(s)}ds$$

avec $c_0 > 2$. Comme dans [Bri50, Yan00], on décale maintenant la droite d'intégration à la verticale $\Re(s) = -c_1, 0 < c_1 < 1$, en prenant en compte les résidus de l'intégrande à $s = 2$ et $s = 0$ et aux zéros non-triviaux de la fonction zêta. Clairement, le résidu de l'intégrande à $s = 2$ est $6\zeta(3)/\pi^2 t^2$. Le pôle à $s = 0$ est du second ordre. Pour trouver son résidu on utilise les développements bien connus :

$$\zeta(s+1) = \frac{1}{s} + C + ...,$$

$$\Gamma(s) = \frac{1}{s} - C + ...$$

$$\zeta(s-1) = \zeta(-1) + \zeta'(-1)s + ...$$

$$t^{-s} = 1 - s\ln t + ...$$

et

$$\frac{1}{\zeta(s)} = -2 + 2s\ln(2\pi) + ...,$$

où C est la constante d'Euler.
En multipliant ces séries, on obtient que le résidu désiré est :

$$-\frac{1}{6}\ln t - 2\zeta'(-1) - \frac{1}{6}\ln(2\pi).$$

Ensuite, les résidus aux zêta-zéros sont accumulés par la somme :

$$\sum_r t^{-r}\Gamma(r)\zeta(r+1)\zeta(r-1) = g(t), \tag{11.6}$$

où r parcourt les zêta-zéros non-triviaux.
De cette manière, on obtient

$$\ln S(e^{-t}) = \frac{6\zeta(3)}{\pi^2 t^2} + g(t) - \frac{1}{6}\ln t - 2\zeta'(-1) - \frac{1}{6}\ln(2\pi)$$
$$-\frac{1}{2\pi i}\int_{-c_1-i\infty}^{-c_1+i\infty} t^{-s}\Gamma(s)\zeta(s+1)\frac{\zeta(s-1)}{\zeta(s)}ds.$$

Pour estimer la dernière intégrale on utilise les bornes suivantes pour les fonctions zêta et gamma :

$$\zeta(1-c_1+iy) \ll |y|^{c_1/2}\ln|y|,$$
$$\zeta(-c_1-1+iy) \ll |y|^{-1/2-c_1}\ln|y|$$

et

$$\Gamma(-c_1+iy) \ll |y|^{-c_1-1/2}e^{-\pi|y|/2}$$

pour

$$|y| \geq y_0 > 0$$

(voir par exemple [Ivi03], p. 25 et p. 492).
On estime $1/\zeta(s)$ en utilisant la représentation bien connue

$$\zeta(s) = \chi(s)\zeta(1-s), \quad \chi(s) = \frac{(2\pi)^s}{2\Gamma(s)\cos(\pi s/2)} \tag{11.7}$$

(voir [Ivi03, p. 9]).
Pour $s = -c_1 \pm |y|$, la formule de Stirling implique que

$$|\chi(-c_1 \pm |y|)| = (\frac{|y|}{2\pi})^{c_1+1/2}(1 + O(\frac{1}{|y|}),$$

$|y| \geq y_0 > 0$. De plus,

$$|\zeta(1-(-c_1+i|y|)| = |\zeta(1+c_1+i|y|)| \geq (1-\epsilon)\zeta(1+c_1), 0 < \epsilon < 1,$$

d'après [Ivi03, Thm. 9.1, p. 235]. Ce qui montre que

$$\frac{1}{\zeta(-c_1+iy)} \ll |y|^{c_1+1/2}.$$

D'où l'intégrande est absoluement intégrable et l'intégrale tend vers 0 quand $t \to 0^+$ car

$$\frac{1}{2\pi i}\int_{-c_1-i\infty}^{-c_1+i\infty} t^{-s}\Gamma(s)\zeta(s+1)\frac{\zeta(s-1)}{\zeta(s)}ds$$
$$\ll t^{c_1}\int_0^\infty e^{-\pi|y|/2}|y|^{-(c_1+1)/2}\ln^2|y|dy \ll t^{c_1}.$$

Ainsi, on obtient :

$$\ln S(e^{-t}) = \frac{6\zeta(3)}{\pi^2 t^2} + g(t) - \frac{1}{6}\ln t - 2\zeta'(-1) - \frac{1}{6}\ln(2\pi) + o(1), \quad t \to 0^+. \tag{11.8}$$

La proposition suit en changeant la variable t en $1-z$ et en prenant les exposants de gauche et de droite de l'équation (11.8).

On conclut la preuve de la proposition 11.7 par un équivalent de la fonction $g(t)$ quand $t \to 0^+$. En utilisant (11.7), on vérifie que :

$$\zeta(s-1) = \frac{\zeta(2-s)}{2(2\pi)^{1-s}\sin(\pi s/2)\Gamma(s-1)}. \tag{11.9}$$

On réprésente $\zeta(s)$ dans le dénominateur de l'équation (11.5) en utilisant le théorème de factorisation de Hadamard [Tit86, p. 30-31] comme suit :

$$\zeta(s) = \frac{e^{bs}}{2(s-1)\Gamma(s/2+1)} \prod_r \left(1 - \frac{s}{r}\right) e^{s/r}, \tag{11.10}$$

où $b = \frac{1}{6}\ln(2\pi) - 1 - C/2$ et C est la constante d'Euler. On suppose d'abord que le zéro $r = \delta + i\gamma$ $(0 \le \delta \le 1)$ de $\zeta(s)$ est simple (*i.e.*, dans l'équation (11.16) $m_r = 1$). En combinant les équations (11.5), (11.9) et (11.10), on obtient

$$(s-r)F(s) = \frac{t^{-s}s^2(1-s)\Gamma(s/2)\zeta(s+1)\zeta(2-s)e^{-bs}}{2(2\pi)^{1-s}\sin(\pi s/2)\prod_{k\ge 1, r_k \ne r}(1-s/r_k)e^{s/r_k}}. \tag{11.11}$$

Dans cette dernière formule, on suppose que les zéros complexes $r_k = \delta_k + i\gamma_k$ de $\zeta(s)$ sont triés par ordre décroissant de la valeur absolue de leur partie imaginaire $|\gamma_k|$; si certaines valeurs coïncident leur ordre respectif est choisi arbitrairement. Clairement, $0 \le \delta_k \le 1, k = 1, 2, ...$

Nous étudions maintenant le comportement de $(s-r)F(s)$ dans un voisinage de $s = r$, disons, $\{s : |s-r| < \epsilon\}$, où $\epsilon > 0$ est suffisamment petit. Soit T un nombre entier satisfaisant l'inégalité $T + 1/K_1 \ln T < |\gamma|$, où $K_1 > 0$ sera spécifié plus loin. De plus, on note $K_1, K_2, ...$ des constantes entières positives. On représente le produit dans le dénominateur de l'équation (11.11) de la manière suivante :

$$\prod_{k\ge 1, r_k \ne r} (1 - s/r_k)e^{s/r_k} = \Pi_1(s)\Pi_2(s),$$

où

$$\Pi_1(s) = \prod_{r_k \ne r, T \le |\gamma_k| < T+1} (1 - s/r_k)e^{s/r_k}, \quad \Pi_2(s) = \prod_{r_k \ne r, |\gamma_k| \notin [T, T+1)} (1 - s/r_k)e^{s/r_k}.$$

Il est maintenant clair que l'équation (11.11) peut être réécrite comme :

$$(s-r)F(s) = \frac{\psi(s)}{\Pi_1(s)}, \tag{11.12}$$

où

$$\psi(s) = \frac{t^{-s}s^2(1-s)\Gamma(s/2)\zeta(s+1)\zeta(2-s)e^{-bs}}{2(2\pi)^{1-s}\sin(\pi s/2)\Pi_2(s)} \tag{11.13}$$

est analytique dans une voisinage de $s = r$. D'où $\lim_{s\to r}\psi(s) = \psi(r)$. Pour trouver un équivalent de $\psi(r)$ on remarque d'abord que $\zeta(s)$ n'a pas de zéros dont les parties réelles valent 0 ou 1 (voir [Tit86, p. 49]). D'où $\Re(r+1), \Re(2-r) \in (1,2)$, et par [Ivi03, Thm. 1.9, p. 25],

$$\zeta(r+1) \ll \ln|\gamma|, \quad \zeta(2-r) \ll \ln|\gamma|.$$

De plus

$$\frac{1}{|\sin(\pi r/2)|} \ll e^{-\pi|\gamma|/2}, \quad \Gamma(r/2) \ll e^{-\pi|\gamma|/4}$$

et

$$t^{-r} \ll t^{-\theta},$$

où θ est la plus petite borne supérieure pour la partie réelle des zêta-zéros.
Enfin, $1/|\Pi_2(r)|$ prend ses valeurs dans le complémentaire d'un voisinage de 0 car le produit sur ces r_k satisfaisant $|\gamma_k| \le T$ n'a pas de zéro dans le disque $\{s : |s-r| < \epsilon\}$ si ϵ est assez petit et les autres facteurs pour lesquels $|\gamma_k| > T+1$ sont des progressions géométriques de raison $|r/r_k| < 1$.
En combinant ces équivalents avec l'équation (11.13), pour $|s-r| < \epsilon$, on obtient :

$$|\psi(s)| \ll t^{-\theta}|\gamma|^3\ln^2|\gamma|e^{-3\pi|\gamma|/4}. \tag{11.14}$$

Il nous reste à trouver un équivalent de $\lim_{s\to r} 1/|\Pi_1(s)|$. Nous utilisons pour cela quelques faits basiques de la théorie de la distribution des zêta-zéros dans la bande critique. Tout d'abord, on applique le fait bien connu (voir par exemple [Tit86, p. 211]) qui nous dit que, pour un T assez grand, le nombre de zêta-zéros dont la valeur absolue de leur partie imaginaire est dans l'intervalle $[T, T+1[$ est au plus $O(\ln T)$. Soit K_1 la constante multiplicative de ce O. On utilise également un équivalent pour l'écart moyen entre deux zéros successifs de $\zeta(s)$ dans un voisinage de $r = \delta + i\gamma$. Cet équivalent est $\sim 2\pi/\ln|\gamma_k| \sim 2\pi\ln|\gamma|$ (voir [Tit86, pp. 214, 246]). Enfin, il est clair que

$$|r_k| = |\delta_k + i\gamma_k| = O(T) = O(|\gamma|),$$

pour $0 \le \delta, \delta_k \le 1$ et $T \le \gamma, \gamma_k < T+1$ quand T est assez grand. On a donc :

$$\begin{aligned}|\lim_{s\to r}\Pi_1(s)| &= \prod_{r_k\neq r, T\le|\gamma_k|<T+1}|1-\frac{r}{r_k}|e^{\Re(r-r_k)} \ge K_2\prod_{r_k\neq r, T\le|\gamma_k|<T+1}\frac{|r-r_k|}{|r_k|}\\ &\ge K_3\left(\frac{2\pi}{\ln|\gamma|}\right)^{K_1\ln T}T^{-K_1\ln T} \ge K_4\left(\frac{\pi}{|\gamma|\ln|\gamma|}\right)^{K_1\ln|\gamma|} \ge K_5e^{-K_6\ln^2|\gamma|}.\end{aligned}$$

En combinant cet équivalent avec l'équation (11.16) (où $m_r = 1$) et avec la différence entre les équations (11.12) et (11.14), on obtient

$$\begin{aligned}&|Res_{s=r}(F(s)| \ll t^{-\theta}|\gamma|^3\ln^2|\gamma|\exp(\ln^2|\gamma| - 3\pi|\gamma|/4)\\ &\ll t^{-\theta}e^{-K|\gamma|}, \quad K > 0.\end{aligned} \tag{11.15}$$

Bien que tous les zéros connus r de $\zeta(s)$ dans la bande critique soient simples, *i.e.*, $m_r = 1$, ce fait n'a pas encore été prouvé. Actuellement, il n'y a que des équivalents pour m_r (pour des résultats récents dans cette direction, voir [Ivi99]). L'équivalent le plus simple et le plus ancien semble être $m_r \ll \ln|\gamma|$ (voir [Tit86, p. 211]). La preuve que (11.15) est vrai dès que $m_r > 1$ est techniquement plus compliquée. Elle suit le schéma de raisonnement donné plus haut. On doit pour cela manipuler une somme contenant $m_r = O(\ln T)$ sommants, pour $r = \delta + i\gamma, 0 \leq \delta \leq 1$ et $T \leq |\gamma| < T + 1$. Chacun de ces sommants doit contenir, après différentiation, le facteur $((s-r)/t)^{m_r-j}t^{-s}, j = 0, 1, ..., m_r - 1$. Comme $s - r$ et t tendent tous les deux vers 0, il faut garder un équilibre dans chaque sommant, tel que $((s-r)/t)^{m_r-j} \ll 1$. Pour les facteurs restants dans chaque sommant, des équivalents similaires à ceux présentés précédemment semblent être corrects. De cette manière, on peut établir l'équation (11.15).

Pour obtenir l'équivalent final de $g(t)$, on a besoin d'utiliser la borne [Tit86, p. 211] :

$$\sum_{r=\delta+i\gamma, T\leq|\gamma|<T+1} 1 \ll \ln T.$$

D'où, par (11.4),

$$g(t) \ll \sum_{r=\delta+i\gamma} e^{-K|\gamma|/2}t^{-\theta} \ll t^{-\theta}.$$

□

Remarque. Remarquons que

$$Res_{s=r}(F(s)) = \frac{1}{(m_r-1)!}\lim_{s\to r}\frac{d^{m_r-1}}{ds^{m_r-1}}((s-r)^{m_r}F(s)), \tag{11.16}$$

où m_r est la multiplicité du zêta-zéro r.

Remarque. Soit θ la plus petite borne supérieure des parties réelles des zéros non-triviaux de la fonction zêta de Riemann ($\theta = 1/2$ si l'hypothèse de Riemann est vraie). La fonction $g(t)$ satisfait également $g(t) = O(t^{-\theta})$ quand $t \to 0$.

De plus, l'équivalence de la proposition 11.7 nous permet de calculer, par une approche technique découlant d'une analyse de point col [FSZ91], la croissance asymptotique du nombre de chemins NW-convexes de taille (*i.e.*, de longueur) n :

Proposition 11.8. *Soit $p_{NW}(n)$ de nombre de chemins NW-convexes de taille n. On a :*

$$p_{NW}(n) \sim \alpha n^{-11/18}\exp\left(\beta n^{2/3} + g\left(\left(\frac{12\zeta(3)}{n\pi^2}\right)^{1/3}\right)\right),$$

où $g(t) = \sum_r t^{-r}\Gamma(r)\zeta(r+1)\zeta(r-1)$, où r parcourt les zéros non-triviaux de la fonction zêta de Riemann, et où

$$\alpha = \frac{1}{6}\,\frac{2^{5/9}\mathrm{e}^{\frac{(5/2)\,\zeta(3)-2\,\zeta'(-1)\pi^2}{\pi^2}}\sqrt[9]{\zeta\,(3)}3^{\frac{11}{18}}}{\pi^{\frac{8}{9}}} \sim 0.3338488807...$$

$$et\ \beta = \frac{3}{2^{2/3}}\left(\frac{\zeta(3)}{\zeta(2)}\right)^{1/3} = \frac{2^{-1/3}3^{4/3}\zeta(3)^{1/3}}{\pi^{2/3}} \sim 1.702263426...$$

Démonstration. La fonction $S(z)$ peut être analysée d'une manière similaire à celle de la fonction génératrice des partitions d'entiers (voir [FS09b, p. 574-578]). Ceci explique que l'on peut utiliser une anlyse de point col sur l'approximation de $S(z)$ pour obtenir l'asymptotique de $p_{NW}(n)$. Le but de la preuve est de vérifier la H-admissibilité de la fonction $S(z)$. Une fois celle-ci établie, les calculs sont standards et le résultat suit.

Soit $x_n = 1 - \left(\frac{12\zeta(3)}{\pi^2 n}\right)^{1/3}$ la racine de l'équation de point col est

$$t_n = -\ln x_n \sim \left(\frac{12\zeta(3)}{\pi^2 n}\right)^{1/3}$$

i.e., $x_n = e^{-t_n}$.
Nous allons maintenant prouver la propriété asymptotique suivante :

$$|S(x_n e^{i\theta})| = o(S(x_n)/\sqrt{b(x_n)}), \quad n \to \infty, \tag{11.17}$$

uniformément pour $t_n \le |\theta| < \pi$.

On remarque tout d'abord que

$$\frac{|S(e^{-t_n+i\theta})|}{S(e^{-t_n})} = \exp\left(\Re(\ln S(e^{-t_n+i\theta})) - \ln S(e^{-t_n})\right). \tag{11.18}$$

(Ici, $\ln(.)$ représente la branche principale de la fonction logarithme, telle que $\ln y < 0$ si $0 < y < 1$.)
Puis, en posant $\theta = 2\pi u$, pour $t_n/2\pi \le |\theta|/2\pi = |u| < 1/2$, on a

$$\begin{aligned}
&\Re(\ln S(e^{-t_n+i\theta}) - \ln S(e^{-t_n})) = \\
&= \Re\left(-\sum_{k=1}^{\infty}\varphi(k)\ln\left(\frac{1-e^{-kt_n+2\pi iuk}}{1-e^{-kt_n}}\right)\right) \\
&= -\frac{1}{2}\sum_{k=1}^{\infty}\varphi(k)\ln\left(\frac{1-2e^{kt_n}\cos(2\pi uk)+e^{-kt_n}}{(1-e^{kt_n})^2}\right) \\
&= -\frac{1}{2}\sum_{k=1}^{\infty}\varphi(k)\ln\left(1+\frac{4e^{kt_n}\sin^2(\pi uk)}{(1-e^{kt_n})^2}\right) \\
&\le -\frac{1}{2}\sum_{k=1}^{\infty}\varphi(k)\ln\left(1+4e^{kt_n}\sin^2(\pi uk)\right) \\
&\le -\frac{\ln 5}{2}\sum_{k=1}^{\infty}\varphi(k)e^{-kt_n}\sin^2(\pi uk) = -\frac{\ln 5}{2}U_n,
\end{aligned} \tag{11.19}$$

où la dernière inégalité vient du fait que $\ln(1+y) \ge \frac{\ln 5}{4}$ pour $0 \le y \le 4$. De plus, on obtient une borne inférieure pour la somme

$$U_n = \sum_{k=1}^{\infty}\varphi(k)e^{-kt_n}\sin^2(\pi uk). \tag{11.20}$$

On note par $\{d\}$ la partie fractionnaire du nombre réel d, et par $\|d\|$ la distance d à son entier le plus proche, tel que :

$$\|d\| = \begin{cases} \{d\} & \text{si} \quad \{d\} \leq 1/2, \\ 1-\{d\} & \text{si} \quad \{d\} > 1/2. \end{cases}$$

Il n'est pas difficile de montrer que

$$\sin^2(\pi d) \geq 4\|d\|^2 \tag{11.21}$$

On remarque que $\|uk\| = uk$ si $|u|k < 1/2$, c'est-à-dire, si $k < 1/2|u| \leq \pi/t_n$. D'où, en appliquant (11.21), l'inégalité $|u| \geq t_n/2\pi$ et le fait que $\varphi(k) \geq c_0 k/\ln\ln k$ pour $k \geq 3$ (fait dû à Landau), on obtient

$$\begin{aligned} U_n &\geq 4\sum_{k=1}^{\infty} \varphi(k) e^{-kt_n} \|uk\|^2 \\ &\geq 4u^2 \sum_{1\leq k\leq \pi/t_n} k^2 \varphi(k) e^{kt_n} \geq \frac{t_n^2}{\pi^2} \sum_{3\leq k\leq \pi/t_n} k^2 \left(c_0 \frac{k}{\ln\ln k} \right) e^{kt_n} \\ &\geq \frac{c_0 t_n^2}{\ln\ln(\pi/t_n)} t_n^{-4} \int_0^{\pi} y^3 e^{-y} dy \geq c_1 \frac{t_n^{-2}}{\ln|\ln t_n|}. \end{aligned} \tag{11.22}$$

D'où c_0 et c_1 sont des constantes positives. En remplaçant (11.19), (11.20) et (11.22) dans (11.18) et en prenant en compte l'équivalent asymptotique pour t_n, on a

$$|S(x_n e^{i\theta})| = O(S(x_n) \exp(-c_3 n^{2/3}/\ln\ln n)).$$

Ce qui implique directement (11.17) puisque $\sqrt{b(x_n)}$ est d'ordre const.$n^{2/3}$ - bien plus petit que l'ordre exponentiel donné ci-dessus.

□

Remarque. La contribution de $g\left(\left(\frac{12\zeta(3)}{n\pi^2}\right)^{1/3}\right)$ est une fluctuation de très petite amplitude, comme il est classiquement observé dans des analyses similaires. En particulier, cette contribution est imperceptible sur les 1000 premiers termes.

Maintenant, nous nous concentrons sur l'étude du nombre moyen de pas verticaux initiaux dans un chemin NW-convexes (ce qui correspond à la longueur du premier *run* de 1 dans le mot associé).

Lemme 11.9. *Le nombre moyen de pas verticaux initiaux est équivalent à*

$$\frac{\sqrt[3]{18\pi^2 n}}{6\sqrt[3]{\zeta(3)}}.$$

Démonstration. Une manière classique d'aborder ce type de problème est de marquer par un nouveau paramètre la contribution des pas initiaux dans la série génératrice. Ainsi, on obtient la fonction génératrice

$$S(z,u) = (1-zu)^{-1} \prod_{n=2}^{\infty} (1-z^n)^{-\varphi(n)}$$

où clairement, le coefficient $[z^n u^k]S(z,u)$ est le nombre de chemins NW-convexes de longueur n ayant exactement k pas verticaux initiaux. Maintenant, le nombre moyen de pas initiaux verticaux pour un chemin NW-convexe de taille n est juste

$$\frac{[z^n]\frac{\partial S(z,u)}{\partial u}|_{u=1}}{[z^n]S(z,1)}.$$

Nous devons maintenant extraire l'asymptotique de

$$G(z) := \frac{\partial S(z,u)}{\partial u}|_{u=1} = \frac{z}{(1-z)^2}\prod_{n=2}^{\infty}(1-z^n)^{-\varphi(n)}.$$

Là encore, nous procédons par transformée de Mellin et on obtient que :

$$\mathcal{M}[\ln(G(e^{-t}))](s) = \frac{\zeta(s+1)\left(\zeta(s-1)+\zeta(s)\right)\Gamma(s)}{\zeta(s)}$$

Donc,

$$G(z) \sim \frac{\exp\left(\frac{6z\zeta(3)}{\pi^2(1-z)^2}\right)}{(2\pi(1-z)^7)^{\frac{1}{6}}} \cdot \exp\left(\frac{g(1-z)+\zeta(3)}{2\pi^2} - 2\,\zeta'(-1)\right).$$

Enfin, en utilisant une analyse de point col et en divisant par l'asymptotique de $p_{NW}(n)$, on obtient le lemme 11.9. □

En particulier, si l'on renormalise le chemin NW-convexe par $1/n$, la contribution des pas verticaux initiaux pour la forme limite est nulle.

Nous nous intéressons maintenant à la position moyenne du dernier point d'un chemin NW-convexe aléatoire uniforme. Si l'on considère un chemin NW-convexe privé de ses pas initiaux, alors par symétrie, on conclut que la position moyenne de fin est

$$(x_n \sim \frac{n}{2}, y_n \sim \frac{n}{2}).$$

Mais, par le lemme 11.9 et le fait que la longueur renormalisée des pas initiaux verticaux est $o(1)$, on déduit que :

Lemme 11.10. *La position moyenne du point final d'un chemin NW-convexe aléatoire de taille n est*

$$(x_n \sim \frac{n}{2}, y_n \sim \frac{n}{2}).$$

En suivant la même approche, avec un peu plus de travail, on peut prouver que :

Proposition 11.11. *L'abscisse moyenne du point de pente x dans un chemin NW-convexe de taille n renormalisé par $1/n$ est*

$$\frac{1}{2}\left(1-\left(\frac{x}{1+x}\right)^2\right).$$

Démonstration. On déduit du lemme 11.10 :

$$\frac{\partial(1-zv)S(z,h,v)}{\partial h}|_{h=1,v=1} = \frac{z}{2}\frac{\partial(1-zv)S(z,h,v)}{\partial z}|_{h=1,v=1}.$$

De plus, en développant la dérivée, on a

$$\frac{\partial(1-zv)S(z,h,v)}{\partial h}|_{h=1,v=1} = (1-zv)S(z,h,v)|_{h=1,v=1}\sum_{n=2}^{\infty}\sum_{p+q=n,p\wedge q=1}\frac{pz^n}{1-z^n}.$$

On déduit de cela le lemme technique suivant :

Lemme 11.12. *L'identité suivante est vraie :*

$$\sum_{n=2}^{\infty}\frac{z^n}{1-z^n}\sum_{p+q=n,p\wedge q=1}p = \sum_{n=2}^{\infty}\frac{n\varphi(n)z^n}{2(1-z^n)} = \frac{z}{2}\frac{\frac{\partial(1-zv)S(z,h,v)}{\partial h}|_{h=1,v=1}}{(1-zv)S(z,h,v)|_{h=1,v=1}}.$$

Continuons maintenant avec la position moyenne du point de pente x pour $0 \leq x < \infty$. La position du point final, traitée précédemment, est un cas particulier de cette question (pour $x = 0$). Dans ce but, considérons la fonction génératrice

$$F_x(z,u) = (1-z)^{-1}\prod_{n=2}^{\infty}\prod_{p+q=n,p\wedge q=1,\frac{q}{p}>x}(1-z^nu^p)^{-1}\prod_{p+q=n,p\wedge q=1,\frac{q}{p}\leq x}(1-z^n)^{-1}.$$

L'abscisse moyenne du point de pente x dans un chemin NW-convexe de taille n n'est rien d'autre que l'espérance

$$\frac{[z^n]\frac{\partial F_x(z,u)}{\partial u}|_{u=1}}{[z^n]F_x(z,1)}.$$

En développant la dérivée, on a :

$$\frac{\partial F_x(z,u)}{\partial u}|_{u=1} = F_x(z,u)|_{u=1}\sum_{n=2}^{\infty}\frac{z^n}{1-z^n}\sum_{p+q=n,p\wedge q=1,\frac{q}{p}>x}p.$$

Maintenant, nous avoins besoin du petit lemme suivant de théorie des nombres :

Lemme 11.13. *L'équivalence suivante est vraie pour tout x fixé dans $[0,\infty[$:*

$$\sum_{p+q=n,p\wedge q=1,\frac{q}{p}>x}p = \frac{1}{2}\left(1-\left(\frac{x}{1+x}\right)^2\right)n\varphi(n)(1+\varepsilon(n))$$

où $\varepsilon(n) \to 0$ quand n tend vers ∞.

La preuve est immédiate.
Enfin, en utilisant le lemme 11.12, on obtient la proposition 11.11. □

À ce stade, pour prouver que la forme limite est deterministe, nous avons besoin de montrer que l'écart type de l'abscisse est en $o(n)$.

Cette preuve est longue et technique, mais suit le même chemin que ce que nous avons fait pour l'espérance.

On conclut en résolvant l'équation différentielle :

$$\left\{ f(0) = 0, 2f(z) = 1 - \frac{\left(\frac{d}{dz} f(z)\right)^2}{\left(1 + \frac{d}{dz} f(z)\right)^2} \right\}$$

qui explique le fait que la pente de $f(z)$ est x à l'abscisse

$$\frac{1}{2}\left(1 - \left(\frac{x}{1+x}\right)^2\right).$$

En conséquence, on a :

Théorème 11.14. *La forme limite pour le renormalisé par $1/n$ d'un chemin NW-convexe de taille n lorsque n tend vers l'infini est la courbe d'équation $f(z) = \sqrt{2z} - z$.*

11.3 Générateurs de Boltzmann pour les chemins NW-convexes

11.3.1 La classe des polyominos digitalement convexes

Rappelons tout d'abord que les PDC peuvent être décomposés en quatre chemins NW-convexes, chacun d'entre eux étant un multi-ensemble de segments discrets irréductibles, lesquels peuvent être décrits par la théorie des mots. On déduit de cette décomposition la série génératrice des chemins NW-convexes :

$$S(z) = \prod_{n=1}^{\infty} (1 - z^n)^{-\varphi(n)},$$

où $\varphi(n)$ est la fonction indicatrice d'Euler.

La première question qui se pose pour en adapter un générateur de Boltzmann est de déterminer le paramètre de génération x_n, qui est un point central pour la complexité du générateur en taille exacte ou approchée. Afin d'approximer x_n lorsque n tend vers l'infini (*i.e.*, lorsque l'on veut tirer de gros objets), nous utilisons l'asymptotique de $S(z)$ quand $z \to 1$, que nous avons déjà obtenu dans la section précédente.

11.3.2 La distribution de Boltzmann des chemins NW-convexes

La première étape pour analyser la complexité du générateur de Boltzmann en taille exacte ou approchée est de caractériser le type de distribution de Boltzmann : centrée, plate ou pointue (voir la section 3.3.4). Dans cette section nous prouvons que la distribution de Boltzmann est centrée pour les chemins NW-convexes. Ceci

nous assure que l'on a seulement besoin d'un nombre constant d'essais en moyenne pour tirer un objet en taille approchée. Une analyse précise nous permet également de donner la complexité du générateur en taille exacte.

Tout d'abord, on déduit de l'équivalence de $S(z)$ près de sa singularité dominante $\rho = 1$ une expression pour le réglage du paramètre de Boltzmann :

Corollaire 11.15. *Un bon choix de paramètre de Boltzmann x_n pour tirer des chemins NW-convexes de taille n, où n est grand, est*

$$x_n = 1 - \sqrt[3]{\frac{12\zeta(3)}{n\pi^2}}.$$

Démonstration. L'espérance de taille de la sortie est

$$\frac{xS'(x)}{S(x)},$$

qui est une fonction croissante en x. En utilisant l'équivalent de $S(x)$ quand x tend vers 1, on peut trouver une approximation du premier membre de l'équation $\frac{xS'(x)}{S(x)} = n$ pour obtenir :

$$\frac{12\zeta(3)}{(1-x)^3\pi^2} = n.$$

Le résultat suit immédiatement □

On vérifie ainsi la première condition de la distribution centrée.
Nous allons maintenant étudier la fraction

$$\sigma(x)/\mu_1(x).$$

Lemme 11.16.

- *L'espérance de taille de la sortie du générateur de Boltzmann satisfait :*

$$\mu_1(x) \sim \frac{12\zeta(3)}{(1-x)^3\pi^2} \text{ quand } x \text{ tend vers 1.}$$

- *La variance de la taille de la sortie satisfait :*

$$\sigma(x) \sim \frac{6\sqrt{\zeta(3)\,x}}{\pi(1-x)^2} \text{ quand } x \text{ tend vers 1.}$$

Donc la distribution de Boltzmann des chemins NW-convexes est centrée.

L'analyse de la distribution étant faite, il nous reste à décrire le générateur de Boltzmann pour les chemins NW-convexes et les polyominos digitalement convexes. En effet, ici, la spécification ne permet pas de déduire automatiquement un générateur de Boltzmann.

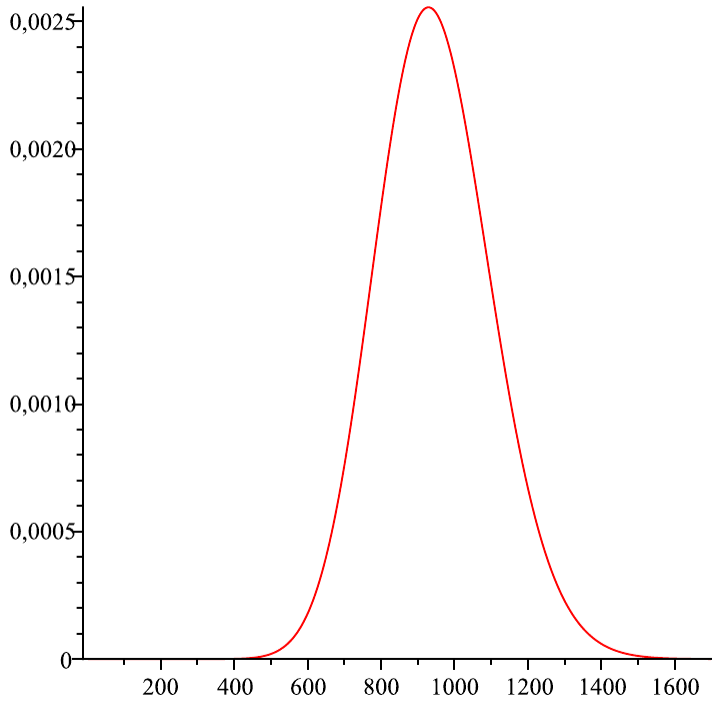


FIGURE 11.5 – La distribution du périmètre d'un chemin NW-convexe tiré avec probabilité $x = 0.8908086616$.

11.3.3 Générateur de mots de Christoffel primitifs

Nous nous concentrons tout d'abord sur la génération d'un couple d'entiers premiers entre eux. Cette classe ne peut pas être décrite par les constructeurs habituels utilisés par les générateurs de Boltzmann, nous développons donc un générateur *ad hoc* qui tire un couple d'entiers premiers entre eux selon une probabilité de Boltzmann (ici, la taille d'un tel couple est la somme de ses éléments). Nous utiliserons ensuite ce générateur pour tirer des chemins NW-convexes.

Nous cherchons donc à tirer deux entiers p, q premiers entre eux avec probabilité :

$$\frac{x^{p+q}}{\sum_{n\geq 1}\varphi(n)x^n}.$$

L'algorithme 18 répond aux critères ci-dessus, avec

$$b(x,n) := \frac{\varphi(n)x^n}{\sum\limits_{n=1}^{\infty}\varphi(n)x^n}.$$

Algorithm 18: Γ_{CP}

Input: Un paramètre de Boltzmann x
Output: Deux entiers premiers entre eux
1 Tirer n avec la probabilité de Boltzmann $b(x, n)$
2 **Do** Tirer p uniformément dans $\{1, ..., n\}$
3 **While** p, n ne sont pas premiers entre eux
4 **return** $(p, n - p)$

La complexité moyenne de l'algorithme est $O(n)$.

Correction

On a $p \wedge n = 1 \Leftrightarrow p \wedge n - p = 1$, donc tirer un entier p premier avec n uniformément dans $\{1, .., n\}$ est équivalent à tirer uniformément p, q premiers entre eux avec $p+q = n$. En choisissant n avec probabilité

$$\frac{\varphi(n)x^n}{\sum_{n=1}^{\infty} \varphi(n)x^n},$$

on obtient un générateur de Boltzmann pour les mots de Christoffel primitifs.

Complexité

Pour évaluer la complexité de cet algorithme deux étapes doivent être analysées : le tirage d'un nombre n avec probabilité $b(x, n)$ et le tirage de deux nombres premiers entre eux.

La complexité du tirage de n est essentiellement liée à l'évaluation de la série génératrice et de $\varphi(k)$ pour tout $k \leq n$. Dans la suite, on considère que l'évaluation de la série génératrice et une table assez grande de valeur de $\varphi(k)$ sont pré-calculés. La complexité du tirage de n est donc en $O(n)$. Expérimentalement, les pré-calculs nécessaires prennent quelques minutes sur un ordinateur personnel standard pour des chemins NW-convexes de tailles allant jusqu'à 100000.

Évaluons maintenant la complexité du tirage de deux nombres premiers entre eux. La probabilité qu'un nombre p tiré uniformément dans $\{1, .., n\}$ soit premier avec n est $\varphi(n)/n$. L'inégalité classique sur l'indicatrice d'Euler (voir [BS96, Thm 8.8.7])

$$\frac{\varphi(n)}{n} > \frac{1}{e^{\gamma} \log\log(n) + \frac{3}{\ln\ln(n)}},$$

valide pour tout entier positif n, prouve que le nombre d'essais moyens dans la boucle est en $O(\log\log(n))$.

Remarque. En fait, l'algorithme fera en moyenne moins d'essais dans la boucle que dans le cas moyen. En effet, la probabilité de Boltzmann fait que l'on a tiré n en favorisant les cas où $\varphi(n)$ est proche de n, autrement dit, les cas où il y a beaucoup d'entiers inférieurs à n premiers avec n.

11.3.4 Générateur aléatoire de chemins NW-convexes

Pour tirer un chemin NW-convexe on utilise l'isomorphisme entre ceux-ci et les multi-ensembles de mots de Christoffel primitifs. Le multi-ensemble non-étiqueté est un constructeur classique pour lequel il existe un générateur de Boltzmann (voir section 3.3.3 ou [FFP07, BFP10]). La complexité du tirage est linéaire, les pré-calculs necessaires s'effectuent toujours en quelques minutes sur un ordinateur personnel.

Une fois que l'on a tiré un multi-ensemble de couples d'entiers premiers entre eux par ce générateur de Boltzmann, on le transforme en un chemin NW-convexe codé sur $\{0,1\}$. La procédure peut ainsi se décrire :

- Tirer m un multi-ensemble de $\textsc{MSet}(PC)$ par un générateur de Boltzmann,
- Trier les éléments (p,q) de m en ordre décroissant de q/p,
- Réécrire chacun de ces couples comme le segment discret de pente q/p codé sur $\{0,1\}^*$,
- Ajouter 1 au début du mot ainsi obtenu.

La complexité en temps de cette procédure est $O(n\ln(n))$, où n est la longueur du mot obtenu, à cause du tri de la deuxième étape.

11.3.5 Complexité de la génération d'un chemin NW-convexe en taille approchée et exacte

Les sections précédentes réunissent tous les éléments nécessaires à déterminer la complexité du générateur de Boltzmann de chemins NW-convexes. Les deux théorèmes suivants traitent respectivement de la complexité du tirage dans la cas d'un générateur en taille approchée et d'un générateur en taille exacte.

Théorème 11.17. *Un générateur de Boltzmann en taille approchée pour les chemins NW-convexes réussit en un essai avec probabilité tendant vers* 1 *quand n tend vers l'infini.*
Le coût global moyen de la génération d'un chemin NW-convexe en taille approchée est

$$O(n\ln(n))$$

en moyenne.

Théorème 11.18. *Un algorithme de Boltzmann de chemins NW-convexes en taille exacte réussit en un nombre d'essais moyen tendant vers*

$$\kappa \cdot n^{2/3}$$

où

$$\kappa = \frac{\sqrt[6]{2}\sqrt[3]{3}\pi^{5/6}}{\sqrt[6]{\zeta(3)}} \approx 4.075517917...$$

quand n tend vers l'infini.
Le coût global de la génération est $O(n^{5/3}\ln(\ln(n)))$ en moyenne.

Remarque. Comme $\varphi(n)$ croît lentement, le paramètre x_n réglé pour tirer de gros objets sera proche de 1, ce qui conduit à des multiplicités importantes dans le multi-ensemble. Une conséquence est que l'on a besoin de calculer la série génératrice des mots de Christoffel primitifs que jusqu'à un ordre relativement petit pour obtenir de bonnes approximations pour nos probabilités. En revanche, la série du multi-ensemble sera évaluée en de nombreux points.

11.3.6 Générateur aléatoire de PDC

Nous venons de montrer comment générer uniformément et indépendamment les uns des autres des chemins NW-convexes. On veut maintenant obtenir un PDC complet en recollant quatre chemins NW-convexes. Cependant, un quadruplet quelconque de chemins NW-convexes indépendants ne donne pas forcément le contour d'un polyomino. Pour que cela soit le cas, on a besoin de satisfaire la condition suivante : les quatre chemins NW-convexes (une fois tournés du bon angle et mis les uns à la suite des autres) doivent former un chemin fermé. Comme les chemins sont NW-convexes, on sait que le chemin ainsi obtenu ne se croisera pas lui-même et ne contiendra pas de demi-tours. L'algorithme 19 permet de générer un quadruplet de chemins NW-convexes vérifiant cette condition.

Algorithm 19: Γ_P

Input: un paramètre x
Output: un quadruplet de chemins NW-convexes formant un PCD

1 **Répéter** :
2 Tirer WN, NE, ES, SW quatre chemins NW-convexes en utilisant des appels indépendants à un générateur de Boltzmann de paramètre
3 **Si** $|WN|_0 + |NE|_1 = |ES|_0 + |SW|_1$ et $|NE|_0 + |ES|_1 = |SW|_0 + |WN|_1$
4 **alors return** (WN, NE, ES, SW)

Il s'agit simplement d'un algorithme naïf qui tire et rejette des quadruplets de chemins NW-convexes tant qu'ils ne satisfont pas cette condition. Il est évident que l'on obtient ainsi un générateur qui tire deux PDC de même taille avec la même probabilité [2]. Plus précisément, pour être fermé un chemin doit contenir autant de pas horizontaux dans sa partie supérieure de W à E (constituée de deux chemins NW-convexes, dont un pivoté d'un quart de tour) que dans sa partie inférieure de E à W (constituée de deux chemins NW-convexes, pivotés d'un demi et de trois quart de tour) et autant de pas verticaux dans sa partie gauche de S à N (constituée des deux chemins NW-convexes pivotés de trois quart de tour et pas pivoté) que dans sa partie droite de N à S (constituée des deux chemins NW-convexes pivotés d'un demi et de trois quart de tour).

2. En fait, on obtient même un générateur de Boltzmann, car la probabilité est proportionnelle à x^n, où n est la taille du PDC tiré.

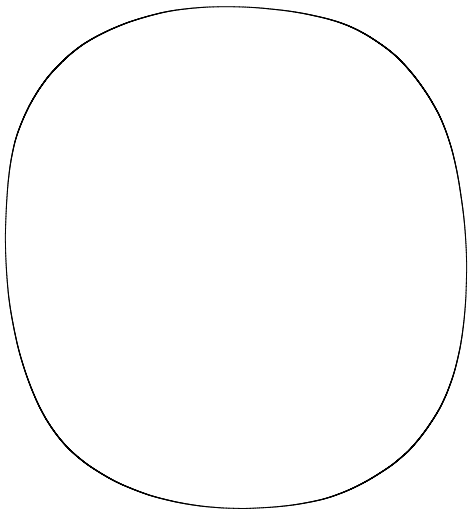


FIGURE 11.6 – Un polyomino de périmètre 81109 tiré avec un paramètre de Boltzmann $x = 0.98$.

Conclusion et perspectives

Nous avons présenté dans ce chapitre une méthode pour tirer aléatoirement uniformément des polyominos digitalement convexes. Notre approche est basée sur des générateurs de Boltzmann nous permettant de tirer de grands PDC (quelques dizaines de milliers en quelques dizaines de minutes sur un ordinateur personnel). La recherche d'un algorithme plus efficace (comme il a été développé dans le cas des produits de Hadamard [BGR10]) reste une question à explorer. On se trouve en effet dans un cas plus complexe que pour les produits de Hadamard en une variable, et une adaptation directe de la méthode présentée dans [BGR10] donne une générateur biaisé.

Notre générateur aléatoire nous a permis de mettre en évidence que les PDC aléatoires admettent une forme limite quand leur taille tend vers l'infini. Nous avons prouvé dans ce chapitre la forme limite des chemins NW-convexes, qui semble être également celle de la partie W à N des PDC, *i.e.*, la contrainte de clôture ne semble pas affecter la forme limite. Les outils nécessaires à l'étude de la forme limite des PDC nous semblent pour le moment hors d'atteinte. Même la question plus simple de l'énumération asymptotique précise des polyominos digitalement convexes (l'ordre de grandeur étant prouvé dans [IKŽ94]) est actuellement un véritable défi. Nous concluons en soulignant que notre travail peut probablement être étendu à des dimensions supérieures, mais un important travail reste à faire ne serait-ce que pour trouver une spécification.

Index

Bibliographie

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1974. AHO a 74 :1 1.Ex.

[Alo94] L. Alonso. Uniform generation of a motzkin word. *Theoret. Comput. Sci*, 134-2 :529–536, 1994.

[Ari08] H. Arimura. Efficient algorithms for mining frequent and closed patterns from semi-structured data. In T. Washio, E. Suzuki, K. Ming Ting, and A. Inokuchi, editors, *PAKDD*, volume 5012 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2008.

[ARS97] L. Alonso, J. L. Rémy, and R. Schott. A linear-time algorithm for the generation of trees, 1997.

[BBJ13] A. Bacher, O. Bodini, and A. Jacquot. Exact-size sampling for motzkin trees in linear time via boltzmann samplers and holonomic specification. In *ANALCO*, pages 52–61, 2013.

[BBJ14] A. Bacher, O. Bodini, and A. Jacquot. Efficient random sampling of binary and unary-binary trees via holonomic equations. *ArXiv e-prints*, January 2014.

[BBM11] A. Bacher and M. Bousquet-Mélou. Weakly directed self-avoiding walks. *J. Comb. Theory Ser. A*, 118(8) :2365–2391, November 2011.

[BBP+12] F. Bassino, M. Bouvel, A. Pierrot, C. Pivoteau, and D. Rossin. Combinatorial specification of permutation classes. In *Proceedings of FPSAC 2012 (24th International Conference on Formal Power Series and Algebraic Combinatorics)*, volume AR, pages 781 – 792, Nagoya, Japon, 2012.

[BC78] E. A. Bender and E. R. Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3) :296 – 307, 1978.

[BDJM13] O. Bodini, Ph. Duchon, A. Jacquot, and L. Mutafchiev. Asymptotic analysis and random sampling of digitally convex polyominoes. Discrete Geometry for Computer Imagery (DGCI), 2013.

[BDM14] O. Bodini, J. David, and Ph. Marchal. Random-bit optimal uniform sampling for rooted planar trees with given sequence of degrees and applications. 2014.

[BDN08] F. Bassino, J. David, and C. Nicaud. Enumeration and random generation of possibly incomplete deterministic automata. *Pure Mathematics and Applications*, 19(2-3) :1–16, 2008.

[Ben74] E. A. Bender. Asymptotic methods in enumeration. *SIAM*, 16(4) :485–515, 1974.

[Bet14] J. Bettinelli. Increasing forests and quadrangulations via a bijective approach. *J. Combin. Theory Ser. A*, 122(0) :107–125, 2014.

[BFKV07] M. Bodirsky, E. Fusy, M. Kang, and S. Vigerske. An unbiased pointing operator for unlabeled structures, with applications to counting and sampling. In *SODA '07 : Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 356–365, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[BFP10] O. Bodini, E. Fusy, and C. Pivoteau. Random sampling of plane partitions. *Comb. Probab. Comput.*, 19(2) :201–226, 2010.

[BFPW13] A. Bernini, L. Ferrari, R. Pinzani, and J. West. Pattern-avoiding Dyck paths. In *Proceedings of FPSAC 2013 (25th International Conference on Formal Power Series and Algebraic Combinatorics)*, Paris, France, 2013.

[BFSS01] C. Banderier, P. Flajolet, G. Schaeffer, and M. Soria. Random maps, coalescing saddles, singularity analysis, and Airy phenomena. *Random Structures and Algorithms*, 19(3-4) :194–246, 2001.

[BG14] O. Bodini and B. Gittenberger. On the asymptotic number of BCK(2)-terms. In *ANALCO, workshop on ANALytic COmbinatorics*, Portland (USA), January 2014.

[BGG11] O. Bodini, D. Gardy, and B. Gittenberger. Lambda-terms of bounded unary height. In *ANALCO, workshop on ANALytic COmbinatorics*, San Francisco (USA), January 2011.

[BGGJ13] O. Bodini, D. Gardy, B. Gittenberger, and A. Jacquot. Enumeration of generalized bci lambda-terms. Technical report, 2013.

[BGJ10] O. Bodini, D. Gardy, and A. Jacquot. Asymptotics and random sampling for formulae in intuitionist logical systems. In *Generation Aleatoire de Structure Combinatoire (Gascom'10), Montréal, Canada*, 2010.

[BGJ13] O. Bodini, D. Gardy, and A. Jacquot. Asymptotics and random sampling for BCI and BCK lambda. *Theoretical Computer Science*, 502 :227–238, 2013.

[BGR10] O. Bodini, D. Gardy, and O. Roussel. Boys-and-girls birthdays and Hadamard products. In *Proceedings of Lattice Paths and Applications'10*, page 5, 2010.

[BJ08] O. Bodini and A. Jacquot. Boltzmann samplers for colored combinatorial objects. In *Generation Aleatoire de Structure Combinatoire (Gascom'08), Bibbiena, Italy.*, 2008.

[BJ13] O. Bodini and A. Jacquot. Boltzmann samplers for v-balanced cycles. *Theor. Comput. Sci.*, 502 :55–63, 2013.

[BL12] O. Bodini and J. Lumbroso. Dirichlet random samplers for multiplicative structures. In *Proceedings of the Ninth Meeting on Analytic Algorithmics and Combinatorics (ANALCO'12)*, pages 92–106. SIAM, January 2012.

[BLL98] F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial Species and Tree-like Structures*. Cambridge University Press, 1998.

[BMW13] M. Bousquet-Mélou and K. Weller. Asymptotic properties of some minor-closed classes of graphs. In *25th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2013)*, 2013.

[BN07] F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1-3) :86–104, August 2007.

[Bod11] O. Bodini. *Autour de la génération aléatoire sous modèle de Boltzmann*. Habilitation à diriger des recherches, UPMC, 2011.

[BP10] O. Bodini and Y. Ponty. Multi-dimensional Boltzmann sampling of languages. *DMTCS Proceedings*, 0(01) :49–64, 2010.

[BPS92] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of underdiagonal walks. In Pierre Leroux and Christophe Reutenauer, editors, *Proceedings of 4th Conference on Formal Power Series and Algebraic Combinatorics (FPSAC'92)*. Université du Québec à Montréal, 1992.

[Bri50] N. A. Brigham. On a certain weighted partition function. *Proc. Amer. Math. Soc.*, 1 :192–204, 1950.

[Brl] Lyndon + Christoffel = digitally convex. *Pattern Recognition*, 42(10) :2239–2246.

[BRS12] O. Bodini, O. Roussel, and M. Soria. Boltzmann samplers for first order combinatorial differential equations. In *Discrete Applied Mathematics*, 2012.

[BS96] E. Bach and J. Shallit. *Algorithmic number theory. Vol. 1*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1996. Efficient algorithms.

[CD09] B. Canou and A. Darrasse. Fast and sound random generation for automated testing and benchmarking in objective caml. In *Proceedings of the 2009 ACM SIGPLAN workshop on ML*, pages 61–70. ACM, 2009.

[CDJ] G. Collet, J. David, and A. Jacquot. Random sampling of ordered trees according to the number of occurences of a pattern. *En cours de rédaction*.

[CDKK08] F. Chyzak, M. Drmota, T. Klausner, and G. Kok. The distribution of patterns in random trees. *Combinatorics, Probability & Computing*, 17(1) :21–59, 2008.

[CFGN10] G. Chapuy, É. Fusy, O. Giménez, and M. Noy. On the diameter of random planar graphs. In *21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10), DMTCS Proceedings*, pages 65–78, 2010.

[CH03] R. Cole and R. Hariharan. Tree pattern matching to subset matching in linear time. *SIAM J. Comput.*, 32(4) :1056–1066, 2003.

[Chu36] A. Church. An Unsolvable Problem of Elementary Number Theory. *Amer. J. Math.*, 58(2) :345–363, 1936.

[CLRC94] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and X. Cazin. *Introduction à l'algorithmique*. Science informatique. Dunod, 1994.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[CMT01] D. Coeurjolly, S. Miguet, and L. Tougne. Discrete curvature based on osculating circle estimation. In *Proceedings of the 4th International Workshop on Visual Form*, IWVF-4, pages 303–312, London, UK, UK, 2001. Springer-Verlag.

[Com74] L. Comtet. *Advanced combinatorics*. D. Reidel Publishing Co., Dordrecht, enlarged edition, 1974.

[CR02] M. Crochemore and W. Rytter. *Jewels of stringology*. World Scientific, 2002.

[Dar08] A. Darrasse. Random XML sampling the Boltzmann way. *Arxiv preprint arXiv :0807.0992*, 2008.

[DBRK71] N. G. De Bruijn, S. O. Rice, and D. E. Knuth. Average height of planted plane trees. Technical Report STAN-CS-71-218, Stanford University (Stanford,CA US), 1971.

[Dev86] L. Devroye. *Non-Uniform Random Variate Generation(originally published with*. Springer-Verlag, 1986.

[DFLS04] Ph. Duchon, Ph. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13 :577–625, 2004.

[DGK+10] R. David, K. Grygiel, J. Kozik, Ch. Raffalli, G. Theyssier, and M. Zaionc. Asymptotically almost all λ-terms are strongly normalizing, 2010. Preprint : arXiv :math.LO/0903.5505v3.

[DP02] S. Dulucq and J.-G. Penaud. Interpretation bijective d'une recurrence des nombres de motzkin. *Discrete Math.*, 256(3) :671–676, October 2002.

[DPRS10a] A. Darasse, K. Panagiotou, O. Roussel, and M. Soria. Biased Boltzmann samplers and generation of extended linear languages with shuffle. In *Proceedings of the 23rd International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'12)*, pages 125–140, 2010.

[DPRS10b] A. Darasse, K. Panagiotou, O. Roussel, and M. Soria. Boltzmann sampler for the shuffle product. In *Proceedings of Gascom'10*, 2010.

[DPT10] A. Denise, Y. Ponty, and M. Termier. Controlled non uniform random generation of decomposable structures. *CoRR*, abs/1006.0423, 2010.

[Duc11] Ph. Duchon. Random generation of combinatorial structures : Boltzmann samplers and beyond. In S. Jain, Roy R. Creasey Jr., Jan Himmelspach, K. Preston White, and Michael C. Fu, editors, *Winter Simulation Conference*, pages 120–132. WSC, 2011.

[Edm60] J. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices Amer. Math. Soc.*, 7, 1960.

[FFP07] Ph. Flajolet, E. Fusy, and C. Pivoteau. Boltzmann sampling of unlabelled structures. In SIAM Press, editor, *Proceedings of ANALCO'07 (Analytic Combinatorics and Algorithms) Conference*, volume 126, pages 201–211, 2007.

[FGGZ07] H. Fournier, D. Gardy, A. Genitrini, and M. Zaionc. Classical and intuitionistic logic are asymptotically identical. In *16th Annual Conference on Computer Science Logic (EACSL), volume 4646 of Lecture Notes in Computer Science*, pages 177–193, 2007.

[FL01] Ph. Flajolet and G. Louchard. Analytic variations on the Airy distribution. *Algorithmica*, 31(3) :361–377, 2001.

[FMJ09] T. Flouri, B Melichar, and J. Janousek. Subtree matching by deterministic pushdown automata. In *IMCSIT'09*, volume 4, pages 659–666. IEEE Computer Society Press, 2009.

[For06] D. J. Ford. *Probabilities on Cladograms : Introduction to the Alpha Model*. Stanford University Dept. of Mathematics, 2006.

[FPS11] Ph. Flajolet, M. Pelletier, and M. Soria. On Buffon machines and numbers. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 172–183. SIAM, 2011.

[FPV98] Ph. Flajolet, P.V. Poblete, and A. Viola. On the analysis of linear probing hashing. *Algorithmica*, 22 :490–515, 1998.

[FS09a] Ph. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[FS09b] Ph. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[FSZ91] Ph. Flajolet, B. Salvy, and P. Zimmermann. Automatic average-case analysis of algorithm. *Theoretical Computer Science*, 79(1) :37–109, 1991.

[Fus05] E. Fusy. Quadratic exact-size and linear approximate-size generation of planar graphs. In *International Conference on Analysis of Algorithms*, volume AD, pages 125–138, 2005.

[FZC94] Ph. Flajolet, P. Zimmerman, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1-2) :1–35, 1994.

[GBN10] D. Gouyou-Beauchamps and C. Nicaud. Random generation using binomialapproximations. AofA '10, pages 359–372, 2010.

[GK09] A. Genitrini and J. Kozik. Quantitative comparison of intuitionistic and classical logics - full propositional system. In *International Symposium on Logical Foundations of Computer Science (LFCS'09), volume 5407 of Lecture Notes in Computer Science*, pages 280–294, Florida, USA, January 2009.

[GL12] K. Grygiel and P. Lescanne. Counting and generating lambda terms, 2012. Preprint : arXiv :math.LO/1210.2610v1.

[Hin93] J. Roger Hindley. BCK and BCI logics, condensed detachment and the 2-property. *Notre Dame J. Formal Logic*, 34(2) :231–250, 1993.

[HO82] Christoph M. Hoffmann and Michael J. O'Donnell. Pattern matching in trees. *J. ACM*, 29(1) :68–95, January 1982.

[II65a] Y. Imai and K. Iséki. Corrections to : "On axiom systems of propositional calculi. I". *Proc. Japan Acad.*, 41 :669, 1965.

[II65b] Y. Imai and K. Iséki. On axiom systems of propositional calculi I. *Proc. Japan Acad.*, 41 :436–439, 1965.

[IKŽ94] A. Ivić, J. Koplowitz, and J. D. Žunić. On the number of digital convex polygons inscribed into an (m, m)-grid. *IEEE Transactions on Information Theory*, 40(5) :1681–1686, 1994.

[IT79] K. Iséki and Sh. Tanaka. An introduction to the theory of BCK-algebras. *Math. Japon.*, 23(1) :1–26, 1978/79.

[Ivi99] A. Ivić. On the multiplicity of zeros of the zeta-function. *Acad. Serbe des Sciences et Arts, Bull. CXVIII Class. des Sci. Math. et naturelles, Sci. Math., No. 24, Belgrade*, pages 119–132, 1999. available at arXiv :math/0501434vl [math.NT], January, 2005.

[Ivi03] A. Ivić. *The Riemann zeta-function : theory and applications*. Dover Books on Mathematics. Dover, 2003.

[Joy81] A. Joyal. Une théorie combinatoire des séries formelles. *Adv. Math.*, 42 :1–82, 1981.

[Kle35a] S C. Kleene. A Theory of Positive Integers in Formal Logic. Part I. *Amer. J. Math.*, 57(1) :153–173, 1935.

[Kle35b] S C. Kleene. A Theory of Positive Integers in Formal Logic. Part II. *Amer. J. Math.*, 57(2) :219–244, 1935.

[KLN08] B. Kerautret, J.-O. Lachaud, and B. Naegel. Comparison of discrete curvature estimators and application to corner detection. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, ISVC '08, pages 710–719, Berlin, Heidelberg, 2008. Springer-Verlag.

[Knu97a] D. E. Knuth. *The Art of Computer Programming : Fundamental Algorithms*, volume 1. Addison Wesley, Reading, Massachusetts, 3rd edition, July 1997.

[Knu97b] D. E. Knuth. *The art of computer programming, volume 2 (3rd ed.) : seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[LBB+13] Ph. Langlois, E. Beffara, J-D Boissonnat, S. Boldo, F. Chazal, E. Gioan, M. Martel, G. Melquiond, Ch. Paul, J. Ramì rez Alfonsì n, L. Vaux, and M. Yvinec. *Informatique Mathématique une photographie en 2013*. Presses Universitaires de Perpignan, 2013.

[Lum13] J. Lumbroso. Optimal discrete uniform generation from coin flips, and applications. *CoRR*, abs/1304.1916, 2013.

[MÖ99] E. Mäkinen. Generating random binary trees - a survey. *Inf. Sci.*, 115(1-4) :123–136, April 1999.

[MDBS09] A. Mougenot, A. Darrasse, X. Blanc, and M. Soria. Uniform random generation of huge metamodel instances. *Model Driven Architecture-Foundations and Applications*, pages 130–145, 2009.

[Mis05] M. Mishna. *A Holonomic Systems Approach to Algebraic Combinatorics*. Publications du Laboratoire de combinatoire et d'informatique mathématique. Université du Québec à Montréal, 2005.

[MTZ00] M. Moczurad, J. Tyszkiewicz, and M. Zaionc. Statistical properties of simple types. *Mathematical Structures in Computer Science*, 10(5) :575–594, 2000.

[NW78] A. Nijenhuis and H. S. Wilf. *Combinatorial algorithms*. Academic Press, 1978.

[Pie13] A. Pierrot. *Combinatoire et algorithmique dans les classes de permutations*. Thèse, Université Paris Diderot, 2013.

[Piv08] C. Piveauteau. *Génération aléatoire de structures combinatoires : méthode de Boltzmann effective*. Thèse, UPMC, 2008.

[PRA11] G. Polya, R. C. Read, and D. Aeppli. *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*. Springer London, Limited, 2011.

[PSS08] C. Pivoteau, B. Salvy, and M. Soria. Boltzmann oracle for combinatorial systems. In *Algorithms, Trees, Combinatorics and Probabilities*, pages 475–488. Discrete Mathematics and Theoretical Computer Science, 2008. Proceedings of the Fifth Colloquium on Mathematics and Computer Science. Blaubeuren, Germany. September 22-26, 2008.

[PSS12] C. Pivoteau, B. Salvy, and M. Soria. Algorithms for combinatorial structures : Well-founded systems and newton iterations. *Journal of Combinatorial Theory, Series A*, 119(8) :1711 – 1773, 2012. Available at http ://fr.arxiv.org/abs/1109.2688.

[PTW09] G. Pólya, R. E. Tarjan, and D. R. Woods. *Notes on Introductory Combinatorics*. Modern Birkhäuser Classics. Birkhäuser Boston, 2009.

[PW09] J. Pitman and M. Winkel. Regenerative tree growth : Binary self-similar continuum random trees and poisson–dirichlet compositions. *The Annals of Probability*, 37(5) :1999–2041, 09 2009.

[Rem85] J. L. Remy. Un procédé itératif de dénombrement d'arbres binaires et son application a leur génération aléatoire. *ITA*, 19(2) :179–195, 1985.

[Rou12] O. Roussel. *Génération aléatoire de structures ordonnées par le modèle de Boltzmann*. Thèse, UPMC, 2012.

[RS09] O. Roussel and M. Soria. Boltzmann sampling of ordered structures. *Electronic Notes in Discrete Mathematics*, 35 :305–310, 2009.

[SF83] J. Steyaert and Ph. Flajolet. Patterns and pattern-matching in trees : An analysis. *Information and Control*, 58(1–3) :19 – 58, 1983.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423, 623–656, July, October 1948.

[Slo] N. J. A Sloane. On-line encyclopedia of integer sequences. publié sous forme électronique à http ://oeis.org/.

[SZ94] B. Salvy and P. Zimmermann. Gfun : a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2) :163–177, 1994.

[Tar85] R. E. Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2) :306–318, 1985.

[Tit86] E. C. Titchmarsh. *The Theory of the Riemann Zeta-Function*. The Clarendon Press, Oxford University Press, 1986. New York.

[Tut01] W. T. Tutte. *Graph Theory*. Cambridge Mathematical Library. Cambridge University Press, 2001.

[Vid10] S. A. Vidal. An optimal algorithm to generate rooted trivalent diagrams and rooted triangular maps. *Theoretical Computer Science*, 411 :2945–2967, 2010.

[VM10] S. A. Vidal and M.Petitot. Counting rooted and unrooted triangular maps. *Journal of Non-linear Systems and Applications*, pages 51–57, 2010.

[Yan00] Y. Yang. Partitions into primes. *Trans. Amer. Math. Soc.*, 352 :2581–2600, 2000.

[Zei91] D. Zeilberger. The method of creative telescoping. *J. Symbolic Comput.*, 11(3) :195–204, 1991.

Résumé

La combinatoire analytique est un domaine qui consiste à appliquer des méthodes issues de l'analyse complexe à des *classes combinatoires* afin d'obtenir des résultats sur leurs propriétés asymptotiques. On utilise pour cela des *spécifications*, qui sont une manière de formaliser la structure (souvent récursive) des objets.

Dans cette thèse, nous nous attachons principalement à trouver des nouvelles spécifications pour certaines classes combinatoires, afin de pouvoir ensuite y appliquer des méthodes efficaces d'énumération ou de génération aléatoire. En effet, pour une même classe combinatoire il peut exister différentes spécifications, basées sur des décompositions différentes, rendant les méthodes classiques d'énumération asymptotique et de génération aléatoire plus ou moins adaptées. Le premier volet de résultats présentés concerne l'algorithme de Rémy et la spécification holonome qui y est sous-jacente, basée sur un opérateur de *greffe*. On y développe un nouvel algorithme, plus efficace, de génération aléatoire d'arbres binaires et un générateur aléatoire d'arbres de Motzkin basé sur le même principe. Nous abordons ensuite des questions relatives à l'étude de sous-classes de λ-termes. Enfin, nous présentons deux autres ensembles de résultats, sur la spécification automatique d'arbres où les occurrences d'un motif donné sont marquées et sur le comportement asymptotique et la génération aléatoire de polyominos digitalement convexes. Dans tous les cas, les nouvelles spécifications obtenues donnent accès à des méthodes qui ne pouvaient pas être utilisées jusque là et nous permettent d'obtenir de nombreux nouveaux résultats.

Abstract

Analytic combinatorics is a field which consist in applying methods from complex analysis to *combinatorial classes* in order to obtain results on their asymptotic properties. We use for that *specifications*, which are a way to formalise the (often recursive) structure of the objects.

In this thesis, we mainly devote ourselves to find new specifications for some combinatorial classes, in order to then apply more effective enumerative or random sampling methods. Indeed, for one combinatorial class several different specifications, based on different decompositions, may exist, making the classical methods - of asymptotic enumeration or random sampling - more or less adapted. The first set of presented results focuses on Rémy's algorithm and its underlying holonomic specification, based on a *grafting* operator. We develop a new and more efficient random sampler of binary trees and a random sampler of Motzkin trees based on the same principle. We then address some question relative to the study of subclasses of λ-terms. Finally, we present two other sets of results, on automatic specification of trees where occurrences of a given pattern are marked and on the asymptotic behaviour and the random sampling of digitally convex polyominoes. In every case, the new specifications give access to methods which could not be applied previously and lead to numerous new results.