Ecole Doctorale d'Informatique, Télécommunications et Électronique de Paris

Génération aléatoire de structures combinatoires : méthode de Boltzmann effective.

THÈSE

présentée et soutenue publiquement le 3 décembre 2008

pour l'obtention du

Doctorat de l'université Pierre et Marie Curie – Paris VI

(spécialité informatique)

par

Carine Pivoteau

Composition du jury

Rapporteurs :	François Bergeron
	Alain Denise
Examinateurs :	Philippe Flajolet
	Patrick Gallinari
	Jacques Malenfant
	Conrado Martínez
	Bruno Salvy
	Michèle Soria (directrice)

Laboratoire d'Informatique de Paris VI

Table des matières

Remerciements

Au cours des trois années qu'a duré ma thèse, j'ai eu le plaisir de rencontrer un grand nombre de chercheurs, d'enseignants et d'étudiants qui ont influé sur mon travail, le plus souvent de façon imperceptible, mais parfois aussi, beaucoup plus radicalement. C'est à eux que vont mes plus grands remerciements. Mon étourderie peut me faire oublier certains d'entre eux ici – je les prie de m'en excuser – mais cela ne diminue en rien ma gratitude.

Je dois avouer, tout d'abord, que j'ai eu l'immense chance d'être très bien encadrée mais aussi très entourée pendant toute la durée de ma thèse : jamais une de mes questions n'est restée sans réponse et jamais un de mes doutes ne m'a angoissée sans que personne ne vienne le faire envoler.

Je remercie, en premier lieu, ma directrice, Michèle Soria – qui suit ma progression depuis mon entrée à l'UPMC, il y a cinq ans – pour m'avoir offert l'opportunité de réaliser cette thèse. Les heures passées à travailler ensemble ont été particulièrement enrichissantes et j'ai bénéficié grandement de son expérience dans mon travail de recherche mais également en tant qu'enseignante. Elle a, par ailleurs, su trouver l'équilibre indispensable entre la liberté dont j'avais besoin et l'encadrement, l'attention et le soutien qui m'ont permis de mener à bien cette thèse. Elle a dirigé mon travail avec une grande justesse, une profonde humanité et beaucoup de gaieté, ce pour quoi je la remercie infiniment.

Je remercie chaleureusement Philippe Flajolet pour tout ce qu'il a fait pour moi depuis le jour où il a accepté de diriger mon stage de Master. J'ai eu le plaisir de pouvoir profiter de son immense savoir mais également de sa gentillesse et de sa bonne humeur perpétuelle. Je le remercie de m'avoir fait confiance, de m'avoir tant appris et de m'avoir permis de rencontrer les gens qui gravitent autour de lui. Je le remercie enfin de continuer à suivre mes travaux d'un œil toujours si bienveillant.

Bruno Salvy fait indéniablement partie des gens qui ont considérablement influencé ma façon de travailler. En me lançant sur la question de "l'oracle", il a donné une direction nouvelle et passionnante à mes recherches. Ce sujet a donné lieu à de nombreuses heures de travail en commun, durant lesquelles j'ai énormément appris à son contact. Je le remercie en particulier pour son enthousiasme quasi inébranlable, son infinie patience et toute la pédagogie dont il a su faire preuve pour répondre à mes questions, même les plus naïves.

Je remercie vivement mes rapporteurs, François Bergeron et Alain Denise, pour avoir accepté de bonne grâce et parfaitement accompli la tâche de rapporter ma thèse. Je remercie également Philippe Flajolet, Patrick Gallinari, Jacques Malenfant, Conrado Martínez et Bruno Salvy d'avoir accepté de prendre part à mon jury.

Depuis mon arrivée au LIP6 en 2005, différent changements se sont produits dans l'équipe à laquelle j'appartiens : elle s'appelait CALFOR à mon arrivée, puis a changé de nom pour SPIRAL et a finalement été restructurée récemment. J'appartiens désormais à l'équipe APR. Dans cet environnement constitué de gens venus d'horizons scientifiques divers, j'ai énormément appris et pu échanger des points de vue sur des sujets extrêmement variés. Je remercie toutes ces personnes pour l'ouverture d'esprit que leur compagnie a pu m'apporter. Je remercie en particulier les membres du groupe "Génération Aléatoire" : Michèle Soria, Maryse Pelletier, Olivier Bodini, Alexis Darrasse, Olivier Roussel, Xiaomin Wang, Alice Jacquot et Yann Ponty.

J'ai été accueillie à bras ouverts au projet ALGO (devenu le projet ALGORITHMS) à l'INRIA Rocquencourt dès le premier jour de mon stage de Master et régulièrement depuis le début de ma thèse. Je remercie tous les membres de ce projet pour cet accueil, pour les longues discussions de la salle café, qu'elles soient scientifiques ou non, pour le temps qu'ils m'ont

consacré, pour leurs conseils avisés et également pour la joie de vivre qu'ils communiquent. Je remercie notamment Frédéric Chyzak pour toujours trouver des solutions adaptées à mes problèmes, quels qu'ils soient, et Philippe Dumas pour les mini-cours de Maths accélérés. Je remercie enfin Virginie pour avoir continué à m'épauler pour toutes les petites tracasseries administratives.

J'adresse des remerciements groupés à l'ensemble de la communauté ALEA pour l'accueil sympathique qui m'a été fait et le plaisir que j'ai trouvé à évoluer en son sein. Je remercie en particulier les membres de l'ANR GAMMA dont les réunions sont toujours si réjouissantes et notamment ma "roommate" occasionnelle, Mathilde Bouvel.

J'ai eu le plaisir de collaborer avec Michèle, Philippe et Bruno, mais également avec Olivier Bodini, toujours plein de nouvelles idées surprenantes et intéressantes et Éric Fusy qui a su apporter des réponses claires et précises à toutes mes questions. Je les remercie pour m'avoir fait confiance et m'avoir impliquée dans leur travail.

J'adresse également des remerciements à Alexis Darrasse, Alin Bostan et Olivier Bodini pour avoir pris le temps de relire plusieurs chapitres de ma thèse : leur aide a été précieuse à un moment où j'en avais grandement besoin. Je remercie particulièrement Alin pour sa lecture méticuleuse et pour avoir si aimablement consacré du temps à faire (et refaire), dans l'urgence, des calculs qui me faisaient tant peur.

Ces trois années m'ont également offert ma première expérience d'enseignement. J'ai eu, pour faire mes premiers pas, la chance de bénéficier de nombreux conseils de mes collègues ; je pense, entre autres, à Michèle, Maryse, Olivier, Stef, Valérie, Anne, ... Je les remercie de m'avoir aidée à m'adapter à ces nouvelles responsabilités.

Je remercie enfin mes proches et amis pour leur soutien sans faille et bien sûr, Aurélien, pour être à mes côtés, tout simplement.

Résumé

La génération aléatoire uniforme est un problème central en combinatoire algorithmique. Dans le modèle de Boltzmann, les structures combinatoires sont engendrées avec une taille variable ce qui permet de concevoir des générateurs efficaces.

Cette thèse vise à rendre effective cette méthode de génération aléatoire pour un grand nombre de classes combinatoires décomposables, en automatisant l'ensemble des traitements intervenant dans la conception des générateurs de Boltzmann.

La première partie est consacrée à l'étude des algorithmes de génération. Nous complétons le dictionnaire initial des générateurs de Boltzmann afin de pouvoir traiter les classes combinatoires non étiquetées. Ces algorithmes génériques sont présentés en détails, validés par des preuves et illustrés sur des exemples classiques en combinatoire. Des données expérimentales viennent également souligner les performances de ces générateurs ; une application à la génération aléatoire de partitions planes complète cette étude.

Dans la méthode de Boltzmann, les générateurs sont paramétrés par une valeur numérique x qui contrôle l'espérance de la taille des structures engendrées. L'uniformité de la génération repose sur une fonction d'oracle qui associe à toute classe combinatoire la valeur de sa série génératrice en x. La seconde partie de ce mémoire présente une méthode de calcul automatique et efficace de cet oracle, par itération de Newton numérique. La validité de cette méthode repose sur la convergence de l'itération de Newton pour les structures combinatoires. Cette itération est ensuite relevée au niveau des séries formelles puis au niveau des valeurs numériques. L'itération sur les séries conduit par ailleurs à un algorithme de complexité quasi-optimale pour calculer les premiers coefficients des séries de dénombrement.

Abstract

Uniform random generation is a central issue in combinatorics. Under the Boltzmann model, combinatorial structures are generated with a randomly varying size, which allows for the design of particularly efficient samplers.

The aim of this thesis is to make this sampling method effective for a large number of decomposable combinatorial classes via an automatization of all the treatments involved in the design of the samplers.

The first part is dedicated to the study of sampling algorithms. We complete the initial dictionary of Boltzmann samplers in order to support unlabelled classes. Those generic algorithms are fully detailed, proved and illustrated by classical examples coming from combinatorics. Experimental data are given to emphasize the efficiency of those samplers. An application to the random generation of plane partitions concludes this study.

In the framework of Boltzmann method, the samplers are parametrized with a numerical value that controls the expected size of the generated structures. The samplers uniformity relies on an oracle function that returns, for any combinatorial class, the value of its generating function at a given point. The second part of this thesis introduces a method to automatically and efficiently compute this oracle, using a numerical Newton iteration. The validity of this approach is based on the convergence of Newton iteration on combinatorial structures. This iteration is then lifted to the level of generating series and finally to numerical values. In addition, the iteration on series leads to a quasi-optimal algorithm to compute the first terms of the counting series.

Introduction

Introduction

La génération aléatoire est un problème central en combinatoire algorithmique. Ses applications sont nombreuses en informatique, mais aussi dans d'autres domaines tels que la physique ou la biologie. Que ce soit pour tester la robustesse d'un programme, vérifier l'adéquation d'un modèle ou confirmer une conjecture, les générateurs aléatoires permettent d'effectuer des simulations afin d'obtenir des données statistiques sur des structures abstraites. L'objet de ces simulations est principalement d'engendrer des structures suffisamment grandes pour rendre observables leurs propriétés limites, d'où la nécessité de concevoir des générateurs performants.

La méthode de Boltzmann [DFLS04] est une méthode générique pour engendrer aléatoirement des structures appartenant à des classes combinatoires décomposables. Elle repose sur un modèle théorique qui ne fixe pas la taille des structures à engendrer, ce qui permet de concevoir des générateurs uniformes en taille approchée dont la complexité est essentiellement linéaire.

Dans [DFLS04], les auteurs proposent un dictionnaire de générateurs pour les classes étiquetées qui peuvent être décrites récursivement à partir de constructions combinatoires classiques. Nous étendons tout d'abord ce dictionnaire aux constructions non étiquetées, en proposant un jeu d'algorithmes génériques et efficaces pour engendrer les structures qu'elles définissent. Nous présentons, par ailleurs, une application de cette méthode à la génération aléatoire de partitions planes ; les simulations effectuées permettent alors d'observer leur forme limite.

L'uniformité d'un générateur de Boltzmann repose sur une fonction d'oracle capable d'évaluer numériquement des séries génératrices en un point donné. Pour rendre effective la méthode de Boltzmann nous proposons un schéma itératif générique qui systématise le calcul de l'oracle. L'idée est d'utiliser une itération de Newton numérique pour l'évaluation des systèmes de séries génératrices implicites. La validité de cette approche repose essentiellement sur la convergence de l'itération de Newton au niveau des structures combinatoires elles-mêmes.

Ainsi, cette thèse contribue à enrichir le modèle de Boltzmann théorique proposé initialement et à rendre effective cette méthode de génération aléatoire pour un grand nombre de classes combinatoires décomposables, en automatisant l'ensemble des traitements liés à la conception des générateurs et en la mettant en application sur des problèmes concrets.

Méthodes

Les structures combinatoires étudiées dans ce mémoire sont des objets munis d'une fonction de taille, tels que les mots, les permutations, les arbres ou encore les graphes. On peut les décrire,

éventuellement récursivement, à partir de structures similaires plus petites ou de structures "plus simples", en utilisant un jeu de constructions combinatoires classiques (union disjointe, produit cartésien, séquence, ensemble et cycle). Ce cadre de travail symbolique, présenté de façon unifiée dans [FS08], permet de définir un grand nombre de classes combinatoires. Dans cet univers de classes *décomposables*, on dispose d'un ensemble d'outils génériques pour décrire et dénombrer les structures. On associe, en particulier, à toute classe combinatoire $\mathcal{C}$ une *série génératrice* $C(z)$, dont le coefficient $c_n = [z^n]C(z)$ énumère les structures de taille n. Ces séries jouent un rôle essentiel dans la façon d'appréhender les traitements liés aux structures combinatoires.

Engendrer aléatoirement et uniformément une structure à l'intérieur d'une classe combinatoire consiste, étant donné une distribution de probabilité uniforme sur les structures de cette classe, à choisir l'une d'elles de façon non déterministe, suivant cette distribution. Bien qu'il soit possible de faire des tirages aléatoires suivant d'autres distributions, nous ne considérerons, dans ce mémoire, que le cas uniforme. Classiquement, la génération aléatoire uniforme dans une classe combinatoire $\mathcal{C}$ porte sur un modèle à taille fixée : pour un entier n positif, on effectue des tirages uniformes dans la sous-classe $\mathcal{C}_n$ des structures de taille n (et de cardinal $c_n < \infty$) ; on attribue donc à toute structure de taille n une probabilité $1/c_n$.

Le *modèle de Boltzmann*, introduit par Duchon, Flajolet, Louchard et Schaeffer [DFLS04], diffère du modèle à taille fixée, au sens où il place une distribution de probabilité sur la totalité de la classe $\mathcal{C}$ et attribue aux structures un poids proportionnel à une exponentielle de leur taille ; ainsi la taille des structures engendrées varie, mais l'uniformité à taille donnée est conservée.

Le modèle de Boltzmann *de paramètre x assigne à toute structure γ appartenant à la classe combinatoire $\mathcal{C}$ la probabilité suivante :*

$$\mathbb{P}_x(\gamma) = \frac{x^{|\gamma|}}{C(x)}, \qquad \textit{avec} \quad C(x) = \sum_{\delta \in \mathcal{C}} x^{|\delta|}.$$

Un générateur de Boltzmann $\Gamma\mathcal{C}(x)$ pour $\mathcal{C}$ est un algorithme qui engendre des $\mathcal{C}$-structures suivant une distribution conforme à ce modèle.

Dans ce modèle, la taille des structures devient une variable aléatoire et un générateur de Boltzmann peut, a priori, engendrer des structures de n'importe quelle taille. En pratique, de trop grandes fluctuations ne sont pas souhaitables. Néanmoins, de nombreuses applications qui utilisent des générateurs aléatoires peuvent tolérer de légères variations autour de la taille visée. En écartant les structures trop petites ou trop grandes, on peut contrôler ces variations et obtenir des *générateurs en taille approchée*. Cette flexibilité sur la taille des structures est le principal atout des générateurs de Boltzmann : avec une tolérance de dix pour cent, il est possible d'engendrer, en quelques secondes, des structures dont la taille est de l'ordre de plusieurs millions.

La *méthode* de Boltzmann est la mise en œuvre de ce modèle théorique dans le cadre symbolique de [FS08]. Se placer dans ce cadre présente l'énorme avantage d'avoir automatiquement accès aux séries génératrices associées à ces classes combinatoires décomposables, sous la forme de systèmes d'équations implicites. De la même façon qu'on associe à chaque construction combinatoire une opération sur les séries génératrices, on peut concevoir systématiquement, pour chacune de ces constructions, le générateur de Boltzmann qui lui est associé. Dans [DFLS04], Duchon *et al.* définissent les méthodes génériques applicables à toute spécification combinatoire étiquetée (les atomes constituant les structures sont tous distincts).

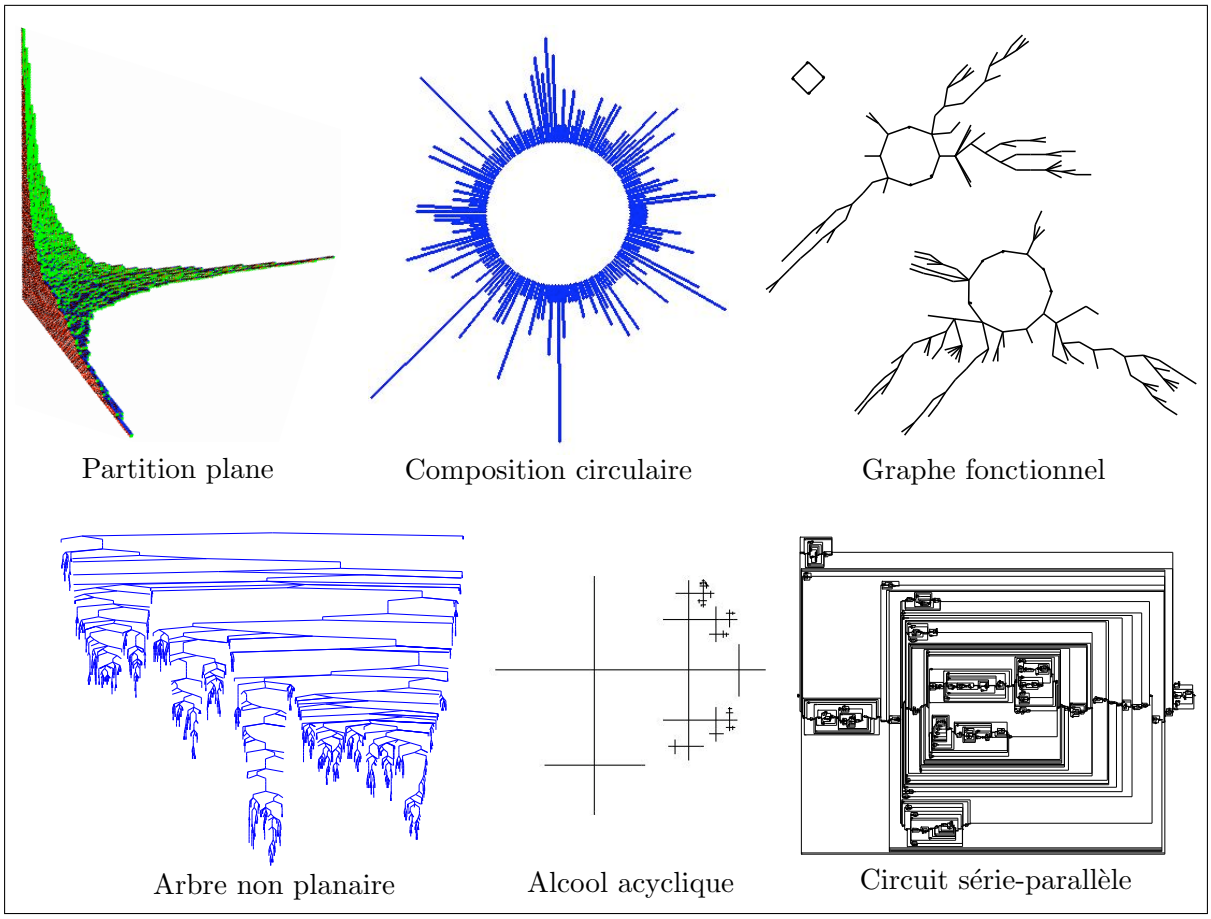


FIG. 1 – Exemples de structures générées.

La première partie de ce mémoire est consacrée à l'étude des générateurs de Boltzmann pour les structures non étiquetées. Ces structures peuvent présenter des symétries qui les rendent naturellement plus difficiles à engendrer de façon uniforme. Les expressions des séries génératrices associées aux spécifications non étiquetées induisent une décomposition des structures qui permet de prendre en compte ces symétries pour l'énumération. Nous montrons que les générateurs de Boltzmann peuvent être conçus en suivant cette même décomposition. Nous présentons en détails les algorithmes génériques pour chaque construction combinatoire non étiquetée, en nous assurant de leur validité et en les illustrant par des exemples classiques (la figure 1 représente différentes structures engendrées). Des données expérimentales viennent également souligner les performances de ces générateurs et une application à la génération aléatoire de partitions planes complète cette étude.

L'uniformité des tirages effectués par un générateur de Boltzmann repose sur une évaluation numérique des séries génératrices. En supposant que ce calcul numérique est dévolu à une boîte noire – l'*oracle de Boltzmann* – les algorithmes obtenus en assemblant les générateurs associés aux constructions de base ont une complexité linéaire en la taille de leur sortie ; ils permettent d'engendrer sans difficulté des structures de plusieurs centaines de milliers de nœuds, ce qui n'était pas envisageable avec les générateurs à taille fixée. C'est essentiellement la tolérance sur la taille des structures qui permet d'améliorer la complexité des générateurs, et ce, de

deux façons. D'une part, la phase de génération proprement dite est accélérée car les choix probabilistes faits par les algorithmes sont plus simples lorsque que la taille n'est pas fixée. D'autre part, il n'est pas nécessaire de calculer les développements des séries génératrices pour obtenir une génération uniforme.

Pour utiliser un générateur de Boltzmann, il ne suffit pas de fixer la taille visée : il faut lui fournir un paramètre "bien choisi" qui contrôle l'espérance de la taille des structures à engendrer, ainsi qu'un oracle, cette fonction externe au générateur et capable d'évaluer numériquement les séries génératrices pour une valeur cohérente du paramètre. Ce dernier point est crucial pour la réalisation des générateurs et une automatisation complète de la méthode de Boltzmann passe par un calcul automatique de l'oracle. La deuxième partie de ce mémoire a donc pour objectif de rendre transparente la conception de cette boîte noire.

Dans de rares cas, les séries génératrices impliquées ont une forme explicite qui rend triviale leur évaluation numérique. En revanche, dans le cas général, non seulement les systèmes implicites qui décrivent les séries n'ont pas de forme close mais ils peuvent également avoir plusieurs solutions réelles en un même point. Si l'on prend, par exemple, la spécification suivante, à laquelle on peut directement associer un système d'équations de séries génératrices $\boldsymbol{C}(z)$:

$$\begin{cases} \mathcal{C}_0 = \mathcal{Z}\mathcal{C}_1\mathcal{C}_2\mathcal{C}_3(\mathcal{C}_1 + \mathcal{C}_2) \\ \mathcal{C}_1 = \mathcal{Z} + \mathcal{Z}\textsc{Seq}(\mathcal{C}_1^2\mathcal{C}_3^2) \\ \mathcal{C}_2 = \mathcal{Z} + \mathcal{Z}^2\textsc{Seq}(\mathcal{Z}\mathcal{C}_2^2\textsc{Seq}(\mathcal{Z}))\textsc{Seq}(\mathcal{C}_2) \\ \mathcal{C}_3 = \mathcal{Z} + \mathcal{Z}(3\mathcal{Z} + \mathcal{Z}^2 + \mathcal{Z}^2\mathcal{C}_1\mathcal{C}_3)\textsc{Seq}(\mathcal{C}_1^2) \end{cases} \Rightarrow \begin{cases} C_0(z) = zC_1(z)C_2(z)C_3(z)(C_1(z) + C_2(z)) \\ C_1(z) = z + \frac{z}{1-C_1(z)^2C_3(z)^2} \\ C_2(z) = z + \frac{z^2}{(1-zC_2(z)^2/(1-z))(1-C_2(z))} \\ C_3(z) = z + \frac{z(3z+z^2+z^2C_1(z)C_3(z))}{1-C_1^2(z)} \end{cases}$$

et que l'on résout ce système pour $z = 0.27$, on obtient, pour C_0, les valeurs réelles positives représentées par la figure 2 (à gauche). Or, s'il existe une unique classe combinatoire $\mathcal{C}_0$ – nous verrons par la suite que c'est le cas – alors on lui associe une unique série génératrice $C_0(z)$ qui ne peut prendre qu'une unique valeur en $z = 0.27$. Comment distinguer la "bonne" solution (en rouge sur le dessin), celle qui correspond à l'évaluation de la série génératrice de $\mathcal{C}_0$:

$$C_0(z) = 18z^5 + 90z^6 + 222z^7 + 1032z^8 + 4446z^9 + 23184z^{10} + 126492z^{11} + 732264z^{12} + \ldots,$$

des autres solutions du système ? Il est possible d'avoir une idée plus "globale" des solutions de ce système, en le résolvant pour différentes valeurs de z. On obtient les courbes de la figure 2 (à droite). Toutes les courbes croissantes (on peut éliminer les autres, puisque les séries génératrices sont à coefficients positifs) sont alors des solutions qui correspondent potentiellement à l'oracle que l'on cherche, mais rien ne permet, a priori, de déterminer quelle est la courbe qui correspond effectivement à la série $C_0(z)$ (également en rouge), parmi toutes les candidates.

Cet exemple montre que fournir un oracle pour une spécification combinatoire ne se réduit pas à la résolution numérique d'un système d'équations fonctionnelles. Aborder le problème de cette façon, c'est oublier la nature combinatoire des équations manipulées et perdre ainsi une grande quantité d'information sur la solution recherchée. L'approche que nous avons adoptée est fondée sur le constat que dans l'univers combinatoire – et sous de "bonnes hypothèses" – les systèmes fonctionnels utilisés pour décrire les structures n'admettent qu'une unique solution. La preuve de ce résultat dû à Joyal [Joy81] repose sur une *itération combinatoire* qui converge vers la solution du système. En propageant cette itération au niveau des séries formelles, on obtient l'unique vecteur de séries génératrices associées aux classes définies par une spécification combinatoire. Enfin cette itération sur les séries est elle-même relevée au niveau des valeurs numériques pour obtenir l'oracle de Boltzmann. La convergence à chaque niveau est alors une

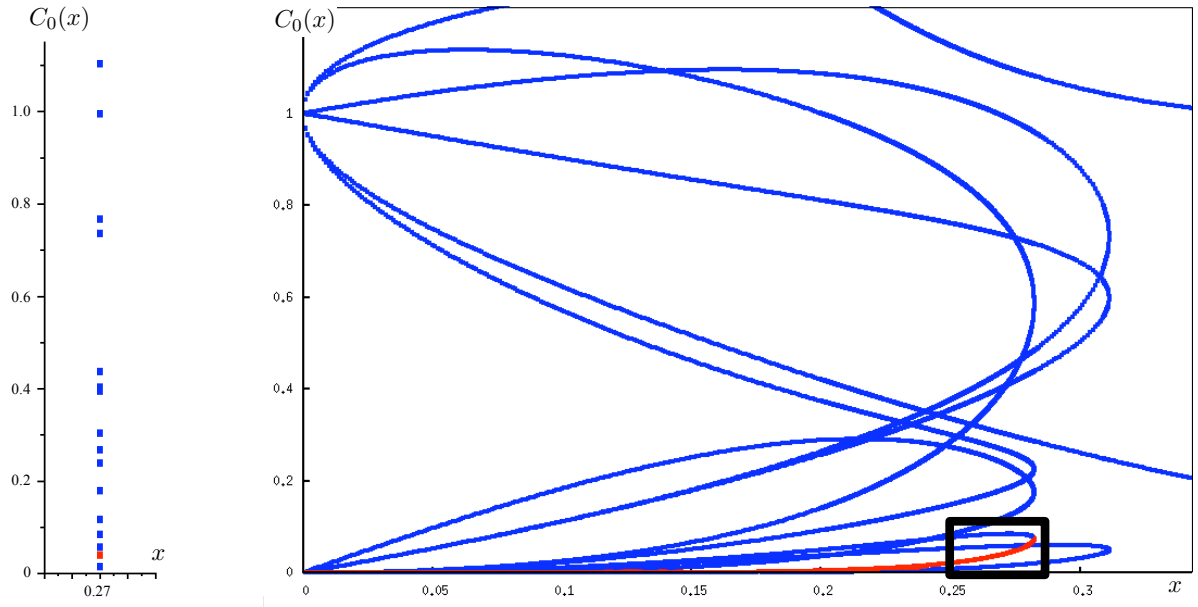


FIG. 2 – Solutions du système $\mathbf{C}(z)$ pour la variable $C_0(z)$ en fonction de z.

conséquence de la convergence au niveau précédent et chaque valeur renvoyée par l'itération numérique correspond à l'évaluation d'une série génératrice dont les coefficients énumèrent des structures combinatoires ; ainsi, au lieu de résoudre un système d'équation en aveugle, l'itération trouve directement son chemin vers l'unique solution pertinente d'un point de vue combinatoire.

De la même façon que les générateurs de Boltzmann sont construits automatiquement à partir des spécifications combinatoires, ce schéma itératif numérique générique systématise le calcul de l'oracle. Supposons que l'on veuille, par exemple, concevoir un oracle pour la classe des arbres généraux non planaires, définie par l'équation combinatoire suivante :

$$\mathcal{T} = \mathcal{Z} \times \textsc{Set}(\mathcal{T}).$$

Cette définition récursive peut être interprétée comme une équation de point fixe sur les structures combinatoires pour une notion de distance adéquate, le *contact*. L'itération :

$$\mathcal{T}^{[n+1]} = \mathcal{Z} \times \textsc{Set}(\mathcal{T}^{[n]}), \text{ avec } \quad \mathcal{T}^{[0]} = 0,$$

converge alors vers $\mathcal{T}$; elle est directement traduite en une itération sur les séries :

$$T^{[n+1]}(z) = z \exp\big(T^{[n]}(z)\big), \text{ avec } \quad T^{[0]}(z) = 0,$$

qui converge vers la série génératrice $T(z) = -W(-z)$ des arbres non planaires ($W(z)$ est la fonction W de Lambert). Enfin, pour tout complexe α tel que $|\alpha| < e^{-1}$, l'itération numérique correspondante :

$$t^{[n+1]} = \alpha \exp(t^{[n]}), \text{ avec } \quad t^{[0]} = 0,$$

converge vers la valeur $T(\alpha) = -W(-\alpha)$. En effet, chaque valeur $t^{[n]}$ est égale à la valeur $T^{[n]}(\alpha)$ de $T(z)$ au point α. La convergence de cette série en α est aussi une conséquence de la nature combinatoire de l'équation.

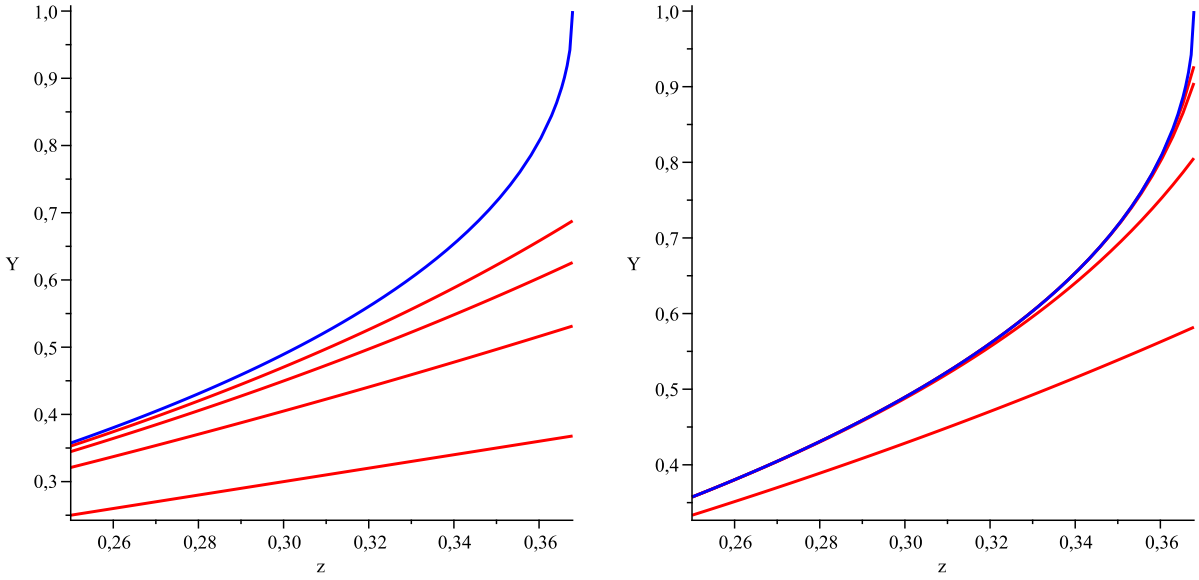


FIG. 3 – Itérations (simple à gauche, et de Newton à droite) sur les séries formelles des arbres généraux non planaires.

La convergence de l'itération de point fixe est typiquement linéaire ; un oracle plus rapide peut être obtenu par *itération de Newton*, avec une convergence quadratique :

$$t^{[n+1]} = t^{[n]} + (\alpha \exp(t^{[n]}) - t^{[n]})/(1 - \alpha \exp(t^{[n]})).$$

Classiquement, cette itération converge vers l'une des racines de l'équation $t = \alpha \exp(t)$, en supposant que le point de départ est choisi convenablement. Ici, et ceci reste vrai pour n'importe quel système combinatoire, en choisissant $t^{[0]} = 0$, l'itération converge vers la racine $T(\alpha)$; la suite des itérées est positive et croissante de sorte qu'à chaque étape, on se rapproche de la solution. Ce phénomène est illustré par la figure 3. Dans les deux cas, la courbe bleue est la série $T(z) = -W(-z)$. Les courbes rouges sont les séries successives obtenues par itération simple (à gauche) et par itération de Newton (à droite). On peut observer qu'elles se rapprochent progressivement de la courbe bleue, plus rapidement pour l'itération de Newton.

Là encore, aussi bien le schéma numérique que celui sur les séries sont basés sur une itération purement combinatoire qui, cette fois-ci, converge quadratiquement vers la solution. Comme précédemment, cette itération est alors transférée au niveau des séries génératrices, et la convergence quadratique est conservée en termes de valuation. Plus qu'une simple étape vers l'oracle numérique, cette itération sur les séries se révèle être une méthode efficace de calcul des coefficients des séries génératrices. Et enfin, la version numérique de l'itération nous fournit un oracle efficace pour les générateurs de Boltzmann.

Contribution et plan détaillé

Introduction Le premier chapitre est une introduction aux techniques de génération aléatoire de structures combinatoires. Le but de ce chapitre n'est pas de donner un historique complet sur ce sujet, mais plutôt de poser les bases sur lesquelles repose la méthode de Boltzmann. Outre un

La complexité (en temps) d'un algorithme peut être mesurée de différentes façons. On pourra s'intéresser, dans un premier temps, à la complexité *arithmétique* qui correspond au nombre d'opérations arithmétiques utilisées par l'algorithme. Dans le cas d'un algorithme de génération aléatoire, un appel à la fonction Random() est considéré comme une opération arithmétique. On distinguera en général les opérations sur des entiers de celles sur les réels.

Quand les nombres manipulés sont de "grands" entiers, comme a^n ou $n!$ (leurs codages binaires sont en $O(n)$ et $O(n \log n)$), on préfère compter le nombre d'opérations effectuées sur les bits. On parle alors de complexité *binaire*.

On dira qu'un algorithme est *quasi-optimal* lorsque sa complexité est de l'ordre de la taille du résultat, à des facteurs logarithmiques près.

FIG. 4 – Complexités.

certain nombre de méthodes *ad hoc* qui peuvent être associées à des générateurs génériques, nous présentons de façon détaillée la méthode récursive de [FZVC94], d'une part comme prétexte à l'introduction de la méthode symbolique de [FS08] qui donne le cadre de travail et les outils que nous utiliserons tout au long de ce mémoire et d'autre part parce qu'elle est, en quelques sortes, l'ancêtre de la méthode de Boltzmann. Enfin, nous consacrons une dernière section au modèle de Boltzmann proprement dit et redonnons, en substance, les résultats de [DFLS04].

Générateurs de Boltzmann non étiquetés La première partie de ce mémoire est entièrement consacrée aux générateurs de Boltzmann non étiquetés.

Dans le chapitre 2, nous donnons une collection d'algorithmes génériques pour les générateurs de Boltzmann correspondant aux constructions de Pólya, c'est à dire les ensembles, multi-ensembles et cycles non étiquetés. Ces constructions, contrairement à celles traitées par Duchon *et al.* dans [DFLS04], donnent lieu à des structures pouvant présenter des symétries. Cette particularité rend légèrement plus délicate la conception de générateurs uniformes, étant donné qu'il faut parvenir à prendre en compte ces symétries. Nous complétons ce dictionnaire avec des constructions portant des contraintes sur leur nombre de composantes.

Ces algorithmes, associés à ceux présentés dans [DFLS04], constituent les "briques de base" permettant d'assembler des générateurs de Boltzmann pour toute classe combinatoire de structures décomposables. Les générateurs ainsi obtenus ont une complexité arithmétique[1] linéaire en la taille des structures engendrées. L'algorithme permettant d'engendrer des ensembles sans répétitions entraîne quant à lui un surcoût qui est le plus souvent constant par rapport à la taille de la sortie.

Nous ponctuons ce chapitre d'exemples de générateurs pour des structures classiques de la combinatoire et concluons avec quelques résultats expérimentaux permettant notamment d'observer les distributions des tailles obtenues ainsi que l'efficacité de la génération.

Ce chapitre reprend les résultats exposés dans l'article *"Boltzmann Sampling of Unlabelled Structures"* [FFP07], écrit en collaboration avec Philippe Flajolet et Éric Fusy.

Nous présentons, au chapitre 3, une application des résultats du chapitre précédent avec un générateur aléatoire de partitions planes. En combinant une bijection due à Pak [Pak02] et nos générateurs de Boltzmann non étiquetés, nous obtenons un générateur de complexité arithmétique quasi-linéaire en taille approchée – plus précisément, $O(n \log^3 n)$, où n est la taille

[1] L'encadré de la figure 4 donne quelques rappels sur les différentes notions de complexité.

visée. En taille exacte, la complexité devient $O(n^{4/3})$. À notre connaissance, c'est le premier algorithme polynomial pour engendrer des partitions planes uniformes par rapport à leur taille (il existe d'autres générateurs polynomiaux, mais pour des partitions uniformes à l'intérieur d'une boîte). Le même principe donne lieu à des générateurs de partitions encadrées (des partitions planes dont deux dimensions sont bornées), et de partitions tordues (*skew*). Nous avons implanté ces générateurs aléatoires et nous avons pu effectuer des simulations jusqu'à des tailles de l'ordre de 10^7 qui nous ont permis d'observer des formes limites et des frontières gelées (c'est le même phénomène que pour les pavages d'un hexagone par des losanges) qui ont récemment été analysées par Cerf et Kenyon [CK01] pour les partitions planes et par Okounkov et Reshetikhin [OR07] pour les partitions tordues.

Ce chapitre reprend les résultats de l'article *"Random sampling of plane partitions"* [BFP06] (et sa version longue [BFP07]), écrit en collaboration avec Olivier Bodini et Éric Fusy.

Calcul de l'oracle La seconde partie de ce mémoire a pour but de présenter notre méthode de calcul de l'oracle de Boltzmann. Les chapitres 4 et 5 présentent les fondements combinatoires de l'oracle. Les deux chapitres suivants établissent le transfert de convergence qui s'opère des structures combinatoires vers les séries formelles et les oracles numériques qui en découlent.

Nous changeons de cadre de travail pour cette partie en oubliant, pour un temps, l'univers des classes spécifiables pour la théorie des espèces de structures combinatoires. Plusieurs raisons motivent ce choix. L'un des points clés de notre approche est que le théorème des espèces implicites de Joyal [Joy81] nous garantit l'existence et l'unicité d'une solution pour les systèmes combinatoires que nous manipulons. Ainsi, on peut montrer que les itérations combinatoires que nous étudions convergent vers l'unique solution du système. C'est l'unicité de cette solution combinatoire qui nous assure que la solution numérique obtenue par itération est bien la valeur attendue comme oracle de Boltzmann. Un autre avantage de ce cadre de travail est qu'il offre un niveau de généricité supplémentaire, au sens où les espèces sont définies indépendamment des constructions combinatoires utilisées pour les décrire ; on pourra donc, par la suite, étendre plus facilement nos résultats à de nouvelles classes de structures.

Pour ces raisons, nous entamons notre étude par un chapitre rappelant quelques éléments de la théorie des espèces de structures [Joy81, BLL98] dont nous aurons besoin pour la suite. Nous redonnons également une preuve du théorème des espèces implicites de Joyal, basée sur une itération de point fixe combinatoire. Ce faisant, nous introduisons quelques propriétés sur les suites d'espèces définies par itération, que nous réutiliserons pour prouver la convergence des itérations de Newton au chapitre suivant.

Le but du chapitre 5 est d'introduire une itération combinatoire plus rapide que l'itération de point fixe (dont la convergence est typiquement linéaire) utilisée pour prouver le théorème des espèces implicites. L'idée d'une itération de Newton pour résoudre une équation combinatoire apparaît dans [DLL82, BLL98]. Nous étendons cette idée à l'ensemble des systèmes combinatoires définissant des familles d'espèces de structures. Pour ce faire, nous faisons appel aux éclosions combinatoires de Labelle [Lab85] afin de donner un sens à l'inverse de la matrice jacobienne qui apparaît dans l'itération de Newton combinatoire, à l'instar de l'itération sur les séries formelles. Nous présentons également une version optimisée de cette itération en utilisant l'idée classique que ce même inverse peut également être obtenu par itération de Newton et le fait qu'il n'est pas nécessaire de le recalculer dans son intégralité à chaque étape. Les deux itérations de Newton (classique et optimisée) ont une vitesse de convergence quadratique, mais la seconde est plus rapide, car elle engendre beaucoup moins de structures inutiles.

Un soin particulier est apporté aux preuves de convergence de ces itérations. Les propriétés

de l'itération de point fixe que nous avons isolées au chapitre précédent restent vraies pour l'itération de Newton et seront transmises au niveau des séries génératrices, ce qui nous permettra de valider notre oracle numérique. Afin d'illustrer nos résultats, nous présentons deux exemples d'espèces "calculées" par itération : les arbres généraux planaires et les circuits série-parallèle.

Au chapitre 6, nous établissons la première partie du transfert de convergence : la convergence des itérations combinatoires se traduit au niveau des séries génératrices par une convergence en termes de valuation. Ce résultat repose sur le fait que les coefficients des séries génératrices obtenues à chaque étape d'une itération énumèrent exactement les structures produites par l'itération combinatoire correspondante. C'est par l'intermédiaire de la convergence pour les séries formelles que nous pourrons ensuite démontrer la convergence numérique. Nous concluons ce chapitre par une étude de la complexité des itérations de Newton pour calculer les premiers coefficients des séries génératrices. Pour cette étude, nous nous replaçons dans le cadre des classes combinatoires décomposables utilisé dans la première partie de ce mémoire, afin de fixer l'ensemble des opérations autorisées sur les séries. Nous montrons alors que notre méthode par itération de Newton permet de calculer ces coefficients avec une complexité (aussi bien arithmétique que binaire) quasi-optimale, améliorant ainsi les algorithmes de génération récursive présentés au chapitre 1.

Le principal résultat du chapitre 7 est un oracle automatique pour les générateurs de Boltzmann sous la forme d'une itération de Newton numérique qui, partant du vecteur nul, converge de façon inconditionnelle vers un vecteur de valeurs numériques qui correspond à l'évaluation des séries génératrices en un point donné. Nous établissons donc la seconde partie du transfert de convergence, des séries formelles vers l'itération numérique. Nous ne donnons pas ici de preuve de la convergence numérique pour des spécifications combinatoires comportant des opérateurs de Pólya, mais nous produisons un exemple d'oracle, pour les graphes série-parallèles non étiquetés, qui indique comment étendre nos résultats à l'ensemble des classes combinatoires spécifiables. Nous concluons sur la présentation de quelques résultats expérimentaux obtenus avec un prototype d'oracle et permettant d'apprécier l'efficacité de notre méthode : en quelques secondes, on obtient, pour des systèmes de plusieurs centaines d'équations, les valeurs des séries donnant lieu à des générateurs de Boltzmann permettant d'engendrer des structures composées de millions de nœuds.

Cette seconde partie reprend et étend les résultats présentés dans l'article *"Boltzmann Oracle for Combinatorial Systems"* [PSSar], écrit en collaboration avec Bruno Salvy et Michèle Soria. Une version longue de cet article est en préparation [PSS08], incluant notamment la preuve de convergence numérique pour les spécifications comportant des opérateurs de Pólya.

Chapitre 1

Génération aléatoire de structures combinatoires

Sommaire

En donnant un bref aperçu des méthodes de génération aléatoire existant dans la littérature, ce chapitre a pour but d'expliquer quelles sont les origines et les principes fondamentaux sur lesquels repose le modèle de Boltzmann. La connaissance de ces techniques nous permettra par la suite de mieux comprendre quelles en sont les spécificités et d'en juger la pertinence. Néanmoins, pour des raisons évidentes, ce chapitre ne donne pas un panorama exhaustif des méthodes de génération aléatoire, et nous laisserons de côté un certain nombre de paradigmes intéressants (génération exhaustive [Knu06, BDLPP99], génération pondérée [Dev86, DRT00],...) et de méthodes efficaces (génération par chaînes de Markov [SJ89, Jer94],...), pour nous intéresser plus particulièrement aux méthodes reliées au modèle de Boltzmann.

La première section est consacrée à quelques exemples classiques d'algorithmes dits *ad hoc*, chacun étant dédié à une classe combinatoire spécifique. En réalité, on peut aisément dégager quelques principes récurrents applicables plus généralement à de nombreuses classes combinatoires. En découlent un certain nombre de techniques pouvant se "greffer" aux algorithmes génériques présentés ensuite, afin d'en améliorer l'efficacité. La section suivante présente la méthode récursive dans ses grandes lignes. Initiée, entre autres, par Nijenhuis et Wilf dans [NW78], cette méthode est basée sur la possibilité de décomposer récursivement les structures que l'on cherche à engendrer. L'uniformité des tirages repose alors sur l'utilisation des coefficients des séries de dénombrement des classes combinatoires considérées. En se plaçant

dans le cadre des structures décomposables, on dispose d'un dictionnaire de traduction automatique des spécifications combinatoires en séries génératrices et donc, en générateurs récursifs.

La dernière section introduit le modèle de Boltzmann de Duchon, Flajolet, Louchard et Schaeffer [DFLS04].

1 Algorithmes *ad hoc*

Cette première section est dédiée à l'étude de plusieurs algorithmes classiques de génération aléatoire[2] que l'on peut regrouper sous l'appellation *ad hoc*, au sens où, contrairement aux méthodes que nous verrons ensuite, ils ont été conçus pour s'appliquer spécifiquement à une classe combinatoire donnée. Cette spécialisation permet notamment de tirer parti de propriétés particulières à chaque classe et conduit, le plus souvent, à un algorithme de génération linéaire en la taille des structures produites. Parmi les nombreux algorithmes *ad hoc* existant dans la littérature, nous avons choisi quelques exemples qui illustrent en particulier deux techniques que nous réutiliserons par la suite : la *génération surjective* ou *bijective* et la méthode du *rejet*.

1.1 Génération surjective, bijective.

La génération surjective consiste à utiliser des relations isomorphiques entre différentes classes combinatoires afin de ramener le problème de la génération aléatoire uniforme dans une classe donnée à un problème plus simple, dans une classe équivalente, à condition que le passage d'une classe à l'autre préserve l'uniformité.

Proposition 1 (Génération surjective)**.** *Soient $\mathcal{B}$ et $\mathcal{C}$ des classes combinatoires, et φ une application surjective de $\mathcal{B}$ dans $\mathcal{C}$ telle que toutes les $\mathcal{C}$-structures ont le même nombre d'antécédents :*

$$\forall \gamma \in \mathcal{C}, \quad |\{\beta \mid \beta \in \mathcal{B} \ \ et \ \ \varphi(\beta) = \gamma\}| = \frac{|\mathcal{B}|}{|\mathcal{C}|}.$$

Si $\mathrm{Gen}\,\mathcal{B}$ *est un générateur aléatoire uniforme de $\mathcal{B}$-structures, alors le processus* $\mathrm{Gen}\,\mathcal{C}$ *qui fait appel à* $\mathrm{Gen}\,\mathcal{B}$ *pour engendrer une $\mathcal{B}$-structure β et renvoie $\varphi(\beta)$ est un générateur aléatoire uniforme de $\mathcal{C}$-structures.*

Démonstration. L'uniformité du générateur $\mathrm{Gen}\,\mathcal{C}$ est obtenue comme une conséquence directe de l'équidistribution des antécédents des $\mathcal{C}$-structures dans $\mathcal{B}$. Si γ est une $\mathcal{C}$-structure quelconque, la probabilité d'engendrer γ avec $\mathrm{Gen}\,\mathcal{C}$ est :

$$\mathbb{P}(\gamma) = \frac{1}{|\mathcal{B}|} \cdot \frac{|\mathcal{B}|}{|\mathcal{C}|} = \frac{1}{|\mathcal{C}|}. \qquad \square$$

La génération bijective est un cas particulier de génération surjective, lorsque $|\mathcal{B}| = |\mathcal{C}|$. Le premier algorithme que nous présentons utilise cette idée pour engendrer des arbres binaires de façon incrémentale : un arbre aléatoire se construit nœud après nœud, une surjection permettant de passer d'un arbre aléatoire de taille n à un arbre aléatoire de taille $n+1$. Le deuxième algorithme, pour engendrer des mots de Dyck, sépare la phase de génération de la transformation surjective. Nous verrons qu'il est possible de recycler cette idée en combinant n'importe quel algorithme obtenu par une méthode automatique à une bijection, pour les classes qui ne sont pas directement spécifiables. C'est notamment le cas de l'algorithme permettant d'engendrer des partitions planes que nous présenterons au chapitre 3.

[2] La présentation de ces algorithmes est inspirée des cours de Master donnés par R. Cori et D. Gouyou-Beauchamp. Les résumés de séminaires [GB93, GB03] forment une version condensée des notes de cours.

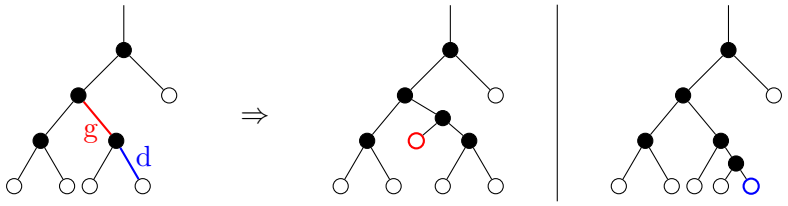


FIG. 1 – Relation entre les arbres binaires de taille n et ceux de taille $n+1$.

1.1.1 Algorithme incrémental de Rémy [Rem85]

Cet algorithme permet de générer efficacement des arbres binaires de façon *incrémentale*. L'idée est d'utiliser la relation suivante pour engendrer un arbre de taille $n+1$ à partir d'un arbre de taille n :

$$2(2n+1)c_n = (n+2)c_{n+1}. \tag{1}$$

Le nombre de Catalan $c_n = \frac{1}{n+1}\binom{2n}{n}$ est ici le nombre d'arbres binaires complets (*i.e.* tous les nœuds ont 0 ou 2 fils) de taille n. Ce sont des arbres ayant n nœuds internes et donc $n+1$ feuilles et $2n+1$ arêtes si l'on compte une arête virtuelle qui part de la racine. Une interprétation de la relation (1) consiste à choisir aléatoirement un arbre de taille n, puis l'une de ses $2n+1$ arêtes en lui attribuant une orientation (droite ou gauche). Il y a donc $2(2n+1)c_n$ choix possibles. On le transforme ensuite en un arbre de taille $n+1$ en greffant un nouveau nœud sur l'arête distinguée, à laquelle on ajoute une feuille (pointée) à gauche ou à droite selon l'orientation choisie. On obtient ainsi un arbre parmi les $(n+2)c_{n+1}$ arbres de taille $n+1$ dont l'une des $n+2$ feuilles est marquée. La figure 1 illustre cette bijection : à partir d'un arbre de taille 4, les deux choix d'une arête et d'une orientation (en rouge et en bleu) aboutissent à deux arbres de taille 5 identiques mais dont les feuilles marquées sont différentes.

On utilise cette relation pour concevoir un algorithme incrémental pour engendrer des arbres uniformément par rapport à leur taille. On commence avec un arbre réduit à une feuille, et à chaque étape on produit un arbre binaire ayant un nœud interne supplémentaire. À l'étape n, en ajoutant aléatoirement une feuille et une orientation à l'arbre de taille $n-1$ déjà obtenu, on fabrique un arbre aléatoire marqué ayant n nœuds internes. On peut obtenir un arbre aléatoire non marqué en *effaçant* la marque. En effet, de cette façon, tout arbre non marqué de taille n peut être obtenu à partir de $n+1$ arbres marqués différents. Par récurrence, chaque arbre de taille n est donc tiré avec la même probabilité $p_n = 1/c_n$:

$$p_n = \frac{p_{n-1}(n+1)}{2(2n-1)} = \frac{n+1}{2(2n-1)c_{n-1}} = \frac{1}{c_n}.$$

On obtient un algorithme qui, en n étapes, engendre un arbre aléatoire uniforme de taille n. Il a donc une complexité linéaire en la taille de l'arbre engendré.

1.1.2 Génération de mots de Dyck

Les mots de Dyck (ou encore mots de parenthèses) sont des mots sur l'alphabet à deux lettres $A = \{a, b\}$, engendrés par la grammaire

$$\mathcal{D} \to \mathcal{E} \mid a\mathcal{D}b\mathcal{D}.$$

Les mots de Dyck de taille $2n$, *i.e.*, ayant $2n$ lettres, sont en bijection avec les arbres binaires de taille n et donc, également énumérés par les nombres de Catalan. Cette bijection, associée au générateur de la section précédente, est un exemple d'algorithme bijectif pour engendrer des mots de Dyck. L'algorithme présenté maintenant est quant à lui un algorithme surjectif, mais non incrémental : en effet, dans un premier temps, on engendre un mot de A^* (avec des contraintes sur les occurrences des lettres) et ensuite on le transforme en mot de Dyck.

Génération d'un mot w de A^*, avec $|w|_a = p$ et $|w|_b = q$ Le générateur de mots de Dyck nécessite un générateur uniforme de mots de A^* composés de p lettres a et q lettres b (on note $\mathcal{L}_{pq}$ le langage de ces mots). Le nombre total de mots dans $\mathcal{L}_{pq}$ est le nombre de façons de choisir les p positions des a parmi les $p+q$ emplacements possibles, *i.e.*,

$$\binom{p+q}{p} = \frac{(p+q)!}{p!\,q!}.$$

En observant que le nombre de mots de $\mathcal{L}_{pq}$ commençant par la lettre a est $\binom{p+q-1}{p-1}$, on obtient la probabilité qu'un mot aléatoire uniforme dans $\mathcal{L}_{pq}$ commence par la lettre a :

$$\frac{(p+q-1)!\,p!}{(p-1)!\,(p+q)!} = \frac{p}{p+q}.$$

On en déduit un algorithme[3] récursif $\mathbf{Gen}\mathcal{L}(p,q)$ qui, pour engendrer un mot de $\mathcal{L}_{pq}$, commence par choisir la première lettre du mot et suivant le cas, le complète par un mot ayant un a ou un b de moins.

```
GenL(p, q)
  si p = 0 alors renvoyer b^q
  sinon
    si q = 0 alors renvoyer a^p
    sinon
      k ← random(p + q);
      si k < p alors renvoyer a.Gen_pq(p − 1, q)
      sinon renvoyer b.Gen_pq(p, q − 1)
```

Par récurrence, tout mot $w = w_0 w_1 \dots w_{p+q}$ de $\mathcal{L}_{pq}$ est engendré avec la probabilité $\mathbb{P}_{p,q}$:

$$\mathbb{P}_{p,q} = \begin{cases} \frac{p}{p+q}\mathbb{P}_{p-1,q} = \frac{p}{p+q}\frac{(p-1)!q!}{(p-1+q)!} = \frac{p!q!}{(p+q)!} & \text{si } w_0 = a \\ \frac{q}{p+q}\mathbb{P}_{p,q-1} = \frac{q}{p+q}\frac{p!(q-1)!}{(p+q-1)!} = \frac{p!q!}{(p+q)!} & \text{sinon.} \end{cases}$$

Ce principe de génération peut être étendu à des mots sur un alphabet k-aire, dont toutes les lettres ont des nombres d'occurrences fixés.

Génération surjective Un mot de Dyck de taille $2n$ est classiquement représenté par un chemin composé de n pas montants ($\nearrow$ pour les lettres a) et n pas descendants ($\searrow$ pour les lettres b), sur le quart de plan positif, partant de l'origine et s'arrêtant sur l'axe des abscisses (la figure 2 donne un exemple de chemin de Dyck de longueur 10). Cette représentation facilite

[3]Cet algorithme est équivalent à un générateur de combinaisons aléatoires de p éléments parmi $p+q$.

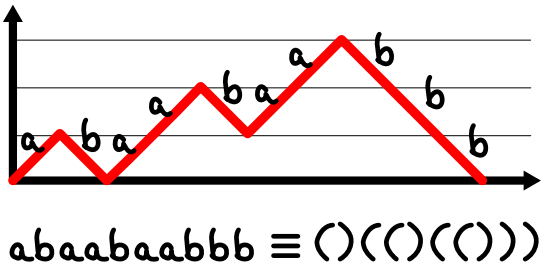


FIG. 2 – Chemin de Dyck de longueur 10 ainsi que le mot et le parenthésage correspondant.

l'interprétation de la relation suivante, qui va nous permettre de transformer un mot quelconque de $\mathcal{L}_{n(n+1)}$ en mot de Dyck :

$$(2n+1)c_n = \binom{2n+1}{n}.$$

Elle s'interprète grâce au lemme cyclique (ou lemme de Raney) [Lot83] :

Lemme 1 (Raney)**.** *Un mot sur l'alphabet $A = \{a, b\}$ composé de n lettres a et $n+1$ lettres b admet une unique factorisation $f = f'f''$ avec $f' \neq \varepsilon$ (parmi les $2n+1$ possibles) telle que $f''f'$ est un mot de Dyck à $2n$ lettres suivi d'un b.*

Un mot de $\mathcal{L}_{n(n+1)}$ se factorise en un unique mot de Dyck de taille n. Pour cela, il suffit de couper le mot au niveau du point le plus bas dans le chemin de Dyck correspondant. Inversement, on peut obtenir l'un des $2n+1$ mots de $\mathcal{L}_{n(n+1)}$ en rajoutant un b à la fin d'un mot de Dyck et en le factorisant suivant l'une des $2n+1$ coupures possibles (cette bijection est illustrée par la figure 3). Un générateur de mots de Dyck consiste donc simplement à tirer uniformément un mot de $\mathcal{L}_{n(n+1)}$ par l'algorithme $\mathbf{Gen}\mathcal{L}(n, n+1)$ et à le transformer en un mot de Dyck marqué (au niveau de la coupure) par cette bijection. Il suffit enfin d'effacer la marque pour obtenir un mot de Dyck de longueur $2n$. L'uniformité est conservée car chaque mot de Dyck de taille $2n$ peut être obtenu d'exactement $2n+1$ façons différentes en enlevant la marque.

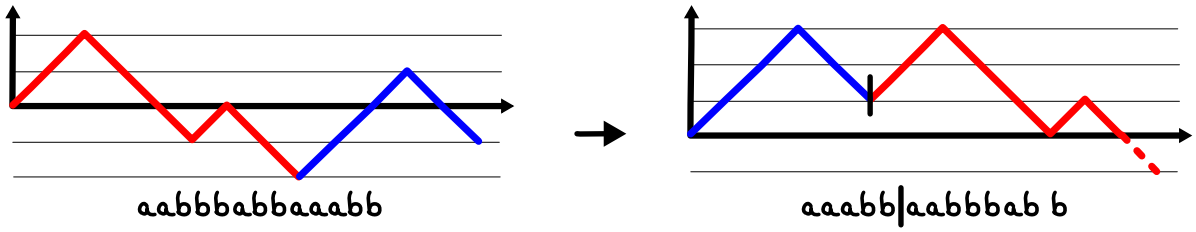


FIG. 3 – Factorisation de Raney sur un mot de taille 13.

1.2 Rejet et facteurs gauches de mots de Motzkin

Le *rejet* est une technique classique de génération aléatoire. L'idée est simple : pour tirer aléatoirement et uniformément des structures appartenant à une classe $\mathcal{C}$, on suppose que l'on sait tirer uniformément des structures dans une classe $\mathcal{B}$ qui contient $\mathcal{C}$. Il suffit alors de tirer aléatoirement des $\mathcal{B}$-structures, jusqu'à obtenir une $\mathcal{C}$-structure. Les éléments rejetés par cet algorithme n'influent pas sur l'uniformité du tirage dans $\mathcal{C}$, étant donné que les éléments de $\mathcal{C}$ sont tous tirés avec la même probabilité.

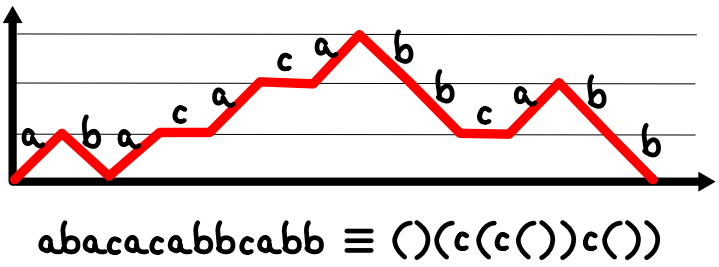


FIG. 4 – Chemin de Motzkin de longueur 13, ainsi que le mot et le parenthésage correspondant.

Proposition 2. *Soient $\mathcal{B}$ et $\mathcal{C}$ des classes combinatoires telles que $\mathcal{C} \subset \mathcal{B}$. Soit* $\operatorname{Gen}\mathcal{B}$ *un générateur aléatoire uniforme de $\mathcal{B}$-structures, alors le processus* $\operatorname{Gen}\mathcal{C}$ *qui fait appel à* $\operatorname{Gen}\mathcal{B}$ *pour engendrer une $\mathcal{B}$-structure, jusqu'à obtention d'une $\mathcal{C}$-structure, est un générateur aléatoire uniforme pour $\mathcal{C}$.*

Démonstration.

$$\mathbb{P}(\gamma \mid \gamma \in \mathcal{C}) = \frac{\mathbb{P}(\text{tirer } \gamma \text{ et } \gamma \in \mathcal{C})}{\mathbb{P}(\text{tirer un élément de } \mathcal{C})} = \frac{1}{|\mathcal{B}|} \cdot \frac{|\mathcal{B}|}{|\mathcal{C}|} = \frac{1}{|\mathcal{C}|}. \qquad \square$$

L'efficacité de la méthode dépend de la proportion des éléments appartenant à la classe visée dans le sur-ensemble dans lequel on fait les tirages. Elle sera d'autant plus efficace que le nombre moyen d'essais à faire avant d'obtenir une $\mathcal{B}$-structure est faible. La probabilité de devoir rejeter une structure est $1 - |\mathcal{C}|/|\mathcal{B}|$, donc le nombre d'essais moyen est $|\mathcal{B}|/|\mathcal{C}|$. Par exemple, on peut imaginer un algorithme qui, pour tirer des mots d'un langage $\mathcal{L}$ sur un alphabet A, se contente de tirer des mots de A^*, jusqu'à obtenir un mot de $\mathcal{L}$. Pour $A = \{a, b, c, d\}$, cet algorithme va engendrer un mot commençant par a au bout de $\frac{4^n}{4^{n-1}} = 4$ essais en moyenne, alors qu'il lui faudra environ $\frac{4^n}{4^{n/2}} = 2^n$ essais pour engendrer un mot miroir ($w = uu$).

Rejet anticipé Un moyen d'améliorer l'efficacité de cette méthode est d'anticiper le rejet en interrompant la génération lorsqu'il devient évident que la structure que l'on est en train d'engendrer n'appartiendra pas à la classe visée. Cette technique a notamment été utilisée par Barcucci *et al.* dans [BPS94], pour engendrer des facteurs gauches de mots de Motzkin, étape préalable pour engendrer des animaux dirigés. Les mots de Motzkin et leurs facteurs gauches sont décrits, sur l'alphabet $\{a, b, c\}$, par les grammaires :

$$\mathcal{M} \to \mathcal{E} \mid c\mathcal{M} \mid a\mathcal{M}b\mathcal{M}, \qquad \text{et} \qquad \mathcal{F} \to \mathcal{M} \mid \mathcal{M}a\mathcal{F}.$$

Tout comme les mots de Dyck, les mots de Motzkin peuvent être représentés par des chemins dans le quart de plan positif. La différence vient du fait que l'on peut, en plus des pas montants et descendants, faire des pas horizontaux ($\to$ pour les lettres c). Un chemin de Motzkin commence en $(0, 0)$ et se termine sur l'axe des abscisses. Un facteur gauche est n'importe quel chemin pouvant se compléter en un chemin de Motzkin. La figure 4 donne un exemple de mot de Motzkin de longueur 13 et le chemin correspondant.

L'idée de [BPS94] est d'engendrer les facteurs gauches par rejet, en tirant des mots dans $\{a, b, c\}^*$. On tire les mots lettre par lettre, chacune ayant une probabilité 1/3 d'apparaître. En partant du simple constat que tout préfixe d'un facteur gauche mot de Motzkin contient au moins autant de a que de b, Barcucci *et al.* proposent d'*anticiper* le rejet, en abandonnant

{1,2,3}	{1,4,5}
{1,2,4}	{2,3,4}
{1,2,5}	{2,3,5}
{1,3,4}	{2,4,5}
{1,3,5}	{3,4,5}

(a) NEXKSB(5, 3)

0000	1010	0101
1000	0010	1101
1100	0011	1001
0100	1011	0001
0110	1111	
1110	0111	

(b) NEXSUB(4)

$\sigma_1 \leftarrow 1$
pour i **de** 2 **à** n **faire**
  $\sigma_i \leftarrow i$
  $k \leftarrow random(i)$
    échanger σ_k et σ_i
renvoyer σ

(c) RANPER(n)

Fig. 5 – Combinatorial Algorithms [NW78].

un mot dès qu'il enfreint cette propriété. De façon plus imagée, cela revient à écarter tout chemin qui repasse sous l'axe des abscisses. En utilisant les méthodes d'analyse asymptotique de [FS08], on peut calculer le coût moyen de la génération d'un facteur gauche de longueur n, en additionnant le nombre de lettres tirées lors des échecs au nombre de lettres du mot engendré. On trouve qu'il faut tirer en moyenne $2n$ lettres pour engendrer un facteur de taille n. Cette méthode, connue sous le nom de rejet *florentin*, donne donc lieu à un algorithme linéaire pour engendrer des facteurs gauches de mots de Motzkin. Sans anticipation, le même algorithme engendrerait en moyenne $\sqrt{n}$ mots de taille n avant d'engendrer un facteur gauche appartenant à $\mathcal{F}$. Il aurait donc une complexité en $O(n\sqrt{n})$.

2 Méthodes automatiques

Bien qu'en général très performantes, les méthodes *ad hoc* présentent un inconvénient intrinsèque évident : pour chaque nouvelle classe de structures à engendrer, il faut inventer un nouvel algorithme. Il est alors naturel d'essayer de trouver un processus générique permettant d'engendrer uniformément des objets de classes combinatoires variées. Cette idée apparaît déjà dans le livre de Nijenhuis et Wilf [NW78] où sont posées les premières pierres de la méthode récursive[4] automatisée et systématisée par Flajolet, Zimmermann et Van Cutsem [FZVC94]. Nous verrons, par la suite, que la méthode de Boltzmann, de Duchon *et al.* [DFLS04], repose sur le même principe de décomposition récursive des objets mais se place dans un cadre de génération en taille approchée qui diffère des méthodes utilisées jusque là.

2.1 Au commencement : Nijenhuis et Wilf

La première partie du livre *Combinatorial Algorithms* [NW78] est dédiée aux algorithmes de génération pour des "familles combinatoires" spécifiques. Pour chaque famille (permutations, compositions, partitions d'entiers ou d'ensembles, ...), les auteurs s'intéressent à deux type d'algorithmes assez différents, pour la génération exhaustive d'une part et la génération aléatoire uniforme d'autre part.

Les algorithmes NEXT et RAND Les premiers algorithmes, de type NEXT visent à donner un ordre total sur les structures (typiquement de même taille) d'une famille combinatoire, de telle façon que le passage d'une structure à la suivante se fait en temps moyen constant.

[4] Il existe d'autre méthodes automatiques pour la génération aléatoire, notamment [BDLP99, HC83, DF98, Gre83, ...], mais nous choisissons de nous concentrer sur la méthode récursive, plus proche de la méthode de Boltzmann.

Lorsque l'algorithme est appelé pour la première fois pour une famille de structures de taille donnée, il produit la structure qui est la première dans l'ordre choisi. Rappelé avec les mêmes paramètres, il engendre successivement tous les objets de même taille dans cet ordre. Ainsi, l'algorithme NEXKSB produit tous les sous-ensembles à k éléments d'un ensemble à n éléments en ordre lexicographique, alors que NEXSUB ordonne tous les sous-ensembles d'un ensemble à n éléments sous forme de code de Gray. Les figures 5.a et 5.b donnent des exemples de séquences obtenues par ces deux algorithmes. Ces routines sont des algorithmes de génération exhaustive dont la complexité est linéaire par rapport au nombre d'objets engendrés.

Les autres algorithmes étudiés dans [NW78] sont de type RAND : on cherche à engendrer aléatoirement une structure de taille n avec une loi uniforme. Comme pour les routines de type NEXT, ces algorithmes exploitent généralement des propriétés propres à chaque classe pour faire ces tirages. Par exemple, RANPER engendre une permutation aléatoire de taille n à partir d'une permutation triée en ordre croissant, par échanges aléatoires successifs de ses éléments (l'algorithme est donné par la figure 5.c).

Schéma général Nijenhuis et Wilf proposent comme méthode alternative un schéma général unique pour concevoir des algorithmes NEXT et RAND des diverses familles combinatoires qu'ils étudient et consistant à exploiter une propriété commune de ces familles : leur décomposabilité. En effet, ces classes d'objets peuvent être construites récursivement à partir d'objets d'*ordre* inférieur, *i.e.*, de taille plus petite et/ou composés de moins d'éléments. Cette décomposition fournit un moyen naturel pour d'une part, ordonner les objets et d'autre part les construire et les tirer au hasard. Par exemple, pour décrire la classe $\mathcal{A}_{n,k}$ des sous ensembles à k éléments d'un ensemble à n éléments, on peut distinguer ceux qui contiennent l'élément n de ceux qui ne le contiennent pas et écrire récursivement :

$$\mathcal{A}_{n,k} = \mathcal{A}_{n-1,k} + \mathcal{A}_{n-1,k-1} \times \{n\} \quad \text{pour } 0 \leq k \leq n, \tag{2}$$

où $\mathcal{A}_{n-1,k-1} \times \{n\}$ correspond aux ensembles formés de l'élément n et d'un ensemble appartenant à $\mathcal{A}_{n-1,k-1}$ et où le signe + dénote l'union disjointe. Cette équation permet dans un premier temps de dénombrer les éléments de $\mathcal{A}_{n,k}$. En notant $a_{n,k}$ la cardinalité de $\mathcal{A}_{n,k}$, on obtient

$$a_{n,k} = a_{n-1,k} + a_{n-1,k-1}, \quad a_{0,0} = 1,$$

où l'on reconnaît la formule du triangle de Pascal : $a_{n,k} = \binom{n}{k} = \binom{n}{k-1} + \binom{n-1}{k-1}$. D'autre part, cette relation fournit un ordre partiel sur les $\mathcal{A}_{n,k}$, le maximum étant $\mathcal{A}_{n,k}$ et le minimum $\mathcal{A}_{0,0}$. On peut représenter cet ordre sous la forme d'un graphe acyclique dirigé G dont les sommets sont les $\mathcal{A}_{i,j}$ dont dépend $\mathcal{A}_{n,k}$. Les arêtes de G sont étiquetées par les éléments ajoutés aux ensembles successifs. La figure 6 représente par exemple le graphe associé à $\mathcal{A}_{4,2}$. L'ensemble des $\mathcal{A}_{n,k}$-structures correspond alors à l'ensemble des chemins possibles de $\mathcal{A}_{n,k}$ à $\mathcal{A}_{0,0}$. On obtient un algorithme de type NEXT en ordonnant ces chemins. Sur ce principe général, on peut construire un algorithme NEXT pour toute classe que l'on sait définir récursivement à la manière de (2). L'algorithme RAND correspondant se contente alors de choisir un rang aléatoire r entre 1 et $a_{n,k}$ et de renvoyer le r-ième élément dans l'ordre de l'algorithme NEXT. Dans [Wil77], un deuxième algorithme dérivé du graphe d'ordre partiel est proposé. Pour engendrer une $\mathcal{A}_{n,k}$-structure, il suffit de suivre une marche aléatoire dans G, en partant du sommet $\mathcal{A}_{n,k}$ et en choisissant, pour chaque sommet $\mathcal{A}_{i,j}$ traversé, d'aller vers le sommet $\mathcal{A}_{i-1,j}$ avec probabilité $\frac{a_{i-j,j}}{a_{i,j}}$ et vers le sommet $\mathcal{A}_{i-1,j-1}$ sinon. C'est sur ce principe, qui se généralise aux autres classes définies récursivement, qu'est fondée la méthode récursive.

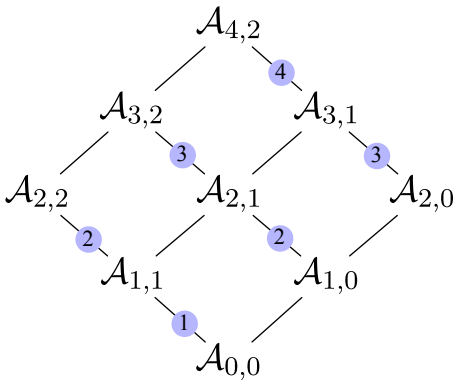


FIG. 6 – Graphe représentant l'ordre partiel sur les $\mathcal{A}_{n,k}$ inférieurs à $\mathcal{A}_{4,2}$.

Dans [NW78], Nijenhuis et Wilf donnent également le schéma général qui permet d'engendrer récursivement des multi-ensembles de structures combinatoires, à condition de disposer d'un générateur pour ces structures. Le multi-ensemble, ainsi que l'union disjointe et le produit cartésien, font partie des constructions combinatoires que l'on peut assembler récursivement pour décrire un grand nombre de classes. L'idée de la méthode récursive est d'adapter le principe de génération que nous venons de voir à l'ensemble de ces constructions.

2.2 Vers l'automatisation : la méthode symbolique [FS08, part A]

Si des méthodes de génération aléatoire complètement automatiques telles que la méthode récursive ou le modèle de Boltzmann ont pu voir le jour, c'est avant tout parce que les traitements combinatoires nécessaires à la génération ont eux-mêmes été préalablement automatisés. La méthode symbolique exposée en détails dans [FS08] offre un cadre de travail générique qui permet de décrire récursivement un grand nombre de classes combinatoires incluant celles que nous avons vues jusqu'à présent (arbres binaires, mots de Dyck ou de Motzkin, permutations, ...). Un autre aspect essentiel pour la génération aléatoire à taille fixée est la capacité de compter les objets que l'on souhaite engendrer. La méthode symbolique fournit également les outils qui permettent d'automatiser le dénombrement des structures.

2.2.1 Classes combinatoires spécifiables

Les structures combinatoires qui vont nous intéresser maintenant sont celles que l'on peut spécifier, éventuellement récursivement, à partir de plus petites structures du même type ou de type plus simple et de structures atomiques, en utilisant des règles de construction classiques de la combinatoire. Intuitivement, ces spécifications sont des grammaires *context-free* analogues à celles des langages formels, mais utilisant des constructions supplémentaires. Nous avons déjà utilisé deux de ces constructions, le produit cartésien noté $\times$ et l'union disjointe, notée $+$, pour décrire les éléments de la classe $\mathcal{A}_{n,k}$ (équation (2), section précédente). La liste des constructions *admissibles* est donnée par la colonne 1 de la table 1. Les notations correspondantes sont donnés par la colonne 2. Pour toute construction $\mathfrak{C}$ ayant un nombre indéterminé de composantes (séquence, ensemble et cycle), on peut rajouter des contraintes pour fixer leur cardinalité (que l'on note $\mathfrak{C}_\ell$). Les mêmes constructions avec au plus ℓ éléments s'obtiennent par union disjointe des $\mathfrak{C}_i$ pour $i \leq \ell$ et celles avec au moins ℓ éléments par soustraction de $\mathfrak{C}_{\ell-1}$.

construction	notation	$\widehat{C}(z)$	$C(z)$
Atome de taille 0, 1	$\mathcal{E},\ \mathcal{Z}$	$1,\ z$	$1,\ z$
Union disjointe	$\mathcal{A}+\mathcal{B}$	$\hat{A}(z)+\widehat{B}(z)$	$A(z)+B(z)$
Produit cartésien	$\mathcal{A}\times\mathcal{B}$	$\hat{A}(z)\cdot\widehat{B}(z)$	$A(z)\cdot B(z)$
Séquence	$\textsc{Seq}(\mathcal{B})$	$1/(1-\widehat{B}(z))$	$1/(1-B(z))$
Séquence, $\#\ell>0$	$\textsc{Seq}_\ell(\mathcal{B})$	$\widehat{B}^\ell(z)$	$B^\ell(z)$
Cycle	$\textsc{Cyc}(\mathcal{B})$	$\log\frac{1}{1-\widehat{B}(z)}$	$\sum_{k=1}^{\infty}\frac{\varphi(k)}{k}\log\frac{1}{1-B(z^k)}$
Cycle, $\#\ell>0$	$\textsc{Cyc}_\ell(\mathcal{B})$	$\frac{1}{\ell}\widehat{B}^\ell(z)$	$[u^\ell]\sum_{k=1}^{\infty}\frac{\varphi(k)}{k}\log\frac{1}{1-u^kB(z^k)}$
Ensemble	$\textsc{Set}(\mathcal{B})$	$\exp(\widehat{B}(z))$	$\exp\left(\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}B(z^k)\right)$
Ensemble, $\#\ell>0$	$\textsc{Set}_\ell(\mathcal{B})$	$\frac{1}{\ell!}\widehat{B}^\ell(z)$	$[u^\ell]\exp\left(uB(z)-\frac{u^2}{2}B(z^2)+\frac{u^3}{3}B(z^3)-\dots\right)$
Multi-ensemble	$\textsc{MSet}(\mathcal{B})$	–	$\exp\left(\sum_{k=1}^{\infty}\frac{1}{k}B(z^k)\right)$
Multi-ensemble, $\#\ell>0$	$\textsc{MSet}_\ell(\mathcal{B})$	–	$[u^\ell]\exp\left(uB(z)+\frac{u^2}{2}B(z^2)+\frac{u^3}{3}B(z^3)+\dots\right)$

Tab. 1 – Constructions admissibles et leurs fonctions génératrices. Le caractère "–" indique qu'il n'y a pas de multi-ensemble étiqueté et on note $\#\ell$ une construction à ℓ composantes.

Définition 1. *Une classe combinatoire $\mathcal{C}_1$ est dite* décomposable *si elle admet une spécification*[5] *de la forme :*

$$\begin{cases}\mathcal{C}_1=\mathcal{H}_1(\mathcal{Z},\mathcal{C}_1,\mathcal{C}_2,\dots,\mathcal{C}_m),\\ \mathcal{C}_2=\mathcal{H}_2(\mathcal{Z},\mathcal{C}_1,\mathcal{C}_2,\dots,\mathcal{C}_m),\\ \quad\vdots\\ \mathcal{C}_m=\mathcal{H}_m(\mathcal{Z},\mathcal{C}_1,\mathcal{C}_2,\dots,\mathcal{C}_m),\end{cases}$$

où chaque $\mathcal{H}_i$ dénote un terme formé à partir des $\mathcal{C}_i$ et d'atomes ($\mathcal{E}$ ou $\mathcal{Z}$) en utilisant les constructions de la table 1. Chaque non-terminal $\mathcal{C}_i$ correspond à une classe combinatoire et les objets dérivés par cette grammaire généralisée, partant de l'axiome $\mathcal{C}_i$, seront appelés des $\mathcal{C}_i$-structures*. La* taille *d'une structure est le nombre de terminaux $\mathcal{Z}$ qui la composent.*

On peut par exemple, réécrire la grammaire des mots de Dyck de la section 1.1.2 sous la forme d'une spécification combinatoire :

$$\mathcal{D}=\mathcal{E}+\mathcal{A}\times\mathcal{D}\times\mathcal{B}\times\mathcal{D},\quad \mathcal{A}=\mathcal{Z},\quad \mathcal{B}=\mathcal{Z},$$

ou encore décrire récursivement l'ensemble des arbres binaires $\mathcal{B}$ qui peuvent être soit une feuille $\mathcal{E}$, soit le produit d'un nœud $\mathcal{Z}$ et de deux sous-arbres binaires $\mathcal{B}\times\mathcal{B}$:

$$\mathcal{B}=\mathcal{E}+\mathcal{Z}\times\mathcal{B}^2.$$

[5] On pourra remarquer qu'à l'inverse, toute spécification de cette forme ne décrit pas une classe combinatoire, au sens où, dans cette définition, rien ne garantit qu'il existe un nombre fini de structures pour chaque sous-classe de taille donnée. Nous donnerons au chapitre 4 des conditions suffisantes pour cela.

non étiqueté		
$\mathcal{P} = \text{MSET}(\text{SEQ}_{\geq 1}(\mathcal{Z}))$	partitions d'entier	$P(z) = \exp\left(\sum_{k=1}^{\infty} \frac{z^k}{k(1-z^k)}\right)$
$\mathcal{Q} = \text{PSET}(\text{SEQ}_{\geq 1}(\mathcal{Z}))$	partitions en sommants distincts	$Q(z) = \exp\left(\sum_{k=1}^{\infty} \frac{(-1)^{k-1} z^k}{k(1-z^k)}\right)$
$\mathcal{C} = \text{SEQ}(\text{SEQ}_{\geq 1}(\mathcal{Z}))$	compositions d'entier	$C(z) = \frac{1}{1-\frac{z}{1-z}}$
$\mathcal{B} = \mathcal{Z} + \mathcal{B} \times \mathcal{B}$	arbres binaires planaires	$B(z) = z + B(z)^2$
$\mathcal{A} = \mathcal{Z} \times \text{SEQ}(\mathcal{A})$	arbres généraux planaires	$A(z) = \frac{z}{1-A(z)}$
étiqueté		
$\mathcal{P}_e = \text{SET}(\text{CYC}(\mathcal{Z}))$	permutations	$P_e(z) = e^{\log 1/(1-z)} = \frac{1}{1-z}$
$\begin{cases} \mathcal{S} = \text{SEQ}_{\geq 2}(\mathcal{P} + \mathcal{Z}) \\ \mathcal{P} = \text{SET}_{\geq 2}(\mathcal{S} + \mathcal{Z}) \end{cases}$	graphes série-parallèle	$\begin{cases} S(z) = \frac{(P(z)+z)^2}{1-(P(z+z))} \\ P(z) = e^{S(z)+z} - S(z) - z \end{cases}$
$\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T}))$	arbres non planaires	$T(z) = ze^{T(z)}$
$\begin{cases} \mathcal{G} = \text{SET}(\text{CYC}(\mathcal{T})) \\ \mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})) \end{cases}$	graphes fonctionnels	$\begin{cases} G(z) = e^{\log \frac{1}{1-T(z)}} = \frac{1}{1-T(z)} \\ T(z) = ze^{T(z)} \end{cases}$

TAB. 2 – Exemples de spécifications combinatoires

La taille d'un arbre correspond alors à son nombre de nœuds internes. On pourrait également décrire des arbres binaires par la spécification $\mathcal{A} = \mathcal{Z} + \mathcal{B}^2$. Un arbre de taille n serait, dans ce cas là, un arbre à n feuilles.

On distingue deux types de structures combinatoires : celles qui sont étiquetées et celles qui ne le sont pas. Une structure étiquetée de taille n est une structure dont les n atomes portent des étiquettes différentes, appartenant à l'ensemble $\{1, \ldots, n\}$. À l'inverse, les atomes d'une structure non étiquetée sont indistinguables. Typiquement, la spécification que nous avons donnée pour les mots de Dyck n'est pas étiquetée alors que celle des arbre peut l'être. D'autres exemples de spécifications pour quelques classes combinatoires classiques sont donnés par les colonnes 1 et 2 de la table 2. On peut décrire ainsi tous les langages algébriques et un certain nombre de structures arborescentes.

2.2.2 Dénombrement et séries génératrices

Pour engendrer aléatoirement des structures de taille fixée en utilisant ces spécifications récursives, une première étape est de savoir les dénombrer. Là encore, la méthode symbolique nous fournit un dictionnaire d'outils automatiques pour cette tâche : les *séries génératrices*.

Séries ordinaires, exponentielles La série génératrice, ou série de dénombrement d'une classe combinatoire $\mathcal{C}$ est une série formelle dont les coefficients comptent les $\mathcal{C}$-structures selon leur taille. Par convention, on notera c_n le nombre de structures de taille n dans la classe $\mathcal{C}$.

Définition 2. *La série génératrice* ordinaire *d'une classe combinatoire non étiquetée $\mathcal{C}$ est la*

série formelle

$$C(z) = \sum_{\gamma \in \mathcal{C}} z^{|\gamma|} = \sum_{n=0}^{\infty} c_n z^n.$$

La série génératrice exponentielle *d'une classe combinatoire étiquetée $\mathcal{C}$ est la série formelle*

$$\widehat{C}(z) = \sum_{\gamma \in \mathcal{C}} \frac{z^{|\gamma|}}{|\gamma|!} = \sum_{n=0}^{\infty} c_n \frac{z^n}{n!}.$$

On note $[z^n]C(z) = c_n$ l'opération consistant à extraire le coefficient de z^n dans $C(z)$.

Par exemple, la série génératrice des arbres binaires est :

$$B(z) = \sum_{n \geq 0} b_n z^n = 1 + z + 2\,z^2 + 5\,z^3 + 14\,z^4 + 42\,z^5 + 132\,z^6 + \dots\ ,$$

où $b_n = \frac{1}{n+1}\binom{2n}{n}$ est le n-ième nombre de Catalan.

La force de la méthode symbolique est que les séries génératrices des classes décomposables peuvent être obtenues comme une traduction automatique des spécifications. À chaque construction combinatoire admissible, on peut associer une opération sur les séries génératrices ordinaires ou exponentielles. Ce dictionnaire est donné par la table 1. La colonne 3 donne les séries génératrices exponentielles correspondant aux structures étiquetées et la colonne 4, les séries ordinaires pour les structures non étiquetées. La colonne 3 de la table 2 donne quelques exemples de séries génératrices classiques.

Séries bivariées, séries génératrices de probabilité Nous terminons cette petite introduction à la méthode symbolique par la présentation des séries bivariées, un outil permettant de manipuler d'autres paramètres que la taille des structures combinatoires décomposables.

Définition 3. *Soit $\mathcal{C}$ un classe combinatoire et un paramètre $\chi : \mathcal{C} \to \mathbb{N}$. La* série génératrice bivariée *associée à la paire $\langle \mathcal{C}, \chi \rangle$ est*

$$C(z,u) = \sum_{\gamma \in \mathcal{C}} u^{\chi(\gamma)} z^{|\gamma|} = \sum_{n,k \geq 0} c_{n,k} u^k z^n \qquad \textit{dans le cas ordinaire/non étiqueté}$$

$$\widehat{C}(z,u) = \sum_{\gamma \in \mathcal{C}} u^{\chi(\gamma)} \frac{z^{|\gamma|}}{|\gamma|!} = \sum_{n,k \geq 0} c_{n,k} u^k \frac{z^n}{n!} \qquad \textit{dans le cas exponentiel/ étiqueté}$$

Dans les deux cas, $c_{n,k}$ est le nombre de $\mathcal{C}$-structures de taille n et dont le paramètre vaut k.

Par exemple, la série bivariée ordinaire des sous-ensembles à k éléments d'un ensemble à n éléments (ou des mots de $\{a,b\}^*$ de taille n, ayant k lettres b) est

$$A(z,u) = \sum_{n,k \geq 0} \binom{n}{k} u^k z^n.$$

On dira qu'un paramètre χ est *hérité* s'il est défini par cas pour $\mathcal{C} = \mathcal{A} + \mathcal{B}$ et par addition sur les composantes pour $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ ou pour $\mathcal{C} = \mathfrak{C}(\mathcal{A})$, avec $\mathfrak{C} \in \{\textsc{Seq}, \textsc{MSet}, \textsc{PSet}, \textsc{Cyc}\}$. Dans le cas de paramètres hérités, les règles de traduction des séries génératrices données plus haut s'appliquent également aux séries bivariées. Par exemple, pour la classe des mots de $\{a,b\}^*$

dont la spécification est $\mathcal{W} = \textsc{Seq}(\mathcal{Z}_a + \mathcal{Z}_b)$, la série bivariée $W(u, z)$ dans laquelle la variable u marque le nombre de lettres a de chaque mot est

$$W(z, u) = \frac{1}{1 - (z + uz)}.$$

La *série génératrice de probabilités* du paramètre χ pris sur la sous-classe $\mathcal{C}_n$ des $\mathcal{C}$-structures de taille n est :

$$p(u) = \sum_k \mathbb{P}_{\mathcal{C}_n}(\chi = k)u^k = \frac{[z^n]C(z, u)}{[z^n]C(z, 1)}$$

et l'espérance de la valeur du paramètre χ sur la classe $\mathcal{C}_n$ est donnée par la formule :

$$\mathbb{E}_{\mathcal{C}_n}(\chi) = \frac{[z^n]\frac{\partial}{\partial u}C(z, u)|_{u=1}}{[z^n]C(z)}.$$

On obtient, par exemple, que le nombre moyen de a dans un mot quelconque de longueur n sur l'alphabet $\{a, b\}$ est :

$$\mathbb{E}_{\mathcal{W}_n}(\text{nombre de } a) = \frac{[z^n]\frac{z}{(1-2z)^2}}{2^n} = \frac{n}{2}.$$

Plus généralement, la série bivariée permettra d'obtenir toutes les caractéristiques de la distribution du paramètre χ (variance, moments d'ordre supérieur, loi limite,...).

2.3 La méthode récursive

Le travail de Flajolet *et al.* [FZVC94] reprend l'idée de tirer partie de la décomposition récursive des classes de structures combinatoires exposée dans [Wil77, NW78] et la systématise pour les structures étiquetées décomposables. Bien que ce travail n'ait pas été publié[6], cette méthode s'étend aux structures combinatoires non étiquetées. Cette section reprend les principaux résultats de [FZVC94] ; leur présentation est inspirée de [Den01].

2.3.1 Principe

L'idée de la méthode récursive est que le processus de génération aléatoire dans une classe combinatoire donnée peut se déduire directement de sa spécification récursive. Engendrer aléatoirement une structure de taille n dans la classe $\mathcal{C}$ définie comme l'union de deux classes $\mathcal{A}$ et $\mathcal{B}$, revient à tirer uniformément une $\mathcal{A}$-structure avec probabilité $|\mathcal{A}|/|\mathcal{C}|$ et une $\mathcal{B}$-structure sinon. Dans le cas d'une classe $\mathcal{C}$ définie comme un produit, un ensemble ou un cycle, tirer aléatoirement une $\mathcal{C}$-structure de taille n consiste à choisir les tailles (et éventuellement le nombre) des sous-structures qui la composent, de telle sorte que la somme de leurs tailles soit n et de telle façon que toutes les $\mathcal{C}$-structures aient une probabilité $1/|\mathcal{C}|$ d'être tirées.

Suivant ce principe, et à condition de savoir dénombrer les structures, les algorithmes de génération associés aux constructions admissibles se déduisent directement des spécifications. Dans un premier temps, ces spécifications sont transformées en spécifications standard utilisant uniquement des unions, des produits binaires et un opérateur de pointage (voir section suivante). Les algorithmes obtenus sont regroupés dans la fonction $GenRec(\mathcal{C}, n)$ de la figure 7. Les coefficients qui dénombrent les structures de taille n, c'est-à-dire les coefficients des séries

[6] On pourra néanmoins consulter les notes d'un séminaire donné par Paul Zimmermann au projet ALGO, sur la méthode récursive appliquée au cas non étiqueté [Zim94b].

GenRec$(\mathcal{C}, n)$

Cas $\mathcal{C} = \mathcal{E}$:	**Cas** $\mathcal{C} = \mathcal{Z}$:	
si $n = 0$ **alors**	**si** $n = 1$ **alors**	
renvoyer $\mathcal{E}$	renvoyer $\mathcal{Z}$	
Cas $\mathcal{C} = \mathcal{A} + \mathcal{B}$:	**Cas** $\mathcal{C} = \mathcal{A} \cdot \mathcal{B}$:	
$U \leftarrow$ random()	$U \leftarrow$ random()	$\mathbb{P}(K = k)$
si $U < (a_n/c_n)$ **alors**	$k \leftarrow 0$; $s \leftarrow (a_0 \cdot b_n)/c_n$	$=$
renvoyer **Gen**$\mathcal{C}(\mathcal{A}, n)$	**tant que** $U > s$ **faire**	$\frac{a_k b_{n-k}}{c_n}$
sinon	$k \leftarrow k + 1$; $s \leftarrow s + (a_k \cdot b_{n-k})/c_n$	
renvoyer **Gen**$\mathcal{C}(\mathcal{B}, n)$	renvoyer $\langle$ **Gen**$\mathcal{C}(\mathcal{A}, k)$, **Gen**$\mathcal{C}(\mathcal{B}, n-k) \rangle$	

FIG. 7 – Algorithme de génération récursive.

génératrices, peuvent être pré-calculés avant la phase de génération proprement dite. Ainsi ce calcul n'est effectué qu'une seule fois, quel que soit le nombre de tirages faits par le générateur. L'algorithme consiste à utiliser des relations de récurrence sur les coefficients, déduites des spécifications standard [FSZ91].

Exemple 1. Les arbres planaires enracinés sont décrits par la spécification

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

En utilisant la définition récursive d'une séquence, cette spécification devient :

$$\mathcal{T} = \mathcal{Z} \times \mathcal{F} \quad \text{et} \quad \mathcal{F} = \mathcal{T} \times \mathcal{F},$$

où $\mathcal{F}$ représente la classe de séquences d'arbres, *i.e.*, des forêts. Les coefficients t_n et f_n comptant le nombre d'arbres et de forêts de taille n s'obtiennent à partir des relations :

$$t_n = f_n - 1 \quad \text{et} \quad f_n = \begin{cases} 1 & \text{si } n = 0 \\ \sum_{k=0}^{n-1} f_k t_{n-k} & \text{sinon.} \end{cases}$$

On dérive directement les générateurs $Gen\mathcal{T}(n)$ et $Gen\mathcal{F}(n)$ à partir de l'algorithme générique de la figure 7.

GenT(n)

renvoyer $\mathcal{Z} \times Gen\mathcal{F}(n-1)$

GenF(n)

si $n = 0$ **alors** renvoyer $\varnothing$
sinon
choisir k avec proba $f_k t_{n-k}/f_n$
renvoyer $Gen\mathcal{T}(k) \times Gen\mathcal{F}(n-k)$

De façon plus schématique, pour générer une structure de taille n appartenant à la classe $\mathcal{C}$, à partir de sa spécification, on construit récursivement un arbre de toutes les dérivations possibles de la règle définissant $\mathcal{C}$ et aboutissant à une $\mathcal{C}$-structure de taille n. On attribue à chaque arête de l'arbre une probabilité $1/k$, où k est le nombre de feuilles dont le chemin à la racine passe par cette arête, *i.e.*, le nombre de structures que l'on peut engendrer en faisant le

choix de cette arête. Il suffit alors de tirer aléatoirement une feuille en suivant les probabilités sur les arêtes. La figure 8 illustre cette idée sur des arbres généraux non planaires. Bien entendu, en réalité, on ne construit pas l'arbre de dérivation, mais on suit un chemin virtuel jusqu'à une feuille en fonction des tirages aléatoires.

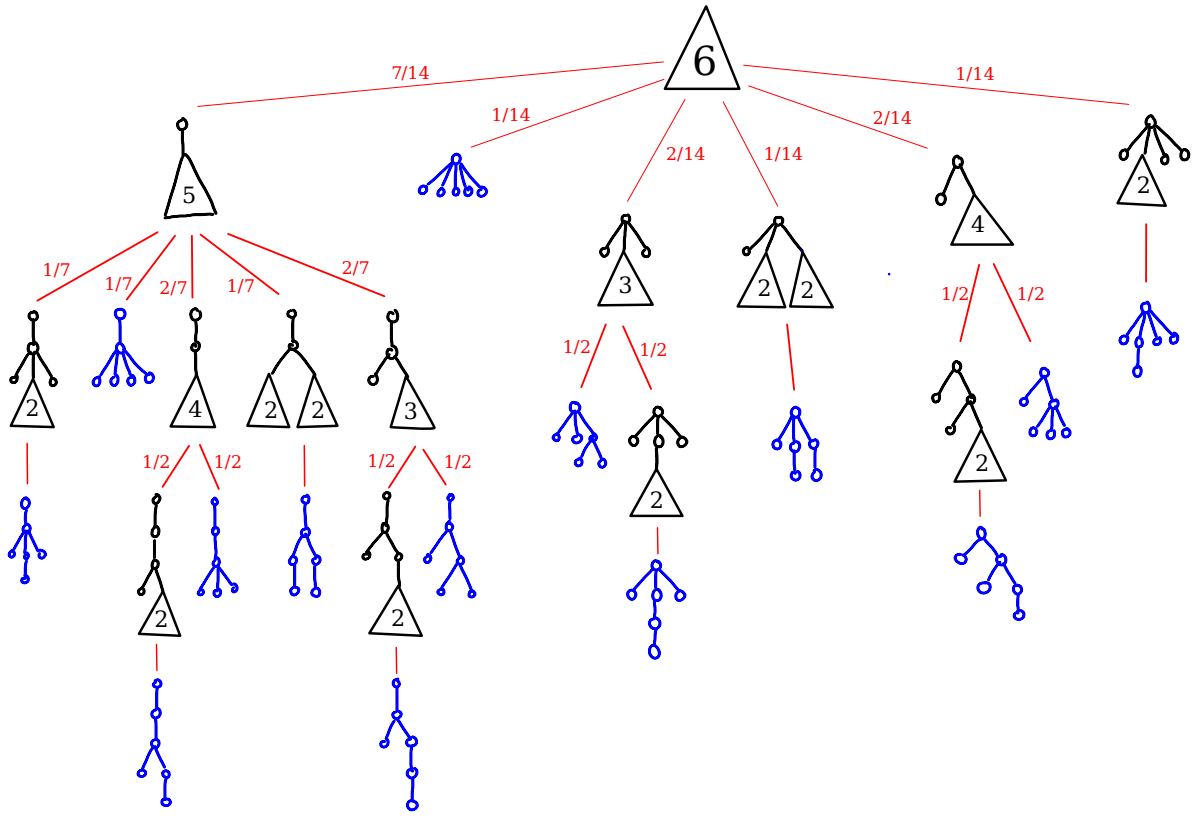


FIG. 8 – La méthode récursive illustrée sur les arbres de Cayley : $\mathcal{T} = \mathcal{Z} \times \textsc{Set}(\mathcal{T})$. Un arbre de taille 6 est obtenu comme le produit d'un nœud et d'une séquence de sous-arbres dont la somme des tailles est égale à 5.

2.3.2 Implantation

Spécification standard Pour pouvoir utiliser le générateur $GenRec(\mathcal{C}, n)$, il faut procéder à un premier pré-traitement sur la spécification de $\mathcal{C}$ qui consiste à la mettre sous une forme standard. Ce principe de standardisation étend la mise sous forme normale de Chomsky pour les grammaires hors contexte. Cette opération consiste essentiellement à réécrire toutes les constructions admissibles en termes d'unions et de produits binaires. Elle est définie de la façon suivante pour les classes étiquetées, par Flajolet, Zimmermann et Van Cutsem :

Définition 4 ([FZVC94]). *Soit $\boldsymbol{\mathcal{C}} = (\mathcal{C}_1, \ldots, \mathcal{C}_m)$ un m-uplet de classes combinatoires. Une* spécification standard *de $\boldsymbol{\mathcal{C}}$ est un système à m équations dont la $i$ème s'écrit sous la forme :*

$$\mathcal{C}_i = \mathcal{E}; \quad \mathcal{C}_i = \mathcal{Z}; \quad \mathcal{C}_i = \mathcal{U}_j + \mathcal{U}_k; \quad \mathcal{C}_i = \mathcal{U}_j \cdot \mathcal{U}_k; \quad \textit{ou} \quad \Theta\mathcal{C}_i = \mathcal{U}_j \cdot \mathcal{U}_k,$$

chaque $\mathcal{U}_i$ appartenant à l'ensemble $\{\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \ldots, \mathcal{C}_m, \Theta\mathcal{C}_1, \ldots, \Theta\mathcal{C}_m\}$.

constructions étiquetées		
$\mathcal{C} = \textsc{Seq}(\mathcal{B})$	$\Rightarrow$	$\mathcal{C} = \mathcal{E} + \mathcal{B} \cdot \mathcal{C}$
$\mathcal{C} = \textsc{Cyc}(\mathcal{B})$	$\Rightarrow$	$\Theta\mathcal{C} = \textsc{Seq}(\mathcal{B}) \cdot \Theta B$
$\mathcal{C} = \textsc{Set}(\mathcal{B})$	$\Rightarrow$	$\Theta\mathcal{C} = \mathcal{C} \cdot \Theta\mathcal{B}$
constructions non étiquetées		
$\mathcal{C} = \textsc{Cyc}(\mathcal{B})$	$\Rightarrow$	$\sum_{k>=1} \varphi(k) \cdot \Delta_k\Theta\mathcal{B} \cdot \textsc{Seq}(\Delta_k\mathcal{B})$
$\mathcal{C} = \textsc{MSet}(\mathcal{B})$	$\Rightarrow$	$\Theta\mathcal{C} = \mathcal{C} \cdot (\Theta\mathcal{B} + \Delta_2\Theta\mathcal{B} + \Delta_3\Theta\mathcal{B} + \dots)$

TAB. 3 – Règles de standardisation des constructions admissibles. $\Delta_k\mathcal{C}$ représente la diagonale à k éléments d'une classe $\mathcal{C}$, c'est à dire l'ensemble des k-uplets de $\mathcal{C}$-structures identiques.

Dans cette définition, le symbole Θ désigne l'opérateur de pointage :

$$\Theta\mathcal{A} = \bigcup_{n=1}^{\infty} (\mathcal{A}_n \times \{1, 2, \dots, n\}),$$

où $\mathcal{A}_n$ est un sous-ensemble de la classe $\mathcal{A}$ contenant uniquement les structures de taille n. En d'autre termes, uns structure pointée appartenant à la classe $\Theta\mathcal{A}$ est une $\mathcal{A}$-structure dont l'un des atomes est distingué, *i.e.* marqué. D'après cette définition, si la classe $\mathcal{C}$ est définie par $\mathcal{C} = \Theta\mathcal{A}$ alors le nombre de structures de taille n dans $\mathcal{C}$ est $c_n = nAn$. L'opération équivalente sur les séries génératrices est une dérivation :

$$\mathcal{C} = \Theta\mathcal{A} \quad \Rightarrow \quad C(z) = \Theta A(z), \quad \text{où } \Theta f(z) = z \cdot \frac{d}{dz} f(z).$$

Pour l'atome, l'union et le produit, on peut également donner la règle pointée correspondante :

$$\mathcal{C} = \mathcal{Z} \Rightarrow \Theta\mathcal{C} = \mathcal{E}, \quad \mathcal{C} = \mathcal{A} + \mathcal{B} \Rightarrow \Theta\mathcal{C} = \Theta\mathcal{A} + \Theta\mathcal{B}, \quad \text{et} \quad \mathcal{C} = \mathcal{A} \cdot \mathcal{B} \Rightarrow \Theta\mathcal{C} = \Theta\mathcal{A} \cdot \mathcal{B} + \mathcal{A} \cdot \Theta\mathcal{B}.$$

Le théorème suivant nous assure qu'il existe bien une forme standard pour toute spécification combinatoire étiquetée et la preuve nous fournit l'algorithme de standardisation.

Théorème 1 ([FZVC94])**.** *Toute famille de classes combinatoires décomposables étiquetées admet une spécification standard.*

La définition d'une spécification standard reste valable pour les classes non étiquetées et le théorème s'étend également. Ainsi, chaque construction donnée dans la table 1 se réécrit en utilisant la règle correspondante de la table 3. On peut également donner les règles de standardisation pour les constructions ayant des contraintes de cardinalité, par exemple :

$$\mathcal{C}^{(k)} = \textsc{Set}_k(\mathcal{B}) \;\Rightarrow\; \Theta\mathcal{C}^{(k)} = \mathcal{C}^{(k-1)} \cdot \Theta\mathcal{B}, \qquad \text{avec } \mathcal{C}^{(1)} = \mathcal{B}$$

On dispose donc d'une collection d'algorithmes qui permettent de transformer toute spécification combinatoire rentrant dans le cadre de la définition 1 en une spécification standard. Ainsi transformée, on pourra alors appliquer le principe de génération récursive.

Exemple 2. La classe $\mathcal{G}$ des graphes fonctionnels est spécifiée par la grammaire étiquetée $\{\mathcal{G} = \textsc{Set}(\textsc{Cyc}(\mathcal{T})),\ \mathcal{T} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{T})\}$. La forme standard de cette grammaire est : $\{\mathcal{T} = \mathcal{Z} \cdot \mathcal{N}_0,\ \Theta\mathcal{N}_0 = \mathcal{N}_0 \cdot \Theta\mathcal{T},\ \Theta\mathcal{G} = \mathcal{G} \cdot \Theta\mathcal{N}_1,\ \Theta\mathcal{N}_1 = \mathcal{N}_2 \cdot \Theta\mathcal{T},\ \mathcal{N}_2 = \mathcal{E} + \mathcal{N}_3,\ \mathcal{N}_3 = \mathcal{T} \cdot \mathcal{N}_2\}$.

Du fait de la standardisation des spécifications, on pourrait, théoriquement, avoir à engendrer récursivement des structures pointées ; dans ce cas, il suffirait d'*effacer* les marques pour obtenir les structures voulues ; le nombre de marquages possibles pour une structure ne dépend que de sa taille, et donc le tirage resterait uniforme. En pratique, il suffit de ne jamais marquer les structures, l'opérateur de pointage servant uniquement au calcul des probabilités associées aux tirages aléatoires.

Complexité Pour engendrer des structures de taille n, le processus de génération a une complexité arithmétique en $O(n^2)$, dès que la spécification contient une règle produit. Effectivement, on constate que les règles de la forme $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ sont particulièrement coûteuses : pour engendrer une $\mathcal{C}$-structure dont la taille est exactement n, il faut choisir les tailles des structures qui formeront le produit de façon à garantir l'uniformité, et ce choix se fait de façon séquentielle. L'astucieux moyen, proposé dans [FZVC94], pour améliorer la complexité globale des générateurs récursifs est d'utiliser plutôt une recherche de type *boustrophédon*. La complexité du générateur devient $O(n \log n)$ dans le pire cas.

Avec un algorithme naïf, le pré-traitement effectué pour calculer les coefficients de séries génératrices a une complexité arithmétique quadratique (voir [FSZ91]) ; néanmoins, de récents travaux permettent d'abaisser sensiblement cette complexité ([vdH02], cette thèse, chapitre 6). Cependant, dans ce cas, la complexité arithmétique donne une approximation trop optimiste de la complexité effective de l'algorithme ,étant donné que les entiers manipulés peuvent être très grands. Pour pallier ce problème, Denise et Zimmermann [DZ99] ont montré qu'en utilisant une arithmétique flottante, on pouvait se ramener, pour le pré-traitement et la génération, à des complexités en bits proches des complexités arithmétiques.

Librairies existantes Cette méthode de génération aléatoire est largement utilisée. Elle est implantée en Maple par la librairie `Combstruct` (anciennement `Gaïa` [Zim94a]), ainsi que dans le package MuPad-Combinat [HT03] (anciennement CS-MuPad [DDZ98]). Cette librairie implante les améliorations de [DZ99], liées à l'utilisation de l'arithmétique flottante. Elle offre également la possibilité de produire automatiquement du code C pour les générateurs aléatoires.

3 La méthode de Boltzmann

La méthode récursive repose en partie sur le dénombrement des structures combinatoires. Pour engendrer aléatoirement et uniformément des structures de taille n à l'intérieur d'une classe $\mathcal{C}$, il faut d'abord savoir les compter, elles-mêmes et toutes les structures intermédiaires qui entrent en jeu dans la spécification de $\mathcal{C}$. Mais compter des structures combinatoires n'est pas un problème trivial, et la phase de pré-traitement nécessaire à la mise en œuvre de la méthode récursive limite bien souvent la taille des structures qu'il est possible d'engendrer à quelques milliers de nœuds. L'idée à l'origine des générateurs de Boltzmann est qu'en *relâchant* un peu la contrainte sur la taille des structures à engendrer, on peut se dispenser de savoir les dénombrer, et ainsi obtenir des générateurs plus efficaces. On ne cherche plus à générer uniformément une structure de taille n, mais une structure dont la taille est approximativement n. Cette taille devient alors une variable aléatoire dont la distribution s'étend à la totalité des structures de la classe $\mathcal{C}$. Mais la contrainte d'uniformité doit être respectée : pour une taille n donnée, n'importe quelle structure de taille n sera invariablement tirée avec la même probabilité. Cette section résume les principaux résultats de Duchon *et al.* [DFLS02, DFLS04].

Les séries génératrices en tant que fonctions analytiques [FS08, part B] Afin de définir le modèle de Boltzmann pour la génération aléatoire, nous devons franchir une étape supplémentaire dans l'utilisation des séries génératrices comme outil d'étude des structures combinatoires. La méthode symbolique manipule les séries génératrices en tant qu'objets purement formels auxquels on peut appliquer des opérations algébriques, mais dès lors qu'on assigne des valeurs complexes au variables apparaissant dans ces séries, on peut les interpréter comme des fonctions analytiques. Ce faisant, il est possible d'obtenir une grande quantité d'information sur le comportement asymptotique des coefficients qui dénombrent les structures combinatoires. L'analyse des singularités (les points où la fonction cesse d'être analytique) joue un rôle primordial dans cette approche. En particulier, l'un des principes fondamentaux est que l'ordre de grandeur exponentiel des coefficients d'une série génératrice est dicté par la localisation des singularités dominantes de cette fonction et les facteurs sous-exponentiels sont liés à la nature de ces singularités. Une importante collection d'outils pour l'analyse asymptotique des coefficients des séries génératrices est présentée dans [FS08].

Dans le modèle de Boltzmann, c'est en tant que fonctions analytiques que les séries génératrices interviennent. L'uniformité des tirages repose sur leur évaluation en un point réel positif choisi à l'intérieur de leur disque de convergence, c'est-à-dire inférieur à la singularité dominante réelle positive[7]. De plus, les outils d'analyse asymptotique de [FS08] peuvent intervenir dans les "réglages" des générateurs en taille fixée (exacte ou approximative) ainsi que dans l'analyse de leur complexité.

3.1 Modèle de Boltzmann

Pour une classe $\mathcal{C}$ donnée, le modèle de Boltzmann de paramètre x attribue à toute structure γ de $\mathcal{C}$ une probabilité qui est proportionnelle à une exponentielle de sa taille.

Définition 5. *Le* modèle de Boltzmann *de paramètre x existe en deux variétés, ordinaire ou exponentiel. Il assigne à toute structure γ dans $\mathcal{C}$ la probabilité suivante :*

$$\textit{ordinaire/non étiqueté :} \qquad \mathbb{P}_x(\gamma) = \frac{x^{|\gamma|}}{C(x)} \qquad \textit{avec} \quad C(x) = \sum_{\delta\in\mathcal{C}} x^{|\delta|} \tag{3}$$

$$\textit{exponentiel/ étiqueté :} \qquad \mathbb{P}_x(\gamma) = \frac{1}{|\gamma|!}\frac{x^{|\gamma|}}{\widehat{C}(x)} \quad \textit{avec} \quad \widehat{C}(x) = \sum_{\delta\in\mathcal{C}} \frac{x^{|\delta|}}{|\delta|!} \tag{4}$$

Un générateur de Boltzmann $\Gamma\mathcal{C}(x)$ pour $\mathcal{C}$ est un algorithme qui engendre des $\mathcal{C}$-structures suivant une distribution conforme à ce modèle, qu'il soit ordinaire ou exponentiel.

On constate que la probabilité associée à une structure ne dépend que de sa taille et du paramètre choisi. Pour un paramètre donné, deux structures de même taille ont donc exactement la même probabilité d'être tirées. On notera également que, contrairement à la méthode récursive, cette probabilité met en jeu la *valeur* de la fonction génératrice au point x, et non plus ses coefficients. Ce point doit être cohérent, c'est à dire choisi à l'intérieur du disque de convergence de la série génératrice de $\mathcal{C}$. En d'autres termes, x doit appartenir à l'intervalle $]0, \rho[$, où ρ est la singularité dominante de la série $C(x)$ (ou $\widehat{C}(x)$). Si cette série converge en sa singularité, il est possible d'étendre ce domaine à $]0, \rho]$. D'un point de vue pratique, cette différence entre la méthode récursive et le modèle de Boltzmann est cruciale. En effet, dans ce modèle, il n'est plus nécessaire

[7] D'après le théorème de Pringsheim (voir [FS08] pour un énoncé précis).

de calculer les séries de dénombrement des structures à engendrer, ce qui évite le pré-calcul des coefficients. En contre-partie, se pose désormais le problème de calculer des valeurs de séries génératrices en un point. Ceci fera l'objet d'une étude détaillée dans la deuxième partie de ce mémoire.

Les générateurs de Boltzmann tels que nous venons de les définir sont dits *libres*, car ils opèrent "librement", sous le seul effet du paramètre x. Autrement dit, à ce stade, la taille des structures engendrées n'est pas contrôlée. On les nomme ainsi par opposition aux générateurs en taille approchée ou exacte que nous verrons dans la section 3.3. Néanmoins, on connaît précisément la distribution des tailles sous ce modèle. Si l'on note N la variable aléatoire correspondant à la taille des structures engendrées, alors la probabilité de tirer une $\mathcal{C}$-structure de taille n par un générateur de paramètre x est :

$$\mathbb{P}_x(N=n) = \sum_{\gamma\in\mathcal{C}} \frac{x^{|\gamma|}}{C(x)} = \frac{c_n x^n}{C(x)}, \qquad \text{dans le cas ordinaire/non étiqueté,}$$

$$\mathbb{P}_x(N=n) = \sum_{\gamma\in\mathcal{C}} \frac{1}{|\gamma|!}\frac{x^{|\gamma|}}{\widehat{C}(x)} = \frac{c_n x^n}{n!\widehat{C}(x)}, \qquad \text{dans le cas exponentiel/ étiqueté.}$$

On en déduit les expressions suivantes pour l'espérance de la taille des structures :

$$\mathbb{E}_x(N) = x\frac{C'(x)}{C(x)} \qquad \text{ou} \qquad \mathbb{E}_x(N) = x\frac{\widehat{C}'(x)}{\widehat{C}(x)},$$

selon le cas, ordinaire, ou exponentiel, ainsi que le moment d'ordre 2 et par conséquent, l'écart type :

$$\mathbb{E}_x(N^2) = x^2\frac{C''(x)}{C(x)} + x\frac{C'(x)}{C(x)} \qquad \text{et} \qquad \sigma = \sqrt{\mathbb{E}_x(N^2) - \mathbb{E}_x(N)^2}.$$

Exemple 3. Les mots binaires sur l'alphabet $\{a, b\}$ sont définis par la spécification :

$$\mathcal{L} = \textsc{Seq}(\mathcal{Z}_a + \mathcal{Z}_b).$$

La série génératrice ordinaire associée est $L(x) = 1/(1-2x)$, avec comme singularité $\rho_L = 1/2$. La probabilité d'un mot w d'être tiré sous modèle de Boltzmann est alors $\mathbb{P}_x(w) = x^{|w|}(1-2x)$. L'espérance de la taille des mots engendrés sous ce modèle est $\mathbb{E}_x(N) = 2x/(1-2x)$. La figure 9.a représente la distribution des tailles des mots engendrés pour différentes valeurs de x. Pour la courbe bleue ($x = 0.49$), par exemple, l'espérance de la taille est $\mathbb{E}_{.49}(N) = 49$.

Exemple 4. Les partitions d'ensemble sont définies par la spécification :

$$\mathcal{P} = \textsc{Set}(\textsc{Set}_{\geq 1}(\mathcal{Z})).$$

La série génératrice exponentielle associée est $L(x) = e^{e^x-1}$. La probabilité pour une partition p d'être tirée sous modèle de Boltzmann est alors $\mathbb{P}_x(p) = x^{|p|}e^{1-e^x}$. Enfin, l'espérance de la taille des partitions engendrées sous ce modèle est $\mathbb{E}_x(N) = xe^x$. La figure 9.b représente la distribution des tailles des partitions engendrées pour différentes valeurs de x. Pour la courbe jaune ($x = 3.5$), l'espérance de la taille est $\mathbb{E}_{3.5}(N) = 115.9$.

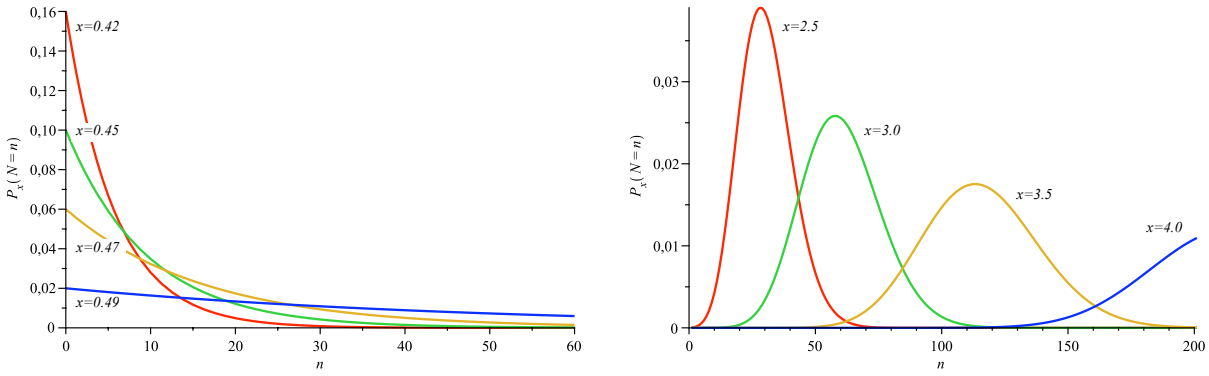


FIG. 9 – À gauche, la distribution des tailles de mots binaires sous modèle de Boltzmann ordinaire ; à droite, la distribution des tailles de partitions d'ensemble sous modèle de Boltzmann exponentiel. Pour chaque taille donnée en abscisse, les points correspondent aux probabilités de tirer une structure de cette taille, pour différentes valeurs du paramètre.

Si χ est un paramètre sur les $\mathcal{C}$-structures, alors la probabilité de tirer une $\mathcal{C}$-structure γ telle que $\chi(\gamma) = k$, sous le modèle de Boltzmann de paramètre x est :

$$\mathbb{P}_x(\chi = k) = \frac{1}{C(x)} \sum_{n \geq 0} c_{n,k} x^n \qquad \text{ou} \qquad \mathbb{P}_x(\chi = k) = \frac{1}{\widehat{C}(x)} \sum_{n \geq 0} c_{n,k} \frac{x^n}{n!}, \tag{5}$$

où $c_{n,k}$ est le nombre de $\mathcal{C}$-structures de taille n dont la paramètre vaut k, *i.e.*, le coefficient de $u^k x^n$ dans la série bivariée $C(x,u)$ associée à la paire $\langle \mathcal{C}, \chi \rangle$. La série génératrice de probabilités $p_x(u)$ de χ est alors :

$$p_x(u) = \sum_k \mathbb{P}_x(\chi = k) u^k = \frac{C(x,u)}{C(x)} \qquad \text{ou} \qquad p_x(u) = \frac{\widehat{C}(x,u)}{\widehat{C}(x)}. \tag{6}$$

3.2 Algorithmes

Le modèle de Boltzmann ainsi défini, intéressons-nous aux générateurs proprement dits. Comme nous l'avons déjà souligné, la méthode récursive et la méthode de Boltzmann ont pour point commun de s'appuyer sur la décomposition récursive des classes combinatoires. Nous garderons donc le cadre de travail des classes décomposables. Décrire des générateurs de Boltzmann pour l'ensemble de ces classes revient alors à donner le dictionnaire des générateurs pour les constructions admissibles. Contrairement à la méthode récursive, il n'est pas nécessaire de standardiser les spécifications. Dans un premier temps nous traiterons les constructions basiques : union, produit et séquence, qu'elles soient étiquetées ou non, puis les constructions d'ensemble et de cycle étiqueté. Les dernières constructions font l'objet d'un chapitre à part (chapitre 2).

3.2.1 Constructions basiques non étiquetées

Dans un premier temps, on ne s'intéresse qu'aux structures non étiquetées. Les générateurs pour les atomes (de taille 0 ou 1) sont triviaux : ils renvoient toujours un unique atome.

Union disjointe Considérons une classe combinatoire $\mathcal{C} = \mathcal{A} + \mathcal{B}$, définie comme l'union disjointe des classes $\mathcal{A}$ et $\mathcal{B}$. Le générateur de Boltzmann $\Gamma+[\mathcal{A}, \mathcal{B}](x)$ pour la classe $\mathcal{C}$ consiste à choisir de tirer une $\mathcal{A}$-structure avec probabilité $A(x)/C(x)$ (par un tirage de Bernoulli) ou une $\mathcal{B}$-structure avec probabilité $1 - A(x)/C(x) = B(x)/C(x)$:

$\Gamma+[\mathcal{A}, \mathcal{B}](x)$
  $p_{\mathcal{A}} \leftarrow \frac{A(x)}{C(x)}$
  si Bern($p_{\mathcal{A}}$) **alors**
    renvoyer $\Gamma\mathcal{A}(x)$
  sinon renvoyer $\Gamma\mathcal{B}(x)$

Une structure aléatoire de $\mathcal{C}$ est engendrée avec probabilité :

$$\mathbb{P}_x(\alpha, \alpha \in \mathcal{A}) = \frac{A(x)}{C(x)} \cdot \frac{x^{|\alpha|}}{A(x)} = \frac{x^{|\alpha|}}{C(x)}$$

si c'est une $\mathcal{A}$-structure et avec probabilité :

$$\mathbb{P}_x(\alpha, \alpha \in \mathcal{B}) = \frac{B(x)}{C(x)} \cdot \frac{x^{|\alpha|}}{B(x)} = \frac{x^{|\alpha|}}{C(x)}$$

si c'est une $\mathcal{B}$-structure.

Produit cartésien On considère désormais une classe $\mathcal{C}$ définie par $\mathcal{C} = \mathcal{A} \times \mathcal{B}$.

Le générateur est obtenu en formant un couple $\langle \alpha , \beta \rangle$, avec α et β tirés par des appels indépendants à $\Gamma\mathcal{A}(x)$ et $\Gamma\mathcal{B}(x)$. La notation $\rangle \langle$ est utilisée pour désigner un paire ou une séquence *ordonnée*.

$\Gamma\times[\mathcal{A}, \mathcal{B}](x)$
  renvoyer $(\Gamma\mathcal{A}(x), \Gamma\mathcal{B}(x))$

La taille de γ correspond à la somme des tailles de α et β, et par définition du produit des séries génératrices, la probabilité associée à γ dans le modèle de Boltzmann est :

$$\mathbb{P}_x(\gamma) = \frac{x^{|\alpha|}}{A(x)} \cdot \frac{x^{|\beta|}}{B(x)} = \frac{x^{|\gamma|}}{C(x)}.$$

Dans le modèle de Boltzmann, pour engendrer une structure correspondant à un produit cartésien, il n'est pas nécessaire de tirer aléatoirement les tailles des composantes du produit. Cette particularité permet d'améliorer nettement la complexité des générateurs de Boltzmann pour l'ensembles des classes décomposables.

Exemple 5. Les arbres binaires sont définis par la spécification :

$$\mathcal{T} = \mathcal{Z} + \mathcal{T}^2.$$

La série génératrice ordinaire associée est $T(x) = x + T^2(x) = \frac{1-\sqrt{1-4x}}{2}$, avec comme singularité $\rho_L = 1/4$. Le générateur est donné par l'algorithme suivant :

$\Gamma T(x)$
  si Bern($x/T(x)$) **alors**
    renvoyer une feuille
  sinon renvoyer $(\Gamma\mathcal{T}(x), \Gamma\mathcal{T}(x))$

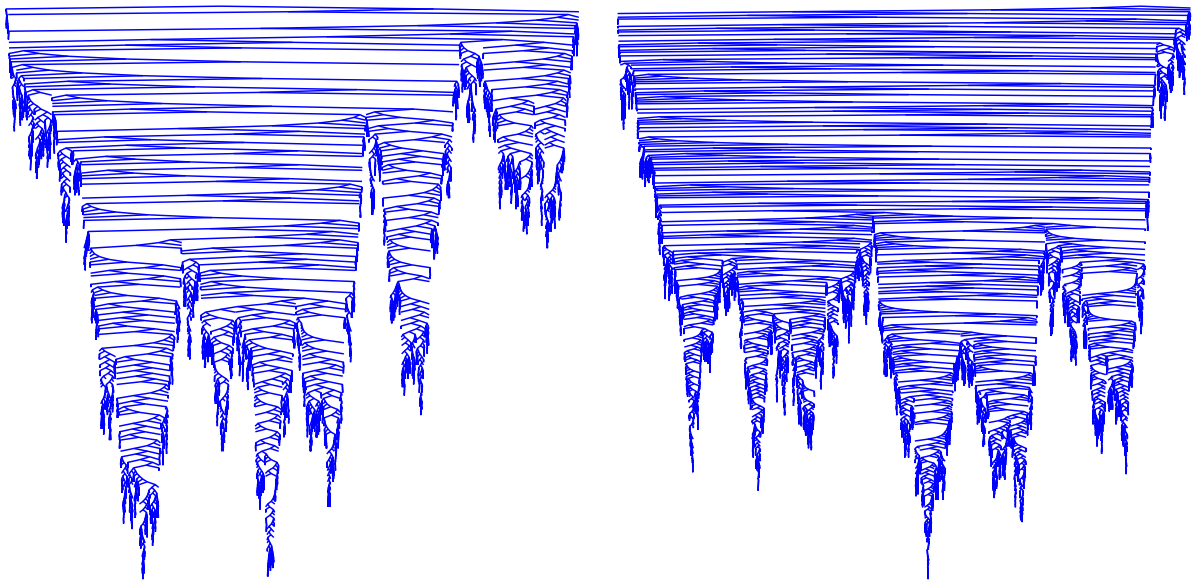


FIG. 10 – Deux arbres binaires planaires ayant respectivement 4469 et 8255 nœuds.

En choisissant comme paramètre la singularité $\frac{1}{4}$ de $T(x)$ (nous verrons, à la section 3.3.2, pourquoi ce choix est judicieux), voici les différentes tailles des premiers arbres engendrés par ce générateur :
$1, 106, 6, 4, 9, 3, 1, 15, 2, 4, 4, 1, 1, 25, 43, 1, 2, 1, 2, 2, 449, 13, 39, 1, 1, 1, 61, 28, 1, 1, 1, 1, 5, 1, 6, 1, 35620, 3, 1, ...$
La figure 10 présente deux arbres obtenus par $\Gamma\mathcal{T}(1/4)$.

Séquence Dans le cas de la séquence, on peut, dans un premier temps, imaginer un générateur utilisant la définition récursive :

$$\mathcal{C} = \textsc{Seq}(\mathcal{A}) \qquad \Rightarrow \qquad \mathcal{C} = \mathcal{E} + \mathcal{A} \cdot \mathcal{C}.$$

En utilisant les algorithmes précédents, on obtient le générateur récursif :

```
Γ⋆SEQ[A](x)
  p ← A(x)
  si Bern(p) alors
    renvoyer (ΓA(x), Γ⋆SEQ[A](x))
  sinon renvoyer ( )
```

Ce générateur fait des tirages de Bernoulli de paramètre $A(x)$ jusqu'à obtenir un échec. Le nombre de $\mathcal{A}$-structures engendrées par ce générateur suit donc une loi géométrique de paramètre $A(x)$. On rappelle que la loi géométrique de paramètre λ est définie par :

$$\mathbb{P}(X = k) = (1 - \lambda)\lambda^k.$$

On peut donc réécrire le générateur, en commençant par tirer aléatoirement le nombre d'éléments qui composent la séquence, suivant la loi géométrique $\text{Geom}(A(x))$, puis en faisant des appels indépendants au générateur $\Gamma\mathcal{A}(x)$:

```
ΓSEQ[A](x)
  k ← Geom(A(x))
  renvoyer le k-uplet (ΓA(x), ..., ΓA(x))
```

En combinant ces trois générateurs (et les atomes), et en utilisant éventuellement la récursion, on peut traiter toutes les classes définies par des grammaires hors-contexte rationnelles et algébriques. On couvre ainsi un grand nombre de classes combinatoires "classiques".

Exemple 6. Un générateur $\Gamma\mathcal{L}(x)$ pour les mots de $\{a, b\}^*$ (voir exemple 3) revient à tirer la taille du mot à engendrer par un appel à $\texttt{Geom}(2x)$, puis à choisir chacune des lettres qui le composent à pile ou face. La figure 11 donne un exemple de mot obtenu par $\Gamma\mathcal{L}(0.499)$. Les a sont représentés par des □ et les b par des ■.

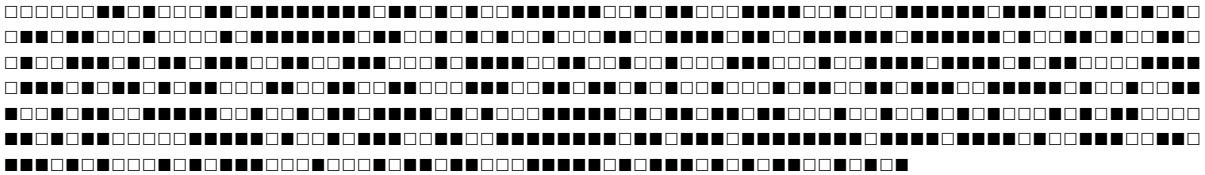


FIG. 11 – Mot binaire de longueur 527 sur l'alphabet $\{a, b\}$.

3.2.2 Constructions étiquetées

Union, produit, séquence Nous venons de voir comment construire des générateurs pour des structures non étiquetées à partir des constructions de base. Pour engendrer des structures étiquetées, on procède de la même façon ; à ceci près que l'on utilise les valeurs des séries génératrices exponentielles et non ordinaires. Les étiquettes des atomes, quant à elles, sont tirées au hasard, *après* la phase de génération proprement dite. Plus précisément, pour une structure γ de taille n, on tire une permutation aléatoire des étiquettes comprises entre 1 et n, que l'on "jette" ensuite sur les atomes composant γ. On ne cherche pas à étiqueter les atomes pendant le processus de génération : il faudrait utiliser des produits étiquetés, trop coûteux en termes de complexité. L'étiquetage à posteriori conserve l'uniformité, étant donné que pour une taille fixée, tous les étiquetages sont équiprobables.

Ensemble et cycle Pour conclure l'étude des constructions étiquetées, il nous reste à donner les algorithmes pour les constructions d'ensemble et de cycle étiquetés. Bien que ce soient des constructions différentes, on va élaborer les algorithmes sur le même modèle que celui de la séquence. On considère les classes $\mathcal{S} = \textsc{Set}(\mathcal{A})$ et $\mathcal{C} = \textsc{Cyc}(\mathcal{A})$. On commence par choisir, suivant la distribution appropriée, le nombre d'éléments k_s (resp. k_c) qui composent l'ensemble (resp. le cycle). Ensuite on fait k_s (resp. k_c) appels indépendants au générateur $\Gamma\mathcal{A}(x)$. D'après l'équation (5) sous modèle de Boltzmann de paramètre x, la probabilité qu'un ensemble appartenant à $\mathcal{S}$ ait k éléments dans $\mathcal{A}$ est :

$$\mathbb{P}_x(K = k) = \frac{1}{\widehat{S}(x)} \sum_{n\geq 0} s_{n,k} \frac{x^n}{n!} = \frac{\widehat{S}_k(x)}{\widehat{S}(x)} = e^{-\hat{A}(x)} \frac{\hat{A}(x)^k}{k!},$$

et la probabilité qu'un cycle appartenant à $\mathcal{C}$ ait k éléments dans $\mathcal{A}$ est :

$$\mathbb{P}_x(K = k) = \frac{1}{\widehat{C}(x)} \sum_{n\geq 0} c_{n,k} \frac{x^n}{n!} = \frac{\widehat{C}_k(x)}{\widehat{C}(x)} = \frac{1}{\log(1 - \hat{A}(x))^{-1}} \frac{\hat{A}(x)^k}{k},$$

où $\widehat{S}_k(x)$ et $\widehat{C}_k(x)$ sont respectivement les séries génératrices des ensembles et des cycles composés de k $\mathcal{A}$-structures. Le nombre de $\mathcal{A}$-structures composant un ensemble suit donc une loi

de Poisson de paramètre $\hat{A}(x)$. On rappelle que la loi de Poisson de paramètre λ est définie par :

$$\mathbb{P}(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}.$$

De la même façon, le nombre de $\mathcal{A}$-structures qui composent un cycle suit une loi logarithmique de paramètre $\hat{A}(x)$. On rappelle que la loi logarithmique de paramètre λ est définie par :

$$\mathbb{P}(X = k) = \frac{1}{\log(1-\lambda)^{-1}}\frac{\lambda^k}{k}.$$

On obtient finalement les générateurs pour les ensembles et les cycles de $\mathcal{A}$-structures suivants :

$\Gamma\textsc{Set}[\mathcal{A}](x)$
  $k \leftarrow \texttt{Poiss}(A(x))$
  `renvoyer le` k`-uplet` $\langle \Gamma\mathcal{A}(x), \ldots, \Gamma\mathcal{A}(x) \rangle$

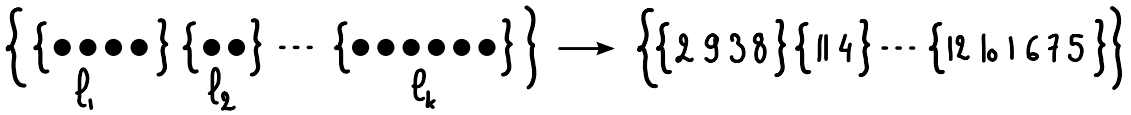
$\Gamma\textsc{Cyc}[\mathcal{A}](x)$
  $k \leftarrow \texttt{Loga}(A(x))$
  `renvoyer le` k`-uplet` $(\Gamma\mathcal{A}(x), \ldots, \Gamma\mathcal{A}(x))$

Ces deux constructions viennent compléter le dictionnaire des générateurs de Boltzmann proposés dans [DFLS04]. On peut désormais assembler des générateurs de Boltzmann pour l'ensemble des classes combinatoires spécifiables étiquetées. Voici, par exemple, un générateur de partitions d'ensemble :

Exemple 7. Le générateur $\Gamma\mathcal{P}(x)$ pour les partitions d'ensemble (voir exemple 4) se déduit de la spécification donnée plus haut et des règles précédentes :

$\Gamma P(x)$
  $k \leftarrow \texttt{Poiss}(e^x - 1))$
  pour i **de** 1 **à** k **faire**
    $\ell_i \leftarrow \texttt{Poiss}_{\geq 1}(x)$
  `renvoyer le` k`-uplet` $\langle \ell_1, \ell_2, \ldots, \ell_k \rangle$

Le résultat est une liste des tailles des sous-ensembles qui composent la partition d'ensemble ; ce que l'on a obtenu est donc le squelette p de la partition et les éléments n'ont pas encore d'étiquettes. Pour obtenir un objet étiqueté, il suffit de tirer une permutation aléatoire dont la longueur correspond au nombre d'éléments de p, avec l'algorithme $RANPER(n)$, par exemple (voir section 2.1) et d'étiqueter les atomes de p dans l'ordre de la permutation.



3.2.3 Générateur pour les lois de probabilités

Les algorithmes que nous avons présentés ici font appel à des générateurs de lois de probabilité, qu'elles soient géométrique, de Poisson ou logarithmique. Ils peuvent tous trois être obtenus à partir d'un même schéma itératif générique. L'idée est de découper l'intervalle $[0, 1]$ selon les probabilités $\mathbb{P}(K = k)$ définissant chaque loi (ces probabilités sont rappelées par la ligne 1 de la table 4), puis de tirer un réel aléatoire dans $[0, 1]$ et de chercher séquentiellement

géométrique $-$ Geom(λ)	Poisson $-$ Poiss(λ)	Poisson $\neq 0$ $-$ Poiss$_{\geq 1}$(λ)	logarithmique $-$ Loga(λ)
$\mathbb{P}(X=k) = (1-\lambda)\lambda^k$	$\mathbb{P}(X=k) = e^{-\lambda}\frac{\lambda^k}{k!}$	$\mathbb{P}(X=k) = e^{-\lambda}\frac{\lambda^k}{k!}$	$\mathbb{P}(X=k) = \frac{1}{\log(1-\lambda)^{-1}}\frac{\lambda^k}{k}$
$p_0 = (1-\lambda)$ $p_{k+1} = \lambda p_k$	$p_0 = e^{-\lambda}$ $p_{k+1} = \lambda p_k \frac{1}{k+1}$	$p_1 = \frac{\lambda}{e^{\lambda}-1}$ $p_{k+1} = \lambda p_k \frac{1}{k+1}$	$p_0 = 1/(\log(1-\lambda)^{-1})$ $p_{k+1} = \lambda p_k \frac{k}{k+1}$

Tab. 4 – Lois de probabilités usuelles.

dans quel sous-intervalle il se trouve. Ce découpage peut être obtenu itérativement à partir des valeurs initiales données par la ligne 3 de la table 4 et de l'incrément donné par la ligne 4. Le schéma séquentiel est le suivant :

GenLoi(x)
  $U \leftarrow$ `random();`
  $S \leftarrow 0$; $k \leftarrow 0$;
  tant que $U < S$ **faire**
    $S \leftarrow S + p_k$;
    $k \leftarrow k + 1$;
  `renvoyer` k

En utilisant ce schéma (en initialisant k à 1, dans le cas de la loi de Poisson non nulle), on obtient les générateurs pour les quatre lois répertoriées[8] dans la table 4. Chaque algorithme ainsi obtenu à une complexité arithmétique linéaire en la taille de la valeur renvoyée.

Pour engendrer une structure aléatoire, un générateur assemblé à partir des règles de construction données dans cette section, suit un arbre de syntaxe dont la taille est celle de la structure elle-même. Pour chaque nœud de cet arbre, le générateur fait éventuellement appel à l'un des générateurs de lois de probabilités donnés ci-dessus, afin de calculer les valeurs intermédiaires correspondant au nombre de fils du nœud courant dans l'arbre de syntaxe. Ce faisant, pour engendrer une structure de taille n, la somme de ces valeurs intermédiaires est trivialement bornée par la taille de l'objet. La complexité globale de la génération est donc linéaire en la taille de la structure engendrée.

Théorème 2 (Générateurs de Boltzmann libres [DFLS04])**.** *Soit $\mathcal{C}$ une classe combinatoire étiquetée spécifiée à l'aide des constructions :*

$$\{\mathcal{E}, \mathcal{Z}, +, \times, \textsc{Seq}, \textsc{Set}, \textsc{Cyc}\},$$

ou non étiquetée spécifiée à l'aide des constructions :

$$\{\mathcal{E}, \mathcal{Z}, +, \times, \textsc{Seq}\}.$$

Soit x une valeur de paramètre cohérente, i.e.*, dans $]0, \rho_{\mathcal{C}}]$. En supposant donné un oracle qui fournit les valeurs des séries génératrices associées à $\mathcal{C}$ au point x, alors le générateur de Boltzmann $\Gamma\mathcal{C}(x)$, assemblé à partir des algorithmes présentés ci-dessus, a une complexité réelle-arithmétique linéaire en la taille de la structure qu'il engendre.*

[8] La loi de Poisson non nulle servira au prochain chapitre.

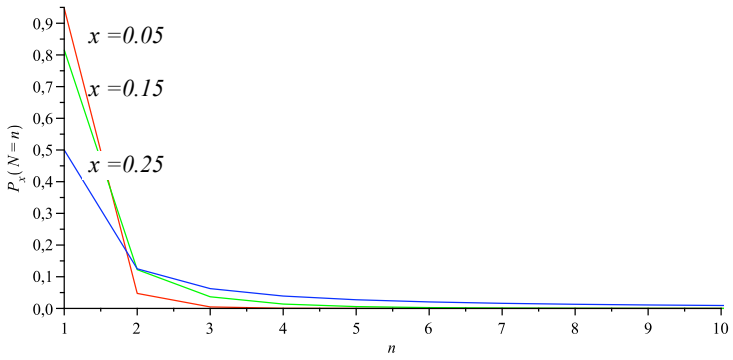


FIG. 12 – Distribution de la taille des arbres binaires sous modèle de Boltzmann.

3.3 Implantation et applications

Un certains nombre de points restent en suspens, pour la mise en œuvre effective de ces générateurs de Boltzmann, et notamment, la question du réglage du paramètre pour obtenir des structures de taille fixée (exacte ou approchée). C'est ce dont nous discutons brièvement dans cette section. Dans certains cas, ce réglage n'est pas suffisant pour obtenir des générateurs en taille approchée ou exacte efficaces. Nous présentons donc également des techniques données dans [DFLS04] pour améliorer cette complexité.

3.3.1 Génération en taille approchée ou exacte

La génération à taille exacte est le modèle utilisé dans les deux premières sections de ce chapitre. La taille des structures à engendrer est passée en paramètre du générateur. Tirer des structures de taille approchée consiste simplement à accepter une tolérance sur cette taille. On fait des tirages dans un ensemble de structures, non plus de taille n, mais dont la taille se situe dans l'intervalle $[n(1-\varepsilon), n(1+\varepsilon)]$. Bien sûr, on souhaite conserver l'équiprobabilité des structures pour chaque taille. Ce type de générateur peut être obtenu à partir d'un générateur de Boltzmann libre auquel on associe une procédure de rejet, comme celle présentée à la section 1.2. L'algorithme obtenu est un générateur $\Gamma\mathcal{C}(x,n)$ qui dépend de la taille voulue et de la valeur x.

Afin d'avoir une procédure de rejet efficace, *i.e.*, qui ne rejette pas trop de structures, il est important de bien choisir le paramètre x. Nous avons vu plus haut que l'on peut exprimer l'espérance de la taille des structures engendrées par un générateur de Boltzmann en fonction de x. Régler le paramètre x de façon à avoir de bonnes chances de tirer une structure de taille n (ou proche de n), consiste à choisir une valeur de x telle que l'espérance de la taille est n, c'est-à-dire la solution $x_n \in]0, \rho]$ de l'équation :

$$x\frac{C'(x)}{C(x)} = n.$$

Le générateur ainsi obtenu n'est plus paramétré par x, mais par n et sa complexité n'est déterminée que par cette valeur et par la forme de la distribution de la classe combinatoire concernée. Dans [DFLS04], les auteurs classifient les distributions des tailles sous modèle de Boltzmann en trois types principaux : concentré, plat ou piqué.

- Une distribution **concentrée**, typiquement celle des partitions d'ensemble (voir figure 9), est un cas de figure où la procédure de rejet est très efficace. En effet, dans ce cas là,

on tire une structure de taille approximativement n en un seul essai avec une probabilité tendant vers 1 quand $n \to \infty$ (voir exemple ci-dessous). Pour obtenir une structure de taille exactement n, le rejet a une complexité $O(n)$. Le générateur en taille approchée (resp. exacte) à donc une complexité globale $O(n)$ (resp. $O(n^2)$).

- Les distributions **plates**, comme par exemple celle des mots de $\{a, b\}^*$ (voir figure 9), sont un cas moins favorable a priori, mais pour lequel on obtient les mêmes complexités théoriques (mais avec des constantes différentes). On les appelle ainsi parce qu'elles s'aplatissent lorsqu'on se rapproche de la singularité. Le rejet pour tirer des structures en taille approchée est alors en $O(1)$, parce qu'intuitivement, quelle que soit la taille visée, la probabilité de tirer une structure de cette taille est sensiblement la même.
- Enfin, les distributions **piquées** sont plus problématiques, car elles concentrent tout le poids sur les structures de très petite taille, même pour une valeur de x proche de la singularité. C'est la forme typique d'une distribution des tailles pour des arbres, et notamment les arbres binaires, dont la distribution est donnée par la figure 12. Le "remède" consiste à pointer les structures (voir section 2.3). En effet, la distribution des tailles des structures pointées redevient alors une distribution plate[9].

On trouvera une discussion beaucoup plus détaillée dans [DFLS04], où les auteurs décrivent notamment comment déterminer la complexité de la génération avec rejet en fonction du type de singularité des séries génératrices.

Exemple 8. La table ci-dessous donne les tailles des mots binaires et des partitions d'ensembles engendrées par un générateur de Boltzmann pour différentes valeurs de l'espérance de la taille (et donc du paramètre associé). Pour les partitions d'ensemble, qui ont une distribution concentrée, on indique en bleu les tailles qui sont dans un intervalle de tolérance de 5% autour de la taille voulue.

n	x_n	tailles de mots binaires engendrés
10	$\frac{5}{11}$	8, 51, 37, 17, 24, 43, 10, 27, 6, 7, 0, 4, 7, 7, 7, 5, 4, 4, 3, 7, 0, 21, 2, 10, 7, 4, 14, 13, 6, 13, ...
100	$\frac{50}{101}$	76, 59, 115, 6, 33, 72, 68, 455, 264, 44, 302, 85, 22, 409, 100, 49, 21, 494, 293, 240, 70, ...
1000	$\frac{500}{1001}$	1322, 2221, 165, 50, 55, 988, 773, 46, 376, 2253, 463, 294, 742, 261, 609, 1900, 620, ...
n	x_n	tailles de partitions d'ensemble engendrées
100	3.385	104, 119, 144, 98, 76, 102, 137, 98, 110, 89, 63, 133, 112, 79, 65, 112, 75, 67, 81, 87, ...
1000	5.249	965, 928, 1146, 916, 986, 1093, 977, 951, 1053, 995, 990, 1096, 1048, 1010, 1042, ...
10^4	7.232	9566, 9960, 10298, 10017, 10337, 10493, 10410, 10409, 9934, 9641, 9909, 9669, 10335, 10016, 10000, 10000, 9736, 10641, 9982, 10188, 9984, 9979, 9897, 10465, 9756, ...

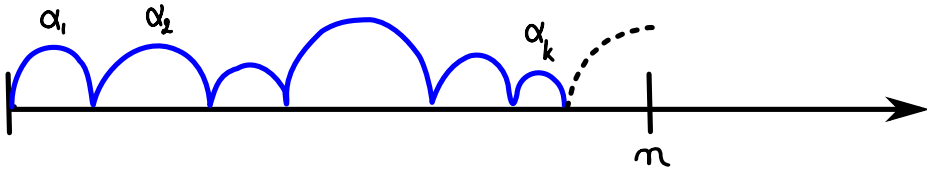
3.3.2 Générateurs singuliers

L'espérance de la taille des structures engendrées sous modèle de Boltzmann est une fonction qui croît avec la valeur de x. Pour obtenir des structures de grande taille avec un générateur $\Gamma\mathcal{C}(x)$, une stratégie consiste donc à se rapprocher aussi près que possible de la singularité dominante ρ de la série $C(x)$. Nous avons également évoqué la possibilité de choisir ρ comme valeur de paramètre lorsque la série converge en ρ. Ce choix donne lieu à ce que l'on appelle des générateurs de Boltzmann *singuliers* dont voici deux exemples aux propriétés intéressantes.

Séquences super-critiques Une séquence $\mathcal{C}$ de $\mathcal{A}$-structures définie par $\mathcal{C} = \textsc{Seq}(\mathcal{A})$ est super-critique si $\rho_A > \rho_C$. Intuitivement, cela signifie que la singularité de la série $C(x)$ provient de la séquence et non pas des structures qui la composent. Cette condition implique

[9] Il est éventuellement nécessaire de pointer plusieurs fois la spécification, pour aplatir la distribution.

notamment que $A(\rho_C) = 1$. L'espérance de la taille des séquences produites par le générateur singulier $\Gamma\mathcal{C}(\rho_C)$ est alors infinie puisque le nombre d'éléments qui composent une telle séquence suit une loi géométrique de paramètre 1. Néanmoins, par *interruption anticipée* du générateur singulier, il est possible d'engendrer des séquences de taille approximativement n : pour une taille n fixée, l'idée est d'engendrer la séquence comme un produit $\alpha_1\alpha_2\alpha_3\ldots$ apparemment infini de $\mathcal{A}$-structures et de stopper le générateur dès que la taille n est dépassée. Il suffit ensuite de garder la séquence ainsi obtenue privée de sa dernière $\mathcal{A}$-structure[10] :



La probabilité d'engendrer une séquence $c = \alpha_1\alpha_2\ldots\alpha_k$ appartenant à $\mathcal{C}$ par ce procédé est :

$$\mathbb{P}_{\rho_C}(\gamma) = \frac{\rho_C^{|\alpha_1|}}{A(\rho_C)} \cdot \frac{\rho_C^{|\alpha_2|}}{A(\rho_C)} \cdot \cdots\cdot \frac{\rho_C^{|\alpha_k|}}{A(\rho_C)} \sum_{\substack{a\in\mathcal{A}\\ |a|>n-|\gamma|}} \mathbb{P}_{\rho_C}(a) \;=\; \rho_C^{|\gamma|} \cdot \mathbb{P}_{\mathcal{A}}(N > n - |\gamma|),$$

où $\mathbb{P}_{\mathcal{A}}(N > n-|\gamma|)$ est la probabilité de tirer une $\mathcal{A}$-structure dont la taille est supérieure à $n-|\gamma|$. La probabilité d'une séquence ne dépend donc que de sa taille et du n choisi, et l'uniformité est bien conservée. Dans [DFLS04], les auteurs prouvent également que l'on produit ainsi une séquence de taille $n + O(1)$ en un seul essai, presque sûrement. Ce résultat étonnant vient du fait que les $\mathcal{A}$-structures engendrées sont de taille relativement *petite*. Associé à une procédure de rejet, ce générateur a une complexité $O(n)$ en taille exacte.

Générateur singulier plafonné Les structures arborescentes, qui forment un sous-ensemble non négligeable des structures spécifiables, semblent être celles pour lesquelles la méthode de Boltzmann est la moins efficace. Néanmoins, l'utilisation de générateurs singuliers permet, là encore, d'améliorer l'efficacité de la génération, et ce, sans passer par l'astuce du pointage. Contrairement au cas précédent, il n'est pas possible d'interrompre la génération dès que la taille voulue est atteinte, car cela biaise complètement la distribution. De plus, en choisissant comme paramètre la singularité de la série génératrice, l'espérance de la taille des arbres devient infinie. On doit alors envisager une procédure qui plafonne la taille des structures engendrées. Plus précisément, cela consiste à fixer une borne sur la taille des structures à engendrer, et à garder, pour chaque génération, un compteur des atomes produits ; si le compteur dépasse la borne fixée, on jette la structure en cours et on recommence. Cette technique s'apparente à celle utilisée par l'algorithme florentin (voir section 1.2), étant donné que l'on anticipe le rejet en interrompant le processus dès que la structure en cours de fabrication devient manifestement trop grande. Pour une classe définie par une spécification récursive irréductible (son graphe de dépendance est fortement connexe) et apériodique (on peut engendrer des structures de toutes les tailles), la complexité du générateur en taille approchée est $O(n)$. Cela vient essentiellement du fait que la somme des tailles des structures produites et rejetées (trop grandes ou trop petites) est en $O(n)$. Le générateur en taille exacte, quant à lui, a une complexité quadratique.

[10] Ce dernier point diffère de ce qui est présenté dans [DFLS04]. En effet, dans l'algorithme d'origine, on garde la dernière $\mathcal{A}$-structure engendrée. Mais, ce faisant, on observe un léger défaut d'uniformité dû au fait que certaines structures de taille $n + \varepsilon$ ont alors une probabilité nulle d'être engendrées.

3.4 Conclusion

La génération aléatoire sous modèle de Boltzmann est une méthode générique basée sur la décomposition récursive des structures combinatoires. Mais contrairement à la méthode récursive, le modèle de Boltzmann autorise une génération en taille approchée qui permet un gain substantiel d'un point de vue de la complexité : la phase de génération devient linéaire dès qu'une légère tolérance sur la taille des structures engendrées est acceptée. Cette amélioration de la complexité rend possible la génération de structures de très grande taille (pouvant aller jusqu'à plusieurs millions) qui n'étaient pas accessibles auparavant. L'introduction de ce modèle offre de nombreuses perspectives de développement. La suite de ce mémoire est consacrée à l'étude de certaines d'entre elles.

Un certain nombre de travaux basés sur la méthode de Boltzmann ont déjà été publiés. Dans [Fus05, Fus07], par exemple, Fusy présente un générateur de Boltzmann de graphes planaires dont la complexité est linéaire en taille approchée. Il utilise pour cela les règles présentées ci-dessus et introduit un nouveau générateur de Boltzmann pour la règle de substitution. L'algorithme complet est obtenu en utilisant les décompositions successives des graphes planaires en graphes de plus faible connectivité. Le travail de Roussel [Rou08] introduit également un générateur de Boltzmann pour une autre construction combinatoire : l'opérateur boîte, qui permet notamment d'engendrer des arbres croissants. Bassino *et al.* [BN07, BDN07, BDN08] utilisent la méthode de Boltzmann pour engendrer efficacement des automates finis déterministes et accessibles. Leurs algorithmes sont basés sur des bijections entre ces automates et des partitions d'un ensemble à $kn + 1$ éléments en k parts. La distribution concentrée des partitions d'ensembles permet d'obtenir un générateur efficace pour cette sous-classe de partitions en associant une procédure de rejet à leur générateur de Boltzmann. Enfin, dans [DS07] Darrrasse et Soria utilisent des générateurs de Boltzmann non seulement pour observer la distribution des degrés dans les RANS[11], mais également comme un outil pour l'analyse de cette distribution. Parallèlement, Bernasconi, Panagiotou, Steger et Weißl [PW07, BPS08b, BPS08a, PS09] analysent le comportement des générateurs de Boltzmann pour en extraire des informations sur les propriétés de certains graphes aléatoires (avec des contraintes structurelles).

[11] Les RANS (random Apollonian network structures) sont un modèle de graphes aléatoires utilisés pour modéliser des réseaux d'interactions.

Première partie

Modèle de Boltzmann non étiqueté

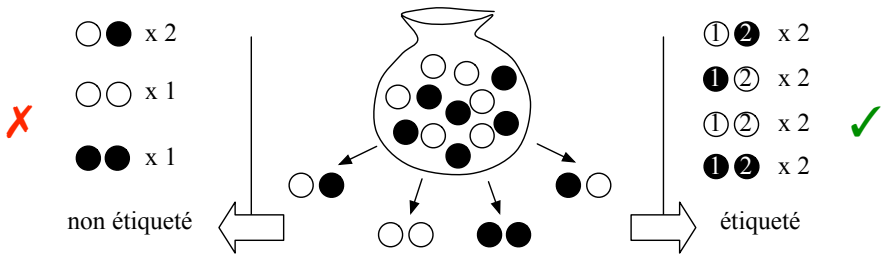
Chapitre 2

Algorithmes génériques pour les opérateurs de Pólya

Sommaire

L'objet de ce chapitre est de compléter le dictionnaire des générateurs de Boltzmann proposé par Duchon *et al.* dans [DFLS04]. Pour couvrir l'ensemble des structures combinatoires spécifiables à partir des constructions admissibles de la table 1 du chapitre précédent (page 22), nous devons notamment traiter le cas des ensembles, multi-ensembles et cycles non-étiquetés, auxquels peuvent s'ajouter des contraintes sur le nombre d'éléments qui les composent. On regroupe en général ces constructions sous le nom d'*opérateurs de Pólya*[12].

Les structures non étiquetées que l'on peut spécifier à l'aide d'ensembles et de cycles peuvent présenter des symétries qui les rendent naturellement plus difficiles à engendrer de façon uniforme. Contrairement aux structures étiquetées et aux structures non étiquetées "rigides" (produits et séquences), l'indépendance des tirages ne garantit plus l'uniformité :



Dans ce chapitre, nous verrons que l'on peut corriger ce phénomène en pondérant les tirages et ainsi produire des algorithmes non biaisés pour engendrer des structures non étiquetées, en utilisant notamment une construction auxiliaire : la *diagonale*.

[12] Ces opérateurs ont, par exemple, été étudiés dans [PR87].

Dans un premier temps, nous présentons pour chaque nouvelle construction, l'algorithme correspondant ainsi qu'un certain nombre d'exemples de mise en œuvre pour des classes de structures combinatoires classiques. Pour compléter cette étude, nous fournissons quelques données expérimentales sur le comportement des générateurs implantés. Ce chapitre reprend les résultats présentés dans [FFP07].

1 Générateurs pour les constructions non étiquetées

Le but de cette section est de montrer que l'on peut désormais automatiquement fournir un générateur de Boltzmann pour toute classe combinatoire décomposable[13] :

Théorème 3. *Soit $\mathcal{C}$ une classe combinatoire non étiquetée, spécifiable à l'aide des constructions suivantes :*

$$\{+, \times, \textsc{Seq}, \Delta_k, \textsc{MSet}, \textsc{MSet}_k, \textsc{Cyc}, \textsc{Cyc}_k\}.$$

Soit x une valeur de paramètre cohérente, i.e.*, dans $]0, \rho_{\mathcal{C}}[$. En supposant donné un oracle qui fournit les valeurs des séries génératrices associées à $\mathcal{C}$ au point x, alors il existe un processus effectif qui produit le générateur de Boltzmann $\Gamma\mathcal{C}(x)$ tel que, dans le pire cas, la complexité réelle-arithmétique de la génération est linéaire en la taille de la structure engendrée.*

La preuve de ce théorème est donnée par l'assemblage des propositions 3 à 9. Le résultat de complexité vient du fait que, comme pour les autres constructions, la complexité d'un tirage est bornée par la taille de l'arbre de syntaxe associé à la structure engendrée (cf. page 37). Dans le cas des ensembles sans répétitions (PSet), la linéarité n'est pas toujours garantie, nous présentons donc ce résultat en marge du théorème principal.

1.1 Multi-ensemble

1.1.1 Multi-ensemble à deux éléments

Nous commençons par expliquer comment concevoir un générateur de Boltzmann pour un multi-ensemble composé d'exactement deux éléments. Cette construction, plus simple que le multi-ensemble général, nous permet d'introduire la notion de *diagonale*, essentielle au traitement des constructions non étiquetées. Nous utiliserons la notation suivante pour désigner un k-uplet non ordonné constitué de k copies de la même structure α :

$$\alpha^{\odot k} = \langle \underbrace{\alpha, \ldots, \alpha}_{k \text{ copies}} \rangle.$$

Paires non ordonnées La classe $\textsc{MSet}_2(\mathcal{A})$ contient tous les multi-ensembles composés de deux $\mathcal{A}$-structures, ou plus simplement, les paires non ordonnées de $\mathcal{A}$-structures. Afin d'écrire la série génératrice associée, on utilise un nouvelle construction : la diagonale d'une classe combinatoire.

Définition 6. *La* diagonale $\Delta_k\mathcal{A}$ *à k éléments (k-diagonale) d'une classe de structures combinatoires $\mathcal{A}$ est l'ensemble des k-uplets de $\mathcal{A}$-structures identiques :*

$$\Delta_k\mathcal{A} = \{\alpha^{\odot k} \mid \alpha \in \mathcal{A}\}.$$

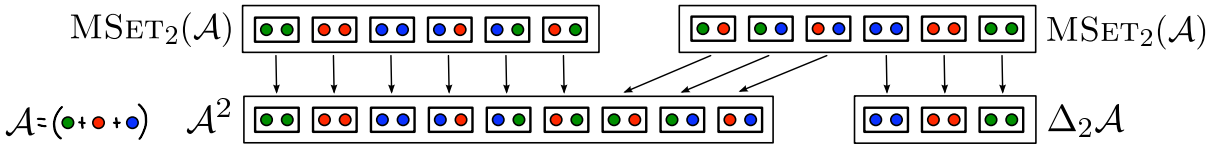
La série génératrice associée est $\Delta_k A(x) = A(x^k)$.

[13] Au sens de la définition 1, page 21, chapitre précédent.

L'isomorphisme suivant nous permet de relier les paires non ordonnées de $\mathcal{A}$-structures à la diagonale $\Delta_2\mathcal{A}$ et aux couples ordonnés de $\mathcal{A}$-structures ($\mathcal{A}^2$) :

$$2\,\mathrm{MSET}_2(\mathcal{A}) \cong \mathcal{A}^2 + \Delta_2\mathcal{A}. \tag{1}$$

On interprète cette relation en ordonnant les éléments des paires de la classe $\mathcal{C} = \mathrm{MSET}_2(\mathcal{A})$. Cela revient à fabriquer deux copies disjointes de $\mathcal{C}$, l'une contenant les paires transformées en couples arbitrairement ordonnés et l'autre contenant les couples symétriques. Ainsi, on obtient tous les couples $(\alpha_1, \alpha_2)_{\alpha_1 \neq \alpha_2}$ de $\mathcal{A}$-structures différentes appartenant à $\mathcal{A}^2$. En revanche, les couples de $\mathcal{A}$-structures identiques (α, α) apparaissent en double, on ajoute donc la diagonale $\Delta_2\mathcal{A}$ pour rééquilibrer (exemple ci-dessous).



De cette relation, on peut déduire la série génératrice de la classe $\mathcal{C}$:

$$C(z) = \frac{1}{2}A^2(z) + \frac{1}{2}A(z^2). \tag{2}$$

Générateur Nous allons utiliser la relation (1) et cette série génératrice pour écrire un générateur pour la construction MSET_2. Pour cela, nous avons besoin du générateur correspondant à la diagonale :

```
ΓΔ_k A(x)
    α ← ΓA(x^k)
    renvoyer le k-uplet ⟨α, ..., α⟩
```

Proposition 3. *L'algorithme* $\Gamma\Delta_k\mathcal{A}(x)$ *est un générateur de Boltzmann pour la classe* $\Delta_k\mathcal{A}$.

Démonstration. Un élément aléatoire $\delta = (\alpha, \ldots, \alpha)$ de $\Delta_k\mathcal{A}$ est engendré avec une probabilité :

$$\mathbb{P}_x(\delta) = \frac{x^{k|\alpha|}}{A(x^k)} = \frac{x^{|\delta|}}{\Delta_k A(x)}. \qquad \square$$

Nous avons désormais tous les outils pour assembler un générateur de multi-ensembles à deux éléments. À partir de la relation (1) et des générateurs pour l'union, le produit cartésien (voir chapitre précédent) et la diagonale, on obtient l'algorithme suivant :

```
ΓMSET2[A](x)
    si Bern( (1/2) A²(x)/C(x) ) = 1
        renvoyer ⟨ΓA(x), ΓA(x)⟩
    sinon
        α ← ΓA(x²)
        renvoyer ⟨α, α⟩
```

Proposition 4. $\Gamma\mathrm{MSET}_2[\mathcal{A}](x)$ *est un générateur de Boltzmann pour la classe* $\mathrm{MSET}_2(\mathcal{A})$.

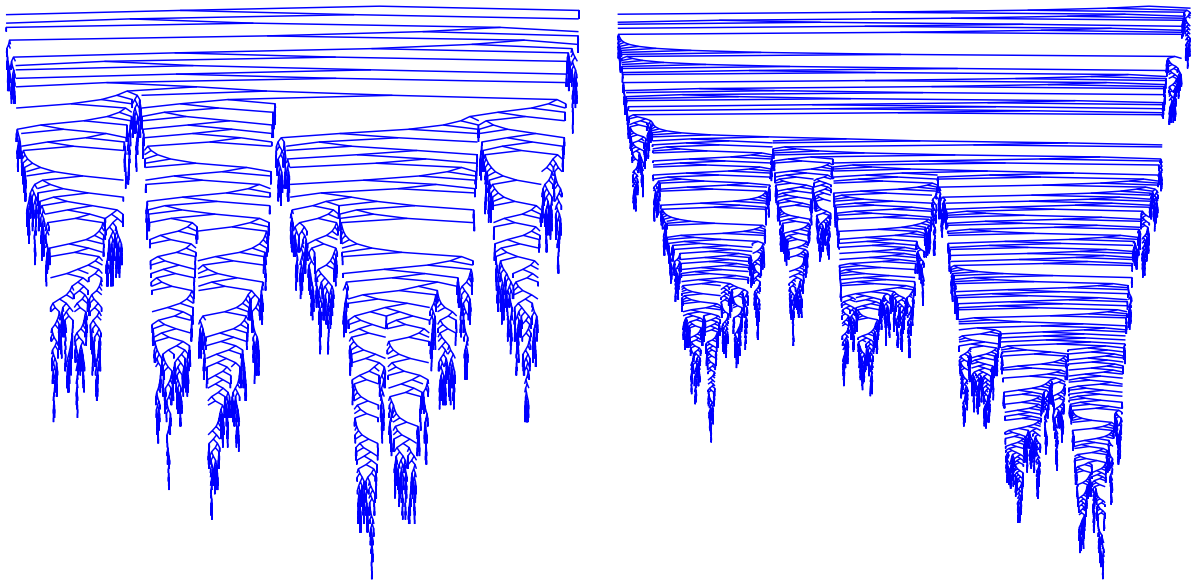


FIG. 1 – Deux arbres binaires non planaires ayant respectivement 3995 et 8554 nœuds.

Démonstration. Soit une paire non ordonnée $\gamma = (\alpha_1, \alpha_2)$ appartenant à $\mathcal{C} = \text{MSET}_2(\mathcal{A})$. Elle est engendrée par $\Gamma\text{MSET}_2[\mathcal{A}](x)$ avec une probabilité $\mathbb{P}_x(\gamma)$:

$$\mathbb{P}_x(\gamma) = \frac{1}{2}\frac{A(x)^2}{C(x)}\ \frac{x^{|\alpha_1|}}{A(x)}\ \frac{x^{|\alpha_2|}}{A(x)} \times 2 = \frac{x^{|\gamma|}}{C(x)},$$

si $\alpha_1 \neq \alpha_2$ et

$$\mathbb{P}_x(\gamma) = \frac{1}{2}\frac{A(x)^2}{C(x)}\ \frac{x^{|\alpha_1|}}{A(x)}\ \frac{x^{|\alpha_1|}}{A(x)} + \frac{1}{2}\frac{A(x^2)}{C(x)}\ \frac{x^{2|\alpha_1|}}{A(x^2)} = \frac{x^{|\gamma|}}{C(x)},$$

sinon ($\alpha_1 = \alpha_2$). □

Exemple 9. Les arbres binaires non planaires enracinés (ou arbres d'Otter[14]) sont spécifiés par l'équation :

$$\mathcal{B} = \mathcal{Z} + \text{MSET}_2(\mathcal{B}).$$

Cette spécification définit la taille d'un arbre comme son nombre de feuilles. À partir de ΓMSET_2, on peut écrire un générateur récursif $\Gamma\mathcal{B}$ pour ces arbres. En choisissant une valeur du paramètre proche de ρ_B, par exemple $x_0 = 0.4026975036$ (voir [FS08, chap. VII.5]), on peut facilement obtenir des arbres de grande taille, comme ceux de la figure 1. Voici les tailles des arbres obtenus par 100 tirages consécutifs de $\Gamma\mathcal{B}(x_0)$:

2, 12, 2, 21, 1, 3, 1, 1, 1, 1, 1, 1, 1, 171, 14, 1, 7, 2, 1, 1, 4, 217, 1, 2, 18, 25, 6, 149, 4, 1, 2, 1, 8, 3, 13, 470, 3, 1, 9079, 11, 2, 1738, 1, 3, 32, 75, 120, 1, 6966, 1, 1, 1, 2, 2, 1, 10, 4, 13, 11, 5, 2, 301, 1, 12, 1, 1, 1, 1, 1, 1, 1, 3, 1, 6, 4, 1, 2, 1, 2, 1, 1, 2, 1, 17, 1, 1, 35, 17, 4, 1, 9, 11, 16, 2, 4, 3, 21, 1, 52, 1.

La forte variabilité des tailles est due à la forme *piquée* de leur distribution (voir chapitre précédent, page 38), qui favorise les petits arbres, mais permet également d'en engendrer de très grands. Ce générateur a permis notamment de fournir des statistiques expérimentales sur la hauteur des arbres d'Otter aléatoires, étudiée dans [BF08].

[14] On les nomme ainsi, suite à leur étude par Otter [Ott48].

1.1.2 Multi-ensemble général

Nous nous intéressons maintenant à la génération de multi-ensembles de structures non étiquetées sans contrainte de cardinalité. Les symétries qui peuvent intervenir dans ces structures nous entraînent à adopter un schéma différent de celui donné pour les ensembles étiquetés. Dans un premier temps, pour engendrer ces multi-ensembles, nous proposons un pseudo-algorithme qui, le plus souvent, débouche sur un processus infini. Bien sûr, cette idée n'aboutit pas à un générateur de Boltzmann viable, mais elle nous fournit les éléments nécessaires à la preuve de validité du second algorithme que nous présentons.

Première version Ce premier algorithme s'appuie sur la définition suivante d'un multi-ensemble $\mathcal{M}$ de $\mathcal{A}$-structures :

$$\mathcal{M} = \textsc{MSet}(\mathcal{A}) \cong \prod_{\gamma \in \mathcal{A}} \textsc{Seq}(\gamma). \tag{3}$$

L'idée est que l'on peut toujours trier les éléments qui composent le multi-ensemble de façon à regrouper les éléments identiques en séquence. Cette définition ne fait intervenir que des produits et des séquences, on peut donc la traduire directement en série génératrice :

$$M(x) = \prod_{\gamma \in \mathcal{A}} \frac{1}{1 - x^{|\gamma|}}. \tag{4}$$

De même, on peut obtenir un générateur de Boltzmann de multi-ensembles de $\mathcal{A}$-structures non étiquetées, en listant les éléments de $\mathcal{A}$ et en tirant une séquence aléatoire de chacun d'entre eux avec le générateur $\Gamma\textsc{Seq}$ vu au chapitre précédent :

```
Γ⋆MSet[A](x)
  M ← ∅
  pour α dans A faire
    k ← Geom(x^|α|)
    ajouter k copies de α à M
  renvoyer M
```

Comme nous l'avons mentionné plus haut, lorsque la classe $\mathcal{A}$ ne contient pas un nombre fini d'éléments, cet algorithme ne termine pas. C'est pourquoi nous parlons plutôt de pseudo-algorithme. Néanmoins, d'un point de vue théorique, c'est un générateur de Boltzmann valide.

Proposition 5. $\Gamma^\star\textsc{MSet}[\mathcal{A}](x)$ *est un générateur de Boltzmann pour la classe* $\textsc{MSet}(\mathcal{A})$.

Démonstration. La probabilité d'une séquence aléatoire α^k appartenant à $\textsc{Seq}(\mathcal{A})$ dans le modèle de Boltzmann est :

$$\mathbb{P}_x(\alpha^k) = \frac{x^{k|\alpha|}}{(1 - x^{|\alpha|})^{-1}}.$$

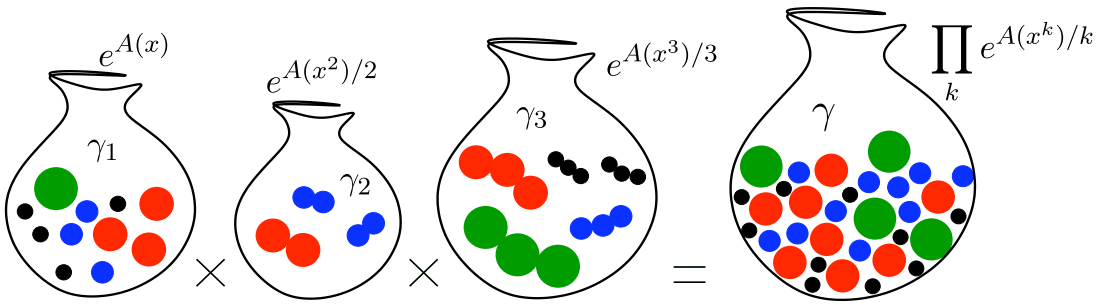
Soit $\gamma = \langle\, \alpha_1^{\odot k_1}, \alpha_2^{\odot k_2}, \ldots, \alpha_\ell^{\odot k_\ell} \,\rangle$ un multi-ensemble aléatoire de $\mathcal{A}$-structures. La probabilité de tirer γ avec le générateur $\Gamma^\star\textsc{MSet}[\mathcal{A}](x)$ est :

$$\mathbb{P}_x(\gamma) = \prod_{i=1}^{\ell} \frac{x^{k_i|\alpha_i|}}{(1 - x^{|\alpha_i|})^{-1}} \prod_{\substack{\beta \in \mathcal{A} \\ \beta \notin \gamma}} \frac{1}{(1 - x^{|\beta|})^{-1}} = \prod_{i=1}^{\ell} x^{k_i|\alpha_i|} \,/\, \prod_{\beta \in \mathcal{A}} (1 - x^{|\beta|})^{-1} = \frac{x^{|\gamma|}}{M(x)}. \qquad \square$$

Deuxième version Le générateur produit à partir de la série génératrice de $\mathcal{M}$ sous la forme donnée par l'équation (4) conduisant à un pseudo-algorithme infini, nous allons maintenant travailler avec une expression équivalente de la série $M(x)$, obtenue par *transformation exp-log* : $f \equiv \exp(\log f)$, de la première :

$$M(x) = \prod_{k=1}^{\infty} \exp\left(\frac{1}{k} A(x^k)\right). \tag{5}$$

L'idée est de suivre la décomposition du multi-ensemble induite par la série génératrice (5). Pour engendrer un multi-ensemble γ de $\mathcal{A}$-structures, on le construit comme une séquence de sous-ensembles $(\gamma_i)_{i\geq 1}$, eux-mêmes obtenus comme des ensembles de i-diagonales de $\mathcal{A}$-structures.



Pour chaque γ_i, on commence par choisir le nombre p de $\mathcal{A}$-structures à engendrer en suivant une loi de Poisson de paramètre $\frac{1}{i}A(x^i)$. Ensuite on tire les $\mathcal{A}$-structures $\alpha_1, \alpha_2, \ldots, \alpha_p$ par des appels indépendants à $\Gamma A(x^i)$. Enfin, on ajoute i copies de chaque α_j au multi-ensemble γ_i.

Nous allons montrer par la suite que le processus que nous venons de décrire est également un générateur de Boltzmann pour un multi-ensemble de $\mathcal{A}$-structures. Néanmoins, celui-ci est tout aussi infini que le précédent. En revanche, on connaît la distribution de l'indice maximal K du produit dans le modèle de Boltzmann :

$$\mathbb{P}_x(K = k) = \frac{\prod_{i=1}^{k} e^{A(x^i)/i}}{M(x)}. \tag{6}$$

Ainsi, en tirant aléatoirement k, l'indice maximal du produit, suivant cette distribution, on peut transformer le produit infini en une boucle finie. Étant donné que l'on tire k de façon à ce qu'il soit précisément le dernier indice pour lequel le sous-ensemble γ_k est non vide, on prend soin de s'assurer que γ_k contient effectivement au moins un élément, en utilisant une loi de Poisson non nulle. On obtient l'algorithme suivant pour engendrer aléatoirement des multi-ensembles :

```
ΓMSet[A](x)
  M ← ∅ ; k ← Indice_Max_MSet(A, x)
  pour i de 1 à k − 1 faire
    p_i ← Poiss(1/i A(x^i))
    répéter p_i fois
      α ← ΓA(x^i)
      ajouter i copies de α à M
  si k ≠ 0 alors
    p_k ←Poiss_≥1(1/k A(x^k))
    répéter p_k fois
      α ← ΓA(x^k)
      ajouter k copies de α à M
  renvoyer M
```

Proposition 6. $\Gamma\text{MSET}[\mathcal{A}](x)$ *est un générateur de Boltzmann pour la classe* $\text{MSET}(\mathcal{A})$.

Pour prouver la validité de l'algorithme $\Gamma\text{MSET}[\mathcal{A}](x)$, la principale difficulté réside dans le fait qu'un même multi-ensemble peut être engendré de plusieurs façons différentes. Contrairement au premier algorithme proposé, ces répétitions ne sont pas uniquement dues au tirage séquentiel utilisé pour mimer le tirage d'un ensemble. Par exemple, le multi-ensemble $\langle\, a, a, a, b \,\rangle$ peut être obtenu de sept façons différentes :

- si l'indice maximal k vaut 1, comme l'une des quatre séquences (a, a, a, b), (a, a, b, a), (a, b, a, a) ou (b, a, a, a) ;
- si $k = 2$, comme l'une des deux séquences (a, b) ou (b, a), associée à la paire $\langle\, a, a \,\rangle$;
- ou enfin, si $k = 3$, comme une occurrence de b, complétée par le triplet $\langle\, a, a, a \,\rangle$.

Plutôt que de compter les différentes façons d'engendrer un même multi-ensemble, la preuve que nous proposons consiste à montrer que les algorithmes $\Gamma^\star\text{MSET}[\mathcal{A}](x)$ et $\Gamma\text{MSET}[\mathcal{A}](x)$ sont équivalents. Plus précisément, on montre comment transformer le premier générateur pour obtenir $\Gamma\text{MSET}[\mathcal{A}](x)$.

Dans un premier temps, le lemme classique suivant nous permet de décomposer la loi géométrique utilisée dans $\Gamma^\star\text{MSET}[\mathcal{A}](x)$ en une somme infinie de lois de Poisson :

Lemme 2. *Soit λ un réel positif et $(Y_i)_{i\geq 1}$ une suite de variables aléatoires indépendantes telles que pour tout $i \geq 1$, Y_i suit une loi de Poisson de paramètre $\frac{\lambda^i}{i}$, avec $\lambda < 1$. Alors la variable aléatoire $= \sum_{i\geq 1} iY_i$ suit une loi géométrique de paramètre λ.*

On peut donc transformer l'algorithme $\Gamma^\star\text{MSET}[\mathcal{A}](x)$ (page 49) en remplaçant le tirage géométrique par une boucle et les tirages de Poisson correspondants. L'algorithme de droite est obtenu en inversant l'ordre des boucles sur i et α dans celui de gauche.

```
M ← ∅                                  M ← ∅
pour α dans A faire                    pour i ≥ 1 faire
  pour i ≥ 1 faire                       pour α dans A faire   } P_i
    k ← Poiss(x^{i|α|}/i)      ≡           k ← Poiss(x^{i|α|}/i)
    ajouter α^{⊙i·k} à M                   ajouter α^{⊙i·k} à M
renvoyer M                             renvoyer M
```

On appelle P_i le processus qui correspond à la i-ième boucle sur les structures de $\mathcal{A}$ dans l'algorithme de droite. Le lemme suivant nous permet de transformer ce processus P_i en une boucle finie d'appels indépendants au générateur $\Gamma\mathcal{A}(x)$.

Lemme 3. *Le processus P_i défini ci-dessus est réalisé par l'algorithme Q_i :*

```
p ← Poiss(1/i A(x^i))
répéter p fois
  α ← ΓA(x^i)
  ajouter α^{⊙i} à M
```

Démonstration. Soit $\gamma_i = \langle\, \alpha_1^{\odot i\cdot k_1}, \alpha_2^{\odot i\cdot k_2}, \ldots, \alpha_\ell^{\odot i k_\ell} \,\rangle$ un multi-ensemble aléatoire de i-uplets de $\mathcal{A}$-structures. On rappelle que la loi de Poisson de paramètre λ est définie par :

$$\mathbb{P}(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}.$$

La probabilité de tirer γ_i par le processus P_i est :

$$\mathbb{P}_{P_i,x}(\gamma_i) = \prod_{j=1}^{\ell} \frac{\left(x^{i|\alpha_j|}/i\right)^{k_j}}{k_j!} \prod_{\beta\in\mathcal{A}} e^{-\frac{x^{i|\beta|}}{i}} = \frac{x^n}{i^s}\, e^{-\frac{A(x^i)}{i}} \prod_{j=1}^{\ell} \frac{1}{k_j!}. \tag{7}$$

où $s = \sum_{j=1}^{\ell} k_j$ et $n = i\sum_{j=1}^{\ell} k_j|\alpha_j|$ sont respectivement le nombre d'éléments et la taille de γ_i.

Le même multi-ensemble γ_i est engendré par le processus Q_i comme une séquence $(\,\beta_1, \beta_2, \ldots, \beta_s\,)$ telle que $|\beta_1| + |\beta_1| + \cdots + |\beta_s| = n$ avec une probabilité :

$$e^{-\frac{A(x^i)}{i}} \frac{\left(A(x^i)/i\right)^s}{s!} \cdot \frac{x^{|\beta_1|}}{A(x^i)} \cdots \frac{x^{|\beta_s|}}{A(x^i)}.$$

Le nombre de séquences différentes qui correspondent à γ_i est $\binom{s}{k_1,\ldots,k_\ell} = \frac{s!}{k_1!\cdots k_\ell!}$, la probabilité d'engendrer γ_i par Q_i est donc :

$$\mathbb{P}_{Q_i,x}(\gamma_i) = \frac{x^n}{i^s}\, e^{-\frac{A(x^i)}{i}} \prod_{j=1}^{\ell} \frac{1}{k_j!}. \tag{8}$$

D'après les équations (7) et (8), les deux processus coïncident. □

L'algorithme final est obtenu en tronquant le produit et en calculant séparément l'indice maximal comme indiqué plus haut. Ceci conclut la preuve de la proposition 6.

Tirage de l'indice maximal Afin de conclure cette section, nous donnons ici quelques indications sur la manière de tirer l'indice maximal de l'algorithme ΓMSet$[\mathcal{A}](x)$. L'équation (6) nous donne la distribution de probabilités de cet indice, il suffit donc de choisir un réel aléatoire U dans l'intervalle $[0, 1]$ et de déterminer k tel que :

$$\frac{\prod_{i=1}^{k-1} e^{A(x^i)/i}}{M(x)} < U \leq \frac{\prod_{i=1}^{k} e^{A(x^i)/i}}{M(x)}.$$

Afin de pouvoir utiliser un algorithme séquentiel similaire au générateur GenLoi(x) présenté au chapitre précédent (page 36), on transforme les produits en sommes en passant au logarithme :

$$\sum_{i=k+1}^{\infty} \frac{1}{i} A(x^i) < \log\frac{1}{U} \leq \sum_{i=k}^{\infty} \frac{1}{i} A(x^i).$$

On obtient alors l'algorithme suivant :

```
Indice_Max_MSet(A, x)
  U ← random(); V ← log 1/U
  p ← log M(x); k ← 0
  tant que U < S faire
    k ← k + 1
    p ← p − A(x^k)/k
  renvoyer k
```

Cette fonction permet de compléter le générateur ΓMSet$[\mathcal{A}](x)$. Nous pouvons désormais engendrer des structures non étiquetées dont la spécification implique un multi-ensemble.

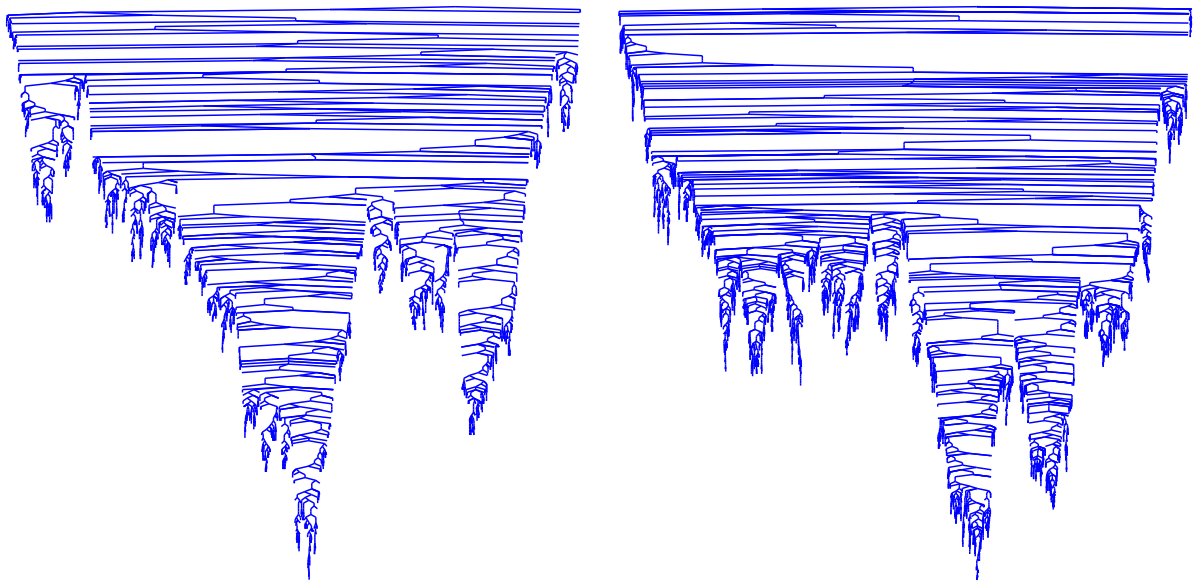


FIG. 2 – Deux arbres généraux non planaires ayant respectivement 4165 et 8271 nœuds.

Exemple 10. Les arbres généraux non planaires enracinés sont spécifiés par l'équation :

$$\mathcal{T} = \mathcal{Z} \times \text{MSET}(\mathcal{T})$$

La construction de multi-ensemble traduit ici l'absence d'ordre entre les différents fils d'un nœud. La taille d'un arbre est le nombre total de nœuds (internes ou externes). À partir de ΓMSET, on peut écrire un générateur récursif $\Gamma\mathcal{T}$ pour $\mathcal{T}$. En choisissant une valeur du paramètre proche de ρ_T, par exemple $x_0 = 0.3383224619$ (voir [FS08, chap. VII.5]), on peut facilement obtenir des arbres de grande taille, comme ceux de la figure 2. Voici les tailles des arbres obtenus lors de 100 tirages consécutifs :

47, 1, 4, 2, 4, 1, 15, 34, 1, 1, 46, 4, 1, 16, 1, 1, 15, 75, 2421, 1517, 690, 3, 1, 5, 2, 1, 1, 2, 7, 1, 1, 1, 1, 13, 361, 7, 8, 41, 1700, 1, 1, 3, 3, 6, 1, 2, 3, 2, 2, 25, 1, 2, 16, 9, 1, 2, 34, 3, 1, 93, 1345, 1, 75, 1, 1, 3, 1, 1, 45, 1, 1, 14, 1, 992, 1, ⧫, 1, 1, 13, 1, 1, 2, 18, 1, 2, 1, 3, 2, 4, 662, 2, 4, 1, 2, 3, 53, 37, 449, 1, 18.

Le générateur utilisé est plafonné (voir chapitre précédent, page 39). Le signe ⧫ indique que le générateur a dépassé la borne choisie (ici 10^4) ; l'arbre en cours a été rejeté.

1.2 Ensemble sans répétitions

Un ensemble sans répétitions[15] est un multi-ensemble tel que les multiplicités de ses éléments sont toutes égales à 1. En d'autres termes, tous les éléments qui composent un ensemble doivent être distincts. Cela correspond à la notion classique d'ensemble. La série génératrice $P(x)$ de la classe $\mathcal{P} = \text{PSET}(\mathcal{A})$ est :

$$P(x) = \prod_{\gamma \in \mathcal{A}} (1 + z^{|\gamma|}) = \prod_{n \geq 1} (1 + x^n)^{a_n} = \exp\left(\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} A(z^k) \right). \tag{9}$$

[15] Par la suite, les ensembles sans répétitions seront simplement appelés des ensembles.

Engendrer des ensembles de $\mathcal{A}$-structures telles qu'elles soient toutes différentes est un problème plus difficile que dans le cas des multi-ensembles. En effet, cela implique une certaine dépendance entre les appels au générateur $\Gamma\mathcal{A}$ que l'on ne sait, a priori, pas traiter. L'idée est alors de suivre la décomposition d'un ensemble induite, non pas par la série génératrice (9), mais par la relation suivante (identité de Vallée [FS08, chap. I.2]) :

$$\mathrm{MSET}(\mathcal{A}) \cong \mathrm{PSET}(\mathcal{A}) \times \mathrm{MSET}(\Delta_2\mathcal{A}). \tag{10}$$

Cette relation s'interprète aisément en remarquant que les éléments qui composent un multi-ensemble peuvent être regroupés par paires d'éléments identiques, avec éventuellement des éléments isolés, selon la parité de leur multiplicité dans le multi-ensemble d'origine. Les paires appartiennent à $\mathrm{MSET}(\Delta_2\mathcal{A})$ et les éléments isolés à $\mathrm{PSET}(\mathcal{A})$. Cette relation se traduit immédiatement en séries génératrices :

$$\prod_{\gamma} \frac{1}{1-z^{|\gamma|}} = \prod_{\gamma} \frac{1+z^{|\gamma|}}{1-z^{2|\gamma|}} = \prod_{\gamma} \left(1+z^{|\gamma|}\right) \prod_{\gamma} \frac{1}{1-z^{2|\gamma|}}.$$

Cette décomposition suggère un algorithme simple pour engendrer des ensembles de $\mathcal{A}$-structures en utilisant le générateur de multi-ensemble $\Gamma\mathrm{MSET}[\mathcal{A}](x)$:

$\Gamma\mathrm{PSET}[\mathcal{A}](x)$
  $M \leftarrow \Gamma$`MSET`$[\mathcal{A}](x)$
  $P \leftarrow \varnothing$
  pour α **dans** M **faire**
    si la mulitplicité **de** α dans M est impaire **alors**
      ajouter α à P
  renvoyer P.

Par rapport aux générateurs de Boltzmann développés jusqu'à présent, cet algorithme occasionne un coût additionnel dû aux éléments engendrés mais qui n'appartiendront pas à l'ensemble final. Nous verrons que le *surcoût*, c'est à dire la somme des tailles des éléments rejetés, est le plus souvent constant. Néanmoins, on peut optimiser le générateur en modifiant l'algorithme $\Gamma\mathrm{MSET}[\mathcal{A}](x)$ pour qu'il n'engendre pas les éléments de $\Delta_i\mathcal{A}$ pour les indices pairs de la boucle principale et qu'il ne garde qu'une seule copie des $\mathcal{A}$-structures engendrées pour les indices impairs. Mais cela ne suffit pas à éliminer toutes les répétitions. Pour compléter le processus d'extraction, on pourra, par exemple, stocker les $\mathcal{A}$-structures dans une table de hachage, afin de vérifier en temps constant si une structure appartient déjà à l'ensemble engendré.

Proposition 7. $\Gamma\mathrm{PSET}[\mathcal{A}](x)$ *est un générateur de Boltzmann pour la classe* $\mathrm{PSET}(\mathcal{A})$. *Si la fonction génératrice* $A(z)$ *de* $\mathcal{A}$ *a un rayon de convergence* ρ *qui satisfait la condition*[16] $\rho < 1$, *alors, pour tout* $x \in]0, \rho[$, *l'espérance du surcoût est majorée par la constante :*

$$K = \frac{2\rho^2 A'(\rho^2)}{1-\rho^2}.$$

Démonstration. La validité est donnée par l'application du lemme de *division* à la décomposition des multi-ensembles induite par la relation (10).

Lemme 4 (Division)**.** *Soient* $\mathcal{H}, \mathcal{K}, \mathcal{L}$ *des classes combinatoires telles que* $\mathcal{H} \cong \mathcal{K} \times \mathcal{L}$. *Étant donné un générateur de Boltzmann* $\Gamma\mathcal{H}(x)$ *pour* $\mathcal{H}$, *la procédure consistant à extraire la première composante d'une structure générée par* $\Gamma\mathcal{H}(x)$ *est un générateur de Boltzmann* $\Gamma\mathcal{K}(x)$ *pour* $\mathcal{K}$.

[16] Cette condition est satisfaite par la plupart des spécifications et est vérifiable de façon algorithmique [FS08].

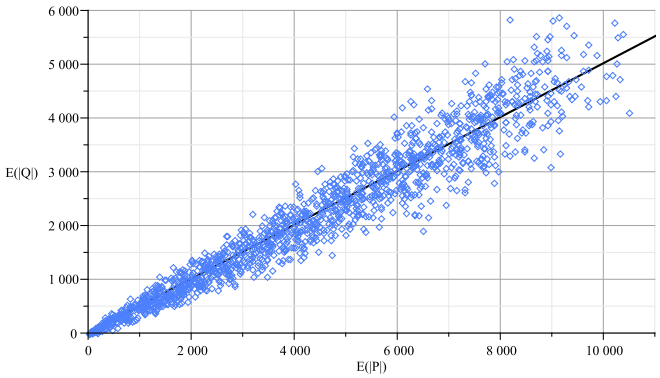


FIG. 3 – Taille des partitions en sommants distincts en fonction de la taille des partitions d'entier correspondantes. Les points bleus correspondent aux valeurs obtenues en utilisant $\Gamma\mathcal{Q}(x)$ pour différentes valeurs de x et la courbe noire correspond aux valeurs obtenues en calculant le rapport des espérances de tailles.

Pour analyser la complexité du générateur, on utilise la série génératrice de probabilité du surcoût induit par le générateur $\Gamma\textsc{PSet}[\mathcal{A}](x)$:

$$\frac{M(x,u)}{M(x)} = \prod_n \left(\frac{1+x^n+u^{2n}(x^{2n}+x^{3n})+\cdots}{1+x^n+x^{2n}+x^{3n}+\cdots}\right)^{a_n} = \prod_n \left(\frac{1-x^{2n}}{1-u^{2n}x^{2n}}\right)^{a_n},$$

où $M(x,u)$ est la série bivariée correspondant aux multi-ensembles de $\mathcal{A}$-structures telle que la variable u marque les éléments rejetés pour extraire les ensembles sans répétitions correspondants. Par dérivation, suivie d'une spécialisation $u=1$, on obtient l'espérance du surcoût :

$$\mathbb{E}_x(\text{surcoût}) = \sum_{n=1}^{\infty} \frac{2nA_nx^{2n}}{1-x^{2n}}.$$

Cette expression est elle-même majorée par K, pour $\rho < 1$. □

Exemple 11. Les partitions d'entiers ($\mathcal{P}$) et les partitions en sommants distincts $\mathcal{Q}$ sont respectivement spécifiées par :

$$\mathcal{P} = \textsc{MSet}(\textsc{Seq}_{\geq 1}(\mathcal{Z})) \qquad \text{et} \qquad \mathcal{Q} = \textsc{PSet}(\textsc{Seq}_{\geq 1}(\mathcal{Z})).$$

On obtient facilement un générateur $\Gamma\mathcal{P}$ pour les partitions d'entiers à partir de $\Gamma\textsc{MSet}$. De plus, en tirant un réel u aléatoire dans $[0,1]$ et en renvoyant $\ln(u)/\ln(x)$, on obtient une loi géométrique de paramètre x avec une complexité arithmétique en $O(1)$. Le coût d'un tirage d'une partition de taille n est donc linéaire en son nombre de sommants, c'est à dire $O(\sqrt{n})$ en moyenne. Enfin, la distribution des tailles des partitions étant concentrée (voir chapitre précédent, page 38), le générateur $\Gamma\mathcal{P}$ permet de tirer des partitions dont la taille peut aller jusqu'à 10^{10} en quelques minutes.

Le générateur $\Gamma\mathcal{Q}$ pour les partitions en sommants distincts, quant à lui, est obtenu par extraction à partir du précédent. Nous sommes dans le cas où la singularité ρ de $Q(x)$ vaut 1. La deuxième partie de la proposition 7 ne s'applique pas. Le surcoût est proportionnel à la taille de la partition. Néanmoins, empiriquement, on observe que le rapport entre la taille d'une partition en sommants distincts et celle de la partition correspondante est d'environ 1/2 (voir figure 3), ce qui altère peu l'efficacité du générateur.

1.3 Cycle

Comme pour les multi-ensembles, l'algorithme que nous proposons pour engendrer des cycles de $\mathcal{A}$-structures appartenant à la classe $\mathcal{C} = \textsc{Cyc}(\mathcal{A})$, est basé sur la décomposition induite par la série génératrice de $\mathcal{C}$:

$$C(x) = \sum_{k\geq 1} \frac{\varphi(k)}{k} \log \frac{1}{1 - A(x^k)}. \tag{11}$$

L'idée est d'engendrer un cycle par répétition d'un motif. Dans un premier temps, on choisit l'ordre de réplication k du cycle (le nombre de répétitions du motif), qui correspond à l'indice de la somme dans $C(x)$, suivant la distribution appropriée :

$$\mathbb{P}(K = k) = \frac{\varphi(k)}{kC(x)} \log \frac{1}{1 - A(x^k)}. \tag{12}$$

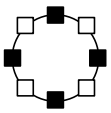
Ce tirage peut être fait grâce à un algorithme similaire à celui qui permet de tirer l'indice maximal d'un multi-ensemble (voir page 52). La longueur j du motif est déterminée par la loi logarithmique de paramètre $A(x^k)$:

$$\mathbb{P}(J = j) = \frac{A(x^k)^j}{j} \left(\log \frac{1}{(1 - A(x^k))} \right)^{-1}. \tag{13}$$

Enfin, le motif lui-même est engendré par une suite de j tirages indépendants de $\Gamma\mathcal{A}(x^k)$. On obtient alors l'algorithme suivant :

$\Gamma\textsc{Cyc}[\mathcal{A}](x)$
  $k \leftarrow$ `Ordre(`x`)`
  $j \leftarrow$ `Loga`$\big(A(x^k)\big)$
  $w \leftarrow (\underbrace{\Gamma\mathcal{A}(x^k), \ldots, \Gamma\mathcal{A}(x^k)}_{j \text{ fois}})$
  `renvoyer un cycle composé de` k `copies de` w

On peut observer que ce générateur n'engendre pas chaque cycle de façon unique. Par exemple le cycle suivant :



peut être engendré indifféremment comme les séquences $(\square\blacksquare)^4$, $(\blacksquare\square)^4$, $(\square\blacksquare\square\blacksquare)^2$, $(\blacksquare\square\blacksquare\square)^2$, $(\square\blacksquare\square\blacksquare\square\blacksquare\square\blacksquare)$ ou $(\blacksquare\square\blacksquare\square\blacksquare\square\blacksquare\square)$. Néanmoins, les distributions utilisées pour choisir l'ordre de réplication et la longueur du motif permettent d'obtenir un générateur uniforme.

Proposition 8. $\Gamma\textsc{Cyc}[\mathcal{A}](x)$ *est un générateur de Boltzmann pour la classe* $\textsc{Cyc}(\mathcal{A})$.

Démonstration. D'après les équation (12) et (13), la probabilité que l'ordre de réplication soit k et la longueur du motif j est :

$$\mathbb{P}(K = k, J = j) = \frac{\varphi(k)}{kj} \frac{A(x^k)^j}{C(x)}.$$

Soit γ un cycle de longueur ℓ et de taille n appartenant à la classe $\textsc{Cyc}(\mathcal{A})$. On peut le décomposer en r répétitions d'un motif *primitif*, c'est à dire qui n'a aucune symétrie par

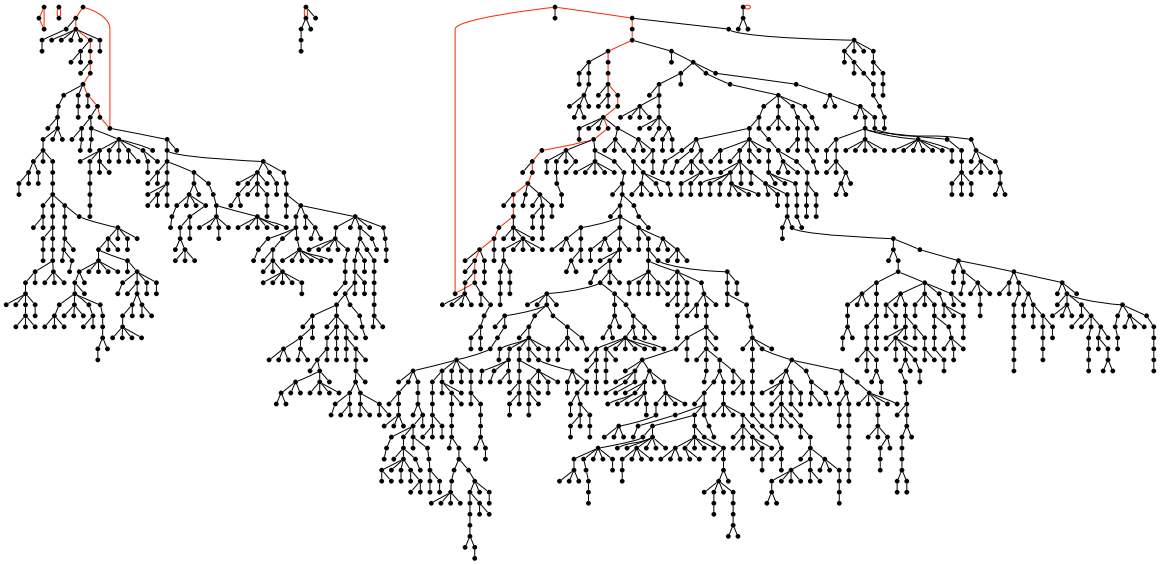


FIG. 4 – Graphe fonctionnel, 1504 nœuds.

décalages cycliques : $\gamma = \mu^r$, tel que $\mu = (\alpha_1, \dots \alpha_s)$ est primitif. Comme nous l'avons déjà fait remarquer, le générateur $\Gamma\textsc{Cyc}[\mathcal{A}](x)$ peut produire γ de différentes façons. Plus précisément, il peut être engendré comme k répétitions d'un motif $\omega = \omega_1 \dots \omega_j$, c'est-à-dire $\gamma = \omega^k$, pour toute valeur de k divisant r. Cela implique notamment que $kj = l$ et $w = \tilde{\mu}^{r/k}$, où $\tilde{\mu}$ représente n'importe quel décalage cyclique de μ. La probabilité que γ soit engendré par $\Gamma\textsc{Cyc}[\mathcal{A}](x)$ est alors :

$$\mathbb{P}_x(\gamma) = s \sum_{kj=l, k|r} \frac{\varphi(k)}{kj} \frac{A(x^k)^j}{C(x)} \frac{x^{k|\omega_1|}}{A(x^k)} \dots \frac{x^{k|\omega_j|}}{A(x^k)}. \tag{14}$$

Le facteur s vient du fait que, par décalages cycliques, γ peut-être obtenu à partir de s séquences différentes. En utilisant la propriété : $\sum_{k|m} \varphi(k) = m$, on peut simplifier :

$$\mathbb{P}(\gamma) = \frac{s}{l} \frac{x^{|\gamma|}}{C(x)} \sum_{k|r} \varphi(k) = \frac{x^{|\gamma|}}{C(x)}. \qquad \square$$

Exemple 12. La classe des graphes fonctionnels $\mathcal{F}$ est décrite par la spécification :

$$\mathcal{F} = \textsc{MSet}(\textsc{Cyc}(\mathcal{T})), \qquad \mathcal{T} = \mathcal{Z} \times \textsc{MSet}(\mathcal{T}).$$

On peut, à partir de $\Gamma\textsc{MSet}$, $\Gamma\textsc{Cyc}$ et du générateur d'arbres $\Gamma\mathcal{T}$ (voir exemple 10), assembler un générateur $\Gamma\mathcal{F}$ pour engendrer des graphes fonctionnels. En choisissant une valeur de paramètre assez proche de la singularité $\rho_F = \rho_T$ de leur série génératrice (par exemple, $x_0 = 0.33802$), voici les tailles des premiers graphes engendrés :

$159, 407, 0, 89, 489, 291, 7, 3711, 103, 15, 1015, 0, 371, 1051, 1904, 73, 66, 25, 2014, 6, 820, \dots$

La figure 4 donne un exemple de graphe fonctionnel obtenu avec $\Gamma\mathcal{F}(x_0)$. Les cycles sont tracés en rouge[17].

[17] Dessin réalisé avec Graphviz.

1.4 Contraintes de cardinalité

Afin de compléter la collection des générateurs de Boltzmann pour les constructions du théorème 3, nous présentons, dans cette section, les outils permettant d'adapter les algorithmes précédents à des contraintes de cardinalité. On pourra ainsi engendrer des ensembles et des cycles ayant un nombre fixé de composants.

Multi-ensemble à k éléments La série génératrice $M_k(x)$ d'un multi-ensemble composé de k structures de la classe $\mathcal{A}$ est donnée par l'extraction du coefficient de u^k dans la série bivariée de ces multi-ensembles où la variable u marque le nombre de composants :

$$M_k(x) = [u^k] \exp\Big(\sum_{i\geq 1} \frac{u^i}{i} A(x^i)\Big). \tag{15}$$

Par exemple, pour $k = 2$, cela correspond à la série de l'équation (2) de la section 1.1.1 et pour $k = 3$, on obtient :

$$M_3(x) = \frac{1}{6}A(x)^3 + \frac{1}{2}A(x)A(x^2) + \frac{1}{3}A(x^3).$$

Plus généralement, $M_k(x)$ est un polynôme en $A(x), A(x^2), \ldots, A(x^k)$. Cela signifie, entre autre, que la série génératrice nous indique une décomposition des multi-ensembles à k éléments en termes d'unions, de produits et de diagonales. Or, pour toutes ces constructions, nous disposons déjà des générateurs de Boltzmann correspondants. Il suffit alors de les assembler pour obtenir un générateur pour $\textsc{MSet}_k(\mathcal{A})$. Par exemple, pour engendrer un élément de $\textsc{MSet}_3(\mathcal{A})$, il suffit de choisir, par un tirage de Bernoulli, l'un des trois types $\langle \alpha, \alpha', \alpha'' \rangle$, $\langle \alpha, \alpha', \alpha' \rangle$ ou $\langle \alpha, \alpha, \alpha \rangle$ et de faire ensuite appel au générateur $\Gamma\mathcal{A}$. Afin de décrire le schéma général de l'algorithme, on introduit la notion de *séquence de partition* :

Définition 7. *Une* séquence de partition *de l'entier k est une suite d'entiers (n_i) telle que :*

$$\sum_{i=1}^{k} i\, n_i = k.$$

On note $\mathcal{P}_k$ l'ensemble des séquences de partition de taille k. Pour $k = 3$, par exemple, on obtient $\mathcal{P}_3 = \{(3,0,0), (1,1,0), (0,0,1)\}$. Pour toute séquence de partition $P \in \mathcal{P}_k$, on introduit la série génératrice :

$$M_P(x) := \prod_{i=1}^{k} A(x^i)^{n_i}/(n_i! i^{n_i}).$$

On observe que :

$$M_k(x) = \sum_{P\in\mathcal{P}_k} M_P(x).$$

Ainsi, on peut retrouver l'équation (16) à partir de $\mathcal{P}_3$. En utilisant ces notations, on peut désormais donner l'algorithme générique pour $\textsc{MSet}_k(\mathcal{A})$:

$\Gamma\textsc{MSet}_k[\mathcal{A}](x)$
  $M \leftarrow \varnothing$
  `tirer` P **dans** $\mathcal{P}_k$ `telle que` $\mathbb{P}_{x,k}(P) = M_P(x)/M_k(x)$
  pour i **de** 1 **à** k **faire**
    `ajouter le` n_i`-uplet` $\langle \Gamma\Delta_i\mathcal{A}(x) \ldots, \Gamma\Delta_i\mathcal{A}(x) \rangle$ `à` M
  `renvoyer` M

Cycle à k éléments Pour engendrer des cycles composés de k éléments, on s'inspire également de leur série génératrice :

$$C_k(x) = [u^k] \sum_{i\geq 1} \frac{\varphi(i)}{i} \log \frac{1}{1-u^i A(x^i)} \tag{16}$$

$$= \frac{1}{k} \sum_{i|k} \varphi(i) A(x^i)^{\frac{k}{i}}, \tag{17}$$

qui se traduit immédiatement en générateur :

$\Gamma\textsc{Cyc}_k[\mathcal{A}](x)$

> `tirer un diviseur` i `de` k `tel que` $\mathbb{P}_{x,k}(i) = \varphi(i)A(x^i)^{k/i}/(kC_k(x))$
>
> $w \leftarrow \langle \underbrace{\Gamma\mathcal{A}(x^i) \ldots, \Gamma\mathcal{A}(x^i)}_{k/i \text{ fois}} \rangle$
>
> `renvoyer un cycle composé de` i `copies de` w

On peut observer que les deux algorithmes précédents miment le comportement de $\Gamma\textsc{MSet}[\mathcal{A}](x)$ et $\Gamma\textsc{Cyc}[\mathcal{A}](x)$ conditionnés pour engendrer des ensembles et des cycles composés de k structures de la classe $\mathcal{A}$. On en déduit la proposition suivante qui vient conclure la preuve du théorème 3.

Proposition 9. *$\Gamma\textsc{MSet}_k[\mathcal{A}](x)$ et $\Gamma\textsc{Cyc}_k[\mathcal{A}](x)$ sont des générateurs de Boltzmann pour les classes $\textsc{MSet}_k(\mathcal{A})$ et $\textsc{Cyc}_k(\mathcal{A})$.*

Autres contraintes Pour compléter la liste des constructions admissibles, on remarque que l'on peut également réaliser des générateurs pour des ensembles ou des cycles composés d'*au plus* k éléments ($\textsc{MSet}_{\leq k}$ et $\textsc{Cyc}_{\leq k}$) en les décomposant comme des unions disjointes de ces constructions ayant $1, 2, \ldots, k$ éléments :

$$\textsc{MSet}_{\leq k}(\mathcal{A}) = \bigcup_{i=1}^{k} \textsc{MSet}_i(\mathcal{A}) \quad \text{et} \quad \textsc{Cyc}_{\leq k}(\mathcal{A}) = \bigcup_{i=1}^{k} \textsc{Cyc}_i(\mathcal{A})$$

Enfin, les constructions $\textsc{MSet}_{\geq k}$ et $\textsc{Cyc}_{\geq k}$ composées d'*au moins* k éléments, ainsi que les ensembles sans répétions $\textsc{PSet}_k$, $\textsc{PSet}_{\leq k}$ et $\textsc{PSet}_{\geq k}$ peuvent être obtenus à partir des générateurs $\Gamma\textsc{MSet}$, $\Gamma\textsc{PSet}$ et $\Gamma\textsc{Cyc}$ en utilisant une procédure de rejet sur le nombre de composants.

Exemple 13. Les graphes série-parallèle $\mathcal{SP}$ représentent des circuits électriques tels que les circuits en parallèle sont montés en série et les circuits en série sont montés en parallèle. Ils sont définis par la spécification suivante :

$$\mathcal{SP} = \mathcal{Z} + \mathcal{S} + \mathcal{P}, \quad \mathcal{S} = \textsc{Seq}_{\geq 2}(\mathcal{Z} + \mathcal{P}), \quad \mathcal{P} = \textsc{MSet}_{\geq 2}(\mathcal{Z} + \mathcal{S})$$

On utilise un multi-ensemble pour décrire un circuit monté en parallèle afin de traduire l'absence d'ordre entre les éléments et une séquence pour les circuits montés en série. Naturellement, ces montages se font sur au minimum deux éléments, d'où les contraintes de cardinalités. On peut aisément assembler un générateur pour ces circuits à partir des algorithmes donnés précédemment. Dans la deuxième partie de ce mémoire, nous reprendrons cet exemple des graphes série-parallèle pour illustrer notre méthode de calcul de l'oracle. Le programme que nous avons implanté a permis, par exemple, d'engendrer le circuit de la figure 5.

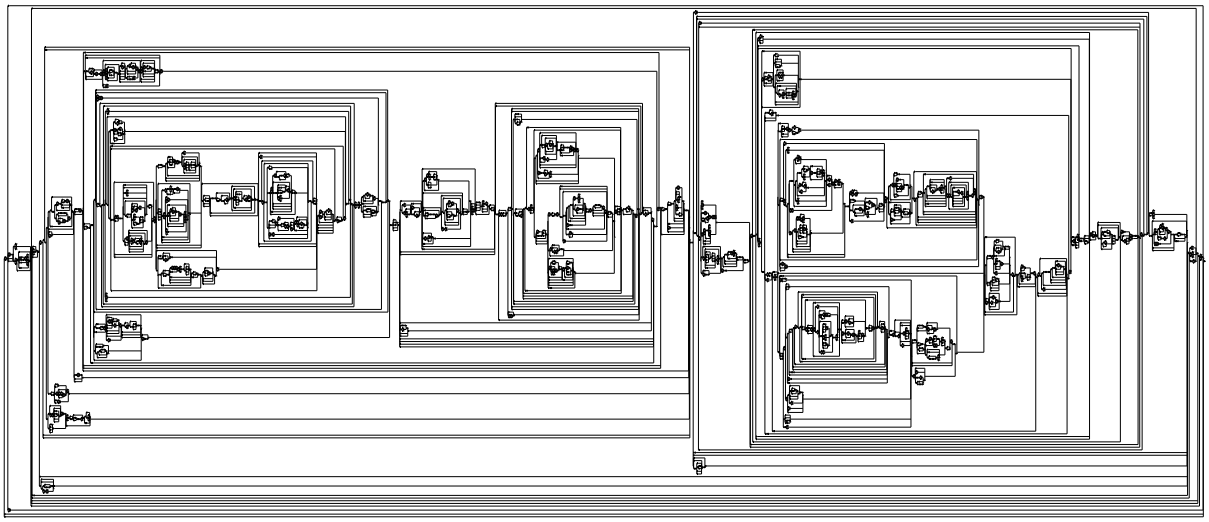


FIG. 5 – Circuit série-parallèle, 3089 nœuds.

2 Implantations

2.1 Générateurs effectifs

Nous avons développé des générateurs de Boltzmann pour tous les exemples traités dans ce chapitre. Nous présentons ici quelques résultats expérimentaux obtenus avec ces générateurs. Nous avons principalement effectué deux types de simulations. Les premières visent à observer de façon empirique la complexité des générateurs en taille approchée. Les secondes donnent un aperçu des distributions de tailles de structures que l'on peut obtenir avec des générateurs de Boltzmann. Ces mesures ont été effectuées avec un processeur Intel à 3.2GHz et 2Go de mémoire.

Complexité empirique Les tables 1 et 2 donnent les temps moyens de génération pour différentes classes combinatoires, en fonction des tailles de structures souhaitées. On tolère une variation de $\pm 10\%$ autour de la taille visée. Chaque mesure est obtenue en faisant une moyenne sur environ 1000 tirages. Les deux générateurs de partitions (partitions d'entiers et partitions en sommants distincts) sont écrits en Maple 10. L'efficacité de ces générateurs vient du fait que, d'une part, les distributions de tailles sont concentrées, ainsi on rejette très peu de partitions, et d'autre part, la génération elle-même a une complexité sous-linéaire, parce que les sommants des partitions sont obtenus par des tirages de lois géométriques, qui ont un coût constant. Il est alors possible d'engendrer des partitions de taille 10^{10} en à peine plus d'une minute.

Les autres générateurs sont codés en OCaml. Les compositions circulaires sont décrites par la spécification $\mathcal{C}_c = \textsc{Cyc}(\textsc{Seq}_{\geq 1}(\mathcal{Z}))$. De même que pour les partitions et les graphes fonctionnels, le générateur de compositions cycliques de taille approximativement n est obtenu en choisissant une valeur de paramètre telle que l'espérance de la taille des compositions engendrées est n. Parallèlement, pour les structures arborescentes (les mobiles sont des arbres tels que les fils de chaque nœud sont organisés cycliquement), on utilise des générateurs singuliers plafonnés (voir chapitre précédent, page 39). Pour tous ces exemples, on observe alors la complexité linéaire attendue.

Classe	Partitions d'entier	Partitions (sommants $\neq$)	Compositions circulaires	Graphes fonctionnels
Taille approchée	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(n)$	$O(n)$
Taille exacte	$O(n)$	$O(n)$	$O(n^2)$	$O(n^2)$
Méthode	$\mathbb{E}_x(N) = n$	$\mathbb{E}_x(N) = n$	$\mathbb{E}_x(N) = n$	$\mathbb{E}_x(N) = n$
$100 \pm 10\%$	0.033 sec.	0.039 sec.	0.0047 sec.	0.0084 sec.
$1000 \pm 10\%$	0.066 sec.	0.078 sec.	0.051 sec.	0.080 sec.
$10^4 \pm 10\%$	0.13 sec.	0.17 sec.	0.57 sec.	0.82 sec.
$10^5 \pm 10\%$	0.29 sec.	0.41 sec.	6.23 sec.	8.36 sec.
$10^6 \pm 10\%$	0.78 sec.	1.07 sec.	60.20 sec.	86.22 sec.
$10^7 \pm 10\%$	2.56 sec.	3.43 sec.	574.37 sec.	–
$10^8 \pm 10\%$	8.73 sec.	10.69 sec.	–	–
$10^9 \pm 10\%$	27.19 sec.	41.16. sec.	–	–
$10^{10} \pm 10\%$	88.46 sec.	155.49 sec.	–	–

Tab. 1 – Complexités et temps de génération pour différentes classes combinatoires.

Classe	Arbres d'Otter	Arbres généraux	Mobiles	Circuits
Taille approchée	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Taille exacte	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Méthode	gén. singulier	gén. singulier	gén. singulier	gén. singulier
$100 \pm 10\%$	0.024 sec.	0.018 sec.	0.021 sec.	0.047 sec.
$1000 \pm 10\%$	0.20 sec.	0.17 sec.	0.19 sec.	0.43 sec.
$10^4 \pm 10\%$	2.20 sec.	1.62 sec.	1.79 sec.	4.76 sec.
$10^5 \pm 10\%$	22.81 sec.	18.13 sec.	17.10 sec.	43.60 sec.
$10^6 \pm 10\%$	239.05 sec.	172.28 sec.	164.35 sec.	531.36 sec.

Tab. 2 – Complexités et temps de génération pour différentes classes combinatoires - suite.

Échantillonnage Le deuxième type d'expériences que nous avons menées permet d'observer les distributions des tailles de structures engendrées avec les même générateurs que précédemment. La figure 6 présente un histogramme[18] des tailles des partitions d'entiers et des graphes fonctionnels obtenus en paramétrant le générateur pour obtenir des structures de taille 1000. Chaque rectangle représente le nombre de structures engendrées dans l'intervalle de tailles correspondant en abscisse. On peut observer la concentration de la distribution des tailles pour les partitions, alors que dans le cas des graphes fonctionnels, on a typiquement une distribution dite "plate" (voir chapitre précédent, page 38).

Les tables 3 et 4 donnent la répartition des tailles d'arbres engendrés lors de respectivement 10^5 et 10^6 tirages avec des générateurs singuliers. Pour chaque classe d'arbres, on donne le nombre d'arbres obtenus dans l'intervalle de taille indiqué par la première ligne. La colonne 2

[18]Dessin réalisé avec R.

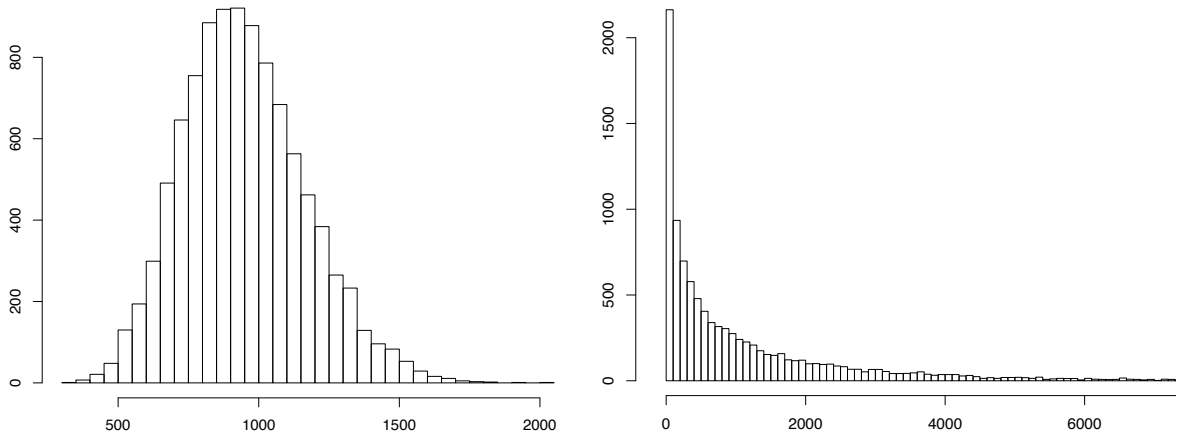


FIG. 6 – Tailles des partitions d'entiers (à gauche) et des graphes fonctionnels (à droite) engendrés par 10^4 tirages de Boltzmann, paramétrés pour obtenir des structures de taille 1000.

correspond aux arbres de taille inférieure à 250, la colonne 3 représente ceux dont la taille est comprise entre 250 et 10^3, et ainsi de suite. La dernière colonne indique le temps total mis pour faire les 10^5 ou 10^6 tirages de chaque simulation. Toutes ces familles d'arbres ont des distributions piquées. On observe donc une concentration des arbres de très petite taille. Néanmoins, on constate également, au vu de ces simulations, qu'il est possible d'obtenir un échantillon raisonnable d'arbres des très grande taille en peu de temps.

taille $\leq$	250	10^3	4.10^3	16.10^3	64.10^3	256.10^3	1024.10^3	+	
Arbres d'Otter	95964	2042	992	512	233	117	69	71	26 min
Arbres généraux	94375	2781	1464	659	371	181	85	77	23 min
Mobiles	96546	1790	826	457	193	94	55	38	14 min
Circuits	94362	2791	1454	691	320	179	103	96	81 min

TAB. 3 – Échantillons de tailles, 100 000 tirages.

2.2 Conclusion

À ce stade, nous disposons d'un ensemble d'algorithmes génériques permettant de dériver des générateurs de Boltzmann pour toute spécification combinatoire non étiquetée (au sens de la définition 1, page 21).

En dehors des exemples présentés dans ce chapitre, la méthode de Boltzmann a déjà été appliquée pour des problèmes concrets, notamment par Fusy pour la génération de cartes planaires [Fus05, Fus07], pour laquelle il introduit des générateurs mixtes etiquetés/non étiquetés et un premier opérateur de substitution. Dans [BFKV07], Bodirsky *et al.* présentent également un opérateur de pointage non biaisé pour les classes combinatoires étiquetées comportant des

taille $\leq$ taille $\leq$	250	10^3	4.10^3	16.10^3	64.10^3	256.10^3	+	
Arbres d'Otter	959612	20394	9966	4959	2515	1241	1313	2h 5min
Arbres généraux	944122	27787	14002	7086	3510	1761	1732	2h 15min
Mobiles	965062	17791	8646	4275	2036	1048	1142	1h 37min
Circuits	945152	27412	13751	6782	3362	1740	1801	5h 54min

TAB. 4 – Échantillons de tailles, 1 000 000 tirages.

opérateurs de Pólya[19]. Ainsi, ils peuvent étendre la génération de Boltzmann à l'opération de substitution dans le cas des structures non étiquetées. Enfin, dans [BJ08], les auteurs considèrent un modèle mixte de structures partiellement étiquetées et définissent les générateurs de Boltzmann correspondants.

Le prochain chapitre présentera une autre application de la génération de Boltzmann non étiquetée, pour engendrer des partitions planes de grande taille.

[19]Contrairement aux structures rigides, si l'on essaie de pointer naïvement une structure comportant des symétries, on ne produit pas n structures pointées de taille n.

Chapitre 3

Application à la génération de partitions planes

Sommaire

Les partitions planes, à l'origine introduites par Young [You01], font l'objet d'un intérêt constant dans de nombreux domaines des mathématiques [BK72, Gan81, Knu70, MK06, Wri31] et de la physique statistique [MN07, Ver96]. Elles ont également joué un rôle crucial dans la résolution de problèmes difficiles en combinatoire [Zei96, Bre99]. La série génératrice qui énumère les partitions planes a une forme particulièrement simple mais ces partitions ne sont pas pour autant spécifiables directement, contrairement aux exemples que nous avons vus jusqu'à présent. Pour produire un générateur aléatoire de partitions planes, nous allons utiliser une bijection due à Pak [Pak02], afin de se ramener à des structures décomposables pour lesquelles ont sait assembler un générateur de Boltzmann. D'autres travaux portent sur la génération aléatoire des partitions planes ; Krattenthaler [Kra99] propose notamment un algorithme bijectif pour engendrer uniformément des partitions planes contenues dans une boîte de dimensions $a \times b \times c$. Cela équivaut à engendrer aléatoirement des pavages d'un hexagone de côtés (a, b, c, a, b, c) par des losanges[20]. La bijection est obtenue en observant la représentation en 3D d'une partition plane (un empilement de cubes) suivant la direction $(-1, -1, -1)$. Notre approche est légèrement différente car nous souhaitons engendrer des partitions uniformément par rapport à leur taille, paramètre relativement naturel puisqu'il correspond au volume de la partition dans sa représentation en 3D. Récemment, plusieurs auteurs ont étudié les propriétés statistiques

[20] Il existe d'autres travaux sur la générations des pavages d'un hexagone par des losanges, notamment ceux de Propp et Wilson[Pro97, Wil97].

des partitions planes de ce point de vue. En particulier, Cerf et Kenyon [CK01] ont déterminé la forme asymptotique des partitions planes à taille fixée. C'est une question qui a également intéressé Okounkov et Reshetikhin [OR03] qui ont étendu cette étude au cas des partitions encadrées ou tordues [OR07]. Nous présenterons dans ce chapitre des générateurs pour ces trois familles de partitions. Nous verrons que l'association de la bijection de Pak et de la méthode de Boltzmann conduit à des algorithmes efficaces permettant d'engendrer, en quelques minutes, des partitions planes dont la taille peut aller jusqu'à 10^7.

Ce chapitre reprend les résultats présentés dans [BFP06, BFP07].

1 Définitions

Les partitions planes (et leurs variantes) sont des structures pour lesquelles il n'existe pas de décomposition naturelle en termes de constructions admissibles. En particulier, elles possèdent une double contrainte d'ordre (horizontal et vertical) qui les rend sensiblement plus complexes qu'il n'y parait. Néanmoins, il existe des classes spécifiables avec lesquelles elles sont en bijection. C'est pourquoi, après avoir défini les classes de partitions planes dont il est question, nous présentons également ces classes spécifiables. L'isomorphisme qui les lie fera l'objet de la section suivante.

1.1 Partitions planes

Définition 8. *Une* partition plane *de taille n est un tableau d'entiers à deux dimensions $(a_{i,j})_{\mathbb{N}_+^2}$. Ses entrées sont décroissantes de bas en haut et de gauche à droite et leur somme vaut n. En d'autres termes,*

$$\forall (i,j) \in \mathbb{N}_+^2, \quad a_{i,j} \geq a_{i,j+1} \geq 0, \quad a_{i,j} \geq a_{i+1,j} \geq 0 \quad \textit{et} \quad \sum_{i,j} a_{i,j} = n. \tag{1}$$

On appelle $\mathcal{P}$ la classe combinatoire formée de l'ensemble des partitions planes. La fonction de taille associée à une partition $P = (a_{i,j})_{\mathbb{N}_+^2}$ est :

$$|P| = \sum_{i,j} a_{i,j}.$$

L'ensemble des entrées non nulles du tableau délimitent la *forme*[21] de la partition plane. Cette forme correspond à une partition d'entier $\lambda = (\ell_1, \ell_2, \ldots, \ell_k)$, où chaque ℓ_i représente le nombre d'éléments non nuls dans la colonne d'indice i de la partition plane et k est le nombre d'éléments non nuls dans la première ligne. Ainsi, une définition alternative pour une partition plane est la donnée d'une forme et d'un remplissage par des entiers non nuls, décroissants en ligne et en colonne. La figure 1 représente, par exemple, une partition plane de taille 22 et de forme $\lambda = (4, 2, 2, 1)$. Le repère en rouge indique l'orientation (le sens de numérotation des lignes et des colonnes) de la partition. Les partitions planes se représentent naturellement en trois dimensions comme des empilements de cubes de hauteur décroissante suivant les axes x et y. La taille de la partition correspond alors au nombre de cubes qui composent l'empilement. La figure 1 représente une partition plane vue suivant la direction $(-1, -1, -1)$.

Nous appelons *rectangle enveloppant* d'une partition $P = (a_{i,j})_{\mathbb{N}_+^2}$, le plus petit intervalle double $R = [1..a] \times [1..b]$ contenant tous les indices (i, j) pour lesquels les entrées de P sont non

[21] Aussi appelée diagramme de Ferrer.

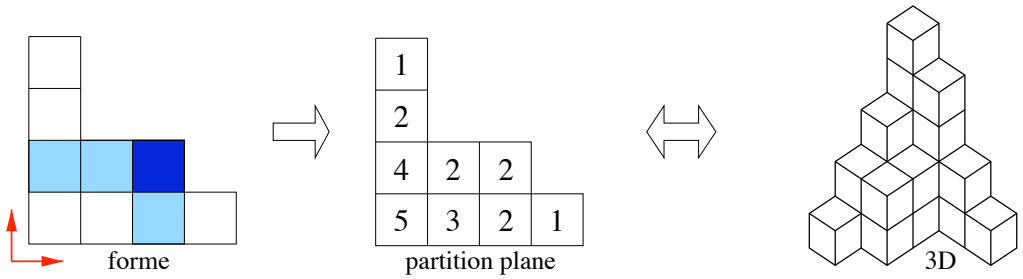


FIG. 1 – Une partition plane et sa représentation en 3 dimensions.

nulles. Le rectangle enveloppant d'une partition de forme $\lambda = (\ell_1, \ell_2, \ldots, \ell_k)$ est (ℓ_1, k). Nous pouvons maintenant définir une variante des partitions planes, les partitions encadrées.

Définition 9. *Étant donnés a et b dans $\mathbb{N}^2$, une partition plane* $(a\times b)$-encadrée *est une partition dont le rectangle enveloppant est contenu dans le rectangle $(a \times b)$. On note $\mathcal{P}_{a,b}$ la classe des partitions planes encadrées par le rectangle $(a \times b)$.*

Par exemple, la classe $\mathcal{P}_{1,1}$ contient toutes les partitions réduites à une seule case ; la partition plane représentée par la figure 1, quant à elle, appartient à la classe $\mathcal{P}_{4,4}$, mais également à toutes les classes $\mathcal{P}_{i,j}$ telles que $i \geq 4$ et $j \geq 4$. Enfin, nous traitons une dernière variété de partitions planes, les partitions tordues.

Définition 10. *Une partition D-*tordue *est un tableau d'entiers positifs $(a_{i,j})_D$ dont les indices appartiennent à un domaine $D \subset [1..a] \times [1..b]$, pour $a \geq 1$ et $b \geq 1$, satisfaisant la condition de monotonicité suivante :*

$$\{(i,j) \in D \Rightarrow (i',j') \in D \text{ si } i' \in [i,a[\text{ et } j \in [j',b[\}.$$

De plus, ses entrées sont décroissantes en ligne et en colonne. On note $\mathcal{S}_D$ la classe des partitions D-tordues. La taille d'une partition tordue est la somme des entiers qui la composent.

Les partitions planes et les partitions encadrées sont des cas particuliers des partitions tordues, pour $D = \mathbb{N}_+^2$ et $D = [1..a] \times [1..b]$. Comme pour les partitions planes, on peut définir alternativement une partition tordue comme un tableau de forme λ/μ, où λ et μ sont des partitions d'entier telles que $\mu \subset \lambda$. La figure 2 donne un exemple de partition D-tordue de taille 43, avec $D = (8,5,5,3,2,1)/(3,2)$.

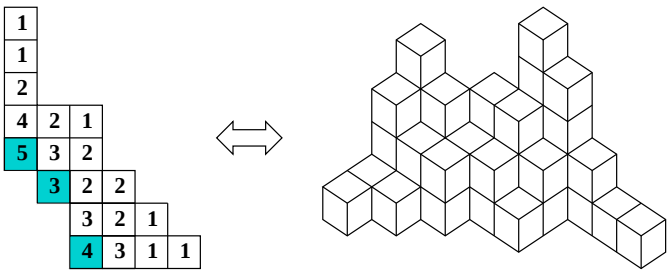


FIG. 2 – Une partition tordue et sa représentation en 3 dimensions.

1.2 Séries génératrices et structures décomposables

Le problème de l'énumération des partitions planes a été résolu par Mac Mahon [Mac96] qui a prouvé la formule remarquable :

$$P(z) = \prod_{r\geq 1}(1-z^r)^{-r}, \tag{2}$$

pour leur série génératrice $P(z)$. Cette formule, d'une étonnante simplicité, est pourtant restée longtemps sans preuve constructive. La première preuve complètement bijective est due à Krattenthaler dans [Kra99]. Cette série génératrice peut se réécrire sous la forme plus familière :

$$P(z) = \prod_{i,j\geq 0}\frac{1}{1-z^{i+j+1}} \tag{3}$$

qui est également la série génératrice d'une classe décomposable non étiquetée $\mathcal{M}$ spécifiée par :

$$\mathcal{M} = \textsc{MSet}(\mathcal{Z}\times\textsc{Seq}(\mathcal{Z})^2). \tag{4}$$

Il existe donc un isomorphisme implicite entre $\mathcal{P}$ et $\mathcal{M}$. Nous verrons dans la section suivante que l'algorithme présenté par Pak dans [Pak02] réalise effectivement cet isomorphisme. Classiquement, $\textsc{Seq}(\mathcal{Z})$ est identifié à la classe des entiers positifs ou nuls, de telle sorte que l'on peut spécifier $\mathcal{M}$ comme un multi-ensemble de couples d'entiers[22], avec la notation simplifiée :

$$\mathcal{M} = \textsc{MSet}(\mathcal{Z}\times\mathbb{N}^2). \tag{5}$$

De façon similaire, nous pouvons définir les classes décomposables isomorphes aux partitions encadrées et tordues :

$$\mathcal{M}_{a,b} = \prod_{\substack{0\leq i<a\\ 0\leq j<b}}\textsc{Seq}(\mathcal{Z}\times\mathcal{Z}^i\times\mathcal{Z}^j) = \textsc{MSet}(\mathbb{N}_{<a}\times\mathbb{N}_{<b}) \simeq \mathcal{P}_{a,b}, \tag{6}$$

$$\mathcal{M}_D = \prod_{(i,j)\in D}\textsc{Seq}(\mathcal{Z}\times\mathcal{Z}^{i-\ell(i)}\times\mathcal{Z}^{j-d(j)}) \simeq \mathcal{S}_D, \tag{7}$$

où $\ell(i)$ est l'abscisse minimale telle que $(\ell(i),j)\in D$ et $d(j)$ est l'ordonnée minimale telle que $(i,d(j))\in D$. Ces spécifications se traduisent en séries génératrices :

$$P_{a,b}(z) = M_{a,b}(z) = \prod_{\substack{0\leq i<a\\ 0\leq j<b}}\frac{1}{1-z^{i+j+1}}, \tag{8}$$

$$S_D(z) = M_D(z) = \prod_{(i,j)\in D}\frac{1}{1-z^{h(i,j)}}. \tag{9}$$

où $h(i,j) = i-\ell(i)+j-d(j)+1$,

[22] Plus précisément des triplets de la forme $(\mathcal{Z},i,j)$, avec i et j des entiers positifs.

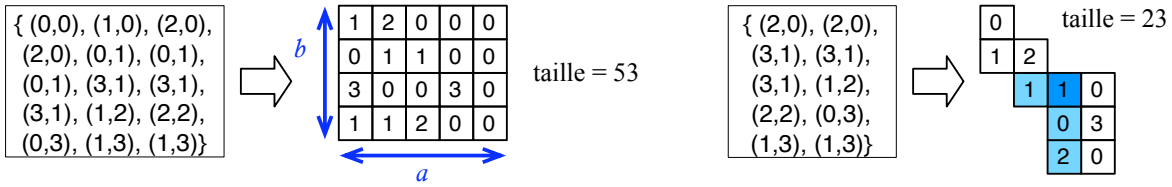


FIG. 3 – $\mathcal{M}_{a,b}$-structure, $\mathcal{M}_D$-structure, et diagrammes associés.

1.3 Diagrammes

Afin de présenter simplement l'algorithme de Pak pour transformer une $\mathcal{M}$-structure en partition plane, nous introduisons une description alternative des multi-ensembles de couples d'entiers sous la forme de *diagrammes* :

Définition 11. *Le* diagramme T *d'une* $\mathcal{M}$*-structure* M *est un tableau à deux dimensions d'entiers positifs ou nuls* $(m_{i,j})_{\mathbb{N}^2}$ *tels que* $m_{i,j}$ *est la multiplicité de* $(\mathcal{Z}, i, j)$ *dans* M*. La taille d'un diagramme* T *est définie de la façon suivante :*

$$|T| = \sum_{i,j} m_{i,j}(i + j + 1).$$

Cette définition reste valable pour les classes $\mathcal{M}_{a,b}$ et $\mathcal{M}_D$ en adaptant les formes des diagrammes aux domaines correspondants (voir exemples, figure 3). On peut également étendre la notion de rectangle enveloppant aux diagrammes.

La taille d'un diagramme, telle que nous venons de la définir, est exactement la taille du multi-ensemble de couples d'entiers correspondant. La taille associée à chaque couple de coordonnées du diagramme est équivalente à la notion classique de *longueur d'équerre*.

Définition 12. *L'*équerre *de la case* (i, j) *dans un tableau de forme* λ *est l'ensemble des cases qui se situent à gauche de* (i, j) *sur la même ligne et en dessous de* (i, j) *dans la même colonne, y compris elle-même. La longueur d'équerre* $h(i, j)$ *de la case* (i, j) *est le nombre total de cases qui composent l'équerre de* (i, j)*.*

L'équerre de la case $(2, 3)$ de la partition plane de la figure 1 et celle de la case $(3, 3)$ de la partition tordue de la figure 2 sont colorées en bleu. Elles sont toutes deux de longueur 4.

2 Générateurs de partitions planes

2.1 La bijection de Pak

Dans [Pak02] (et en résumé dans [Pak06]), Pak donne une preuve bijective pour la série génératrice des partitions planes inversées :

Définition 13. *Une partition plane* inversée *de forme* λ *est un tableau dont la forme est une partition d'entier* λ*, et dont les entrées sont des entiers positifs ou nuls, croissants en ligne et en colonne.*

Les partitions inversées sont en bijection avec les partitions tordues (la bijection consiste en une simple rotation de 180° sur les tableaux). Elles ont donc la même série génératrice[23]

[23] Aussi appelée "hook content formula".

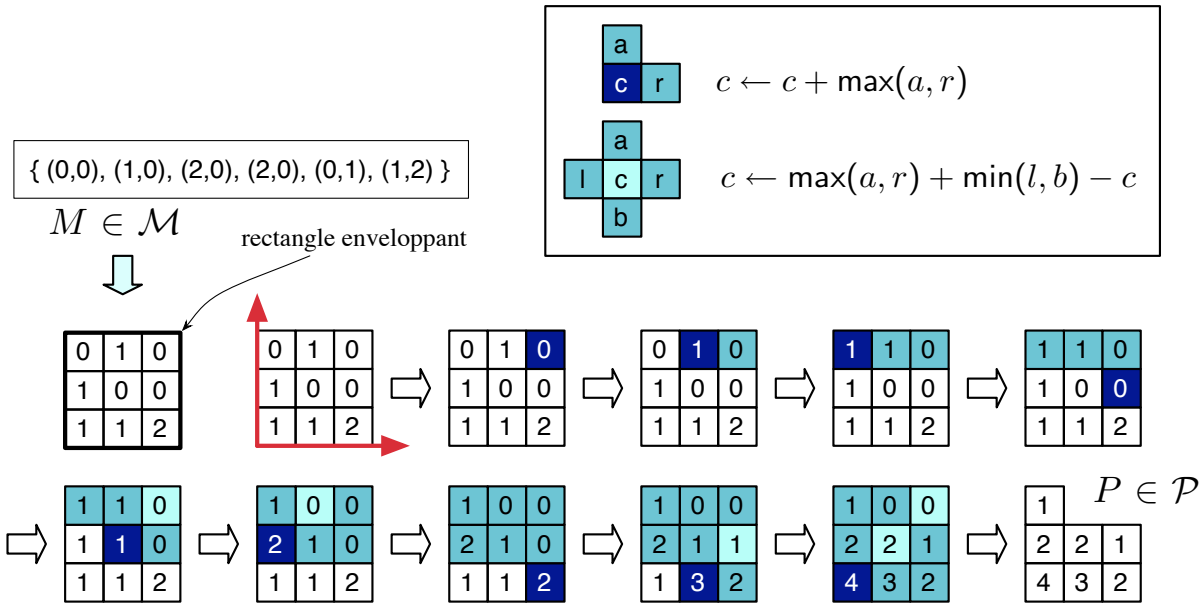


FIG. 4 – L'algorithme de Pak illustré sur un exemple.

et la preuve de Pak réalise une bijection effective entre les partions tordues et les diagrammes appartenant à la classe $\mathcal{M}_D$. Par spécialisation, cet algorithme fournit également la bijection entre les partitions planes ou encadrées et leur diagrammes correspondants.

Nous présentons ici une version modifiée de l'algorithme de Pak, adaptée à notre terminologie. Il reçoit en entrée un diagramme T de taille n dont on peut déterminer le rectangle enveloppant $(\ell \times w)$ et le transforme en une partition plane de taille n, dont le rectangle enveloppant est au plus $(\ell \times w)$. Le traitement effectué est séquentiel : on parcourt les cases du diagramme ligne par ligne, depuis le coin supérieur droit. Pour chaque case, on réalise une transformation locale dépendant uniquement des cases voisines, puis on parcourt les cases situées en diagonale de cette case, auxquelles on applique une seconde transformation, également locale. La complexité globale du processus est $O(M^3)$, où $M = max(\ell, w)$, dans le pire cas. La figure 4 présente un exemple d'exécution de cet algorithme.

Algo_Pak(T)
```
Entrée : un diagramme T de rectangle enveloppant (ℓ × w)
Sortie: une partition plane

pour i de ℓ à 1 faire
  pour j de w à 1 faire
    T[i,j] ← T[i,j] + max(T[j+1,i]), T[i,j+1])
    pour c de 1 à min(w − j, ℓ − i) faire
      x ← i + c
      y ← j + c
      T[x,y] ← max(T[x+1,y], T[x,y+1])
               + min(T[x−1,y], T[x,y−1]) − T[x,y]
renvoyer D
```

Bien que cet algorithme soit relativement simple, prouver qu'il réalise effectivement la bijection

entre $\mathcal{S}_D$ et $\mathcal{M}_D$ n'est pas trivial. L'idée est de montrer par induction que ce processus est réversible et qu'à chaque étape, l'ensemble des cases déjà parcourues forme bien une partition plane dont la taille correspond à celle du diagramme initial restreint au même ensemble de cases.

2.2 Générateurs de diagrammes

On sait désormais transformer un diagramme quelconque en sa partition plane associée. Le problème de la génération aléatoire des partitions planes peut alors se réduire à celui d'engendrer des diagrammes aléatoires avec une distribution uniforme sur leur taille. Étant donné que les diagrammes (et leur variantes) sont spécifiables à l'aide des constructions de multi-ensemble, de séquence et de produit, on peut assembler automatiquement les générateurs correspondants à partir des algorithmes présentés au chapitre 2. L'algorithme est adapté pour renvoyer le multi-ensemble directement sous la forme d'un diagramme T que l'on pourra ensuite transformer en partition plane en utilisant Algo_Pak(T).

$\Gamma\mathcal{M}(x)$
  $k \leftarrow$ `Indice_Max_MSet`$(\mathcal{Z} \times \mathbb{N}^2, x)$
  pour i **de** 1 **à** $k-1$ **faire**
    $p_i \leftarrow$ `Poiss`$(\frac{x^k}{k(1-x^k)^2})$
    répéter p_i **fois**
      $u \leftarrow$ `Geom`(x^i); $v \leftarrow$ `Geom`(x^i)
      $T[u,v] \leftarrow T[u,v] + i$
  $p_k \leftarrow$ `Poiss`$_{\geq 1}(\frac{x^k}{k(1-x^k)^2})$
  répéter p_k **fois**
    $u \leftarrow$ `Geom`(x^k); $v \leftarrow$ `Geom`(x^k)
    $T[u,v] \leftarrow T[u,v] + k$
  `renvoyer` T

Proposition 10. *Le processus $\Gamma\mathcal{P}(x)$ qui applique la procédure Algo_Pak à un diagramme produit par l'algorithme $\Gamma\mathcal{M}(x)$ est un générateur de Boltzmann libre pour la classe $\mathcal{P}$ des partitions planes.*

On pourrait tout à fait utiliser cet algorithme associé à une procédure de rejet pour engendrer également des diagrammes de $\mathcal{M}_{a,b}$ et $\mathcal{M}_D$, mais il y a une façon plus simple et plus efficace de procéder. En effet, nous sommes dans le cas particulier où l'on souhaite engendrer des multi-ensembles d'éléments appartenant à une classe combinatoire finie dont on sait énumérer les structures, puisqu'il s'agit des couples de cordonnées des cases du domaine fini D ($D = [1..a] \times [1..b]$ pour $\mathcal{M}_{a,b}$). Le générateur s'obtient alors comme une traduction directe de la spécification (7) :

$\Gamma\mathcal{M}_D(x)$
  pour (i,j) **dans** D **faire**
    $T[i,j] \leftarrow$ `Geom`$(x^{h(i,j)})$
  `renvoyer` T

Proposition 11. *Le processus $\Gamma\mathcal{S}_D(x)$ appliquant la procédure Algo_Pak au diagramme produit par l'algorithme $\Gamma\mathcal{M}_D(x)$ est un générateur de Boltzmann libre pour la classe des partitions tordues $\mathcal{S}_D$. En choisissant $D = [1..a] \times [1..b]$, on obtient un générateur de Boltzmann pour les partitions encadrées $\mathcal{P}_{a,b}$.*

3 Génération en taille exacte ou approchée

Comme nous l'avons vu dans les chapitres précédents, nous pouvons adapter les générateurs libres $\Gamma\mathcal{P}(x)$ et $\Gamma\mathcal{S}_D(x)$ pour qu'ils engendrent des partitions avec une taille n fixée, exacte ou approchée, en choisissant le paramètre x de telle sorte que l'espérance de la taille des partitions engendrées soit n. On peut alors analyser la complexité des générateurs ciblés ainsi obtenus en fonction de n. Cette section présente ce résultat de complexité ainsi que les principaux éléments qui permettent de le démontrer. La preuve complète et détaillée, due à Éric Fusy, se trouve dans [BFP07]. Elle implique l'utilisation d'outils d'analyse asymptotique tels que la transformée de Mellin et la méthode de col que nous ne présenterons pas ici[24].

3.1 Générateurs ciblés

La première étape est de déterminer comment choisir la valeur de x pour viser la taille n. Pour cela, on calcule l'espérance de la taille des partitions planes engendrées sous modèle de Boltzmann, en utilisant la formule donnée au chapitre 1. Pour les partitions planes, on obtient :

$$\mathbb{E}_x(N) = x\frac{P'(x)}{P(x)} = \frac{2\zeta(3)}{(1-x)^3} + O\left(\frac{1}{(1-x)^2}\right) \qquad \text{quand } x \to 1^-,$$

la deuxième expression impliquant l'utilisation de la transformée de Mellin. L'espérance de la taille des partitions tordues appartenant à $\mathcal{S}_D$ vaut quant à elle :

$$\mathbb{E}_{x,D}(N) = x\frac{S'_D(x)}{S_D(x)} = \frac{|D|}{(1-x)} \qquad \text{quand } x \to 1^-,$$

où $|D|$ est le nombre de cases appartenant au domaine D. Pour engendrer des partitions planes (resp. tordues) de taille n, on choisit comme valeur de paramètre la solution ξ_n de l'équation $\mathbb{E}_x(N) = n$ (resp. la solution ξ_n^D de $\mathbb{E}_{x,D}(N) = n$) :

$$\xi_n := 1 - (2\zeta(3)/n)^{1/3}, \qquad \xi_n^D := 1 - |D|/n. \tag{10}$$

Les générateurs de partitions planes ou tordues en taille approchée ou exacte sont alors obtenus en utilisant ces valeurs de paramètre, associées à une procédure de rejet. Étant donné que la taille d'une partition plane est la même que celle du diagramme dont elle est issue, ce rejet intervient avant même d'invoquer la procédure Algo_Pak. On note $\Gamma\mathcal{P}[n]$ le générateur de partitions planes de taille exactement n et $\Gamma\mathcal{P}[n,\varepsilon]$ celui en taille approchée (de même pour les partitions tordues). On rappelle qu'en taille approchée, on engendre des partitions dont la taille appartient à l'intervalle $[n(1-\varepsilon), n(1+\varepsilon)]$.

Théorème 4 ([BFP07])**.** *Les générateurs de partitions planes $\Gamma\mathcal{P}[n,\varepsilon]$ et $\Gamma\mathcal{P}[n]$ ont respectivement les complexités arithmétiques réelles :*

- $O(n\ln(n)^3)$*, en taille approchée,*
- $O(n^{\frac{4}{3}})$*, en taille exacte.*

Les générateurs de partitions tordues $\Gamma\mathcal{S}_D[n,\varepsilon]$ et $\Gamma\mathcal{S}_D[n]$ ont respectivement les complexités arithmétiques réelles :

- $O(1)$ *quand $n \to \infty$, en taille approchée,*
- $O(|D|n)$*, en taille exacte.*

[24] Voir [FGD95] pour une étude détaillé sur la transformée de Mellin et [FS08, chap. 8] pour la méthode de col.

La table 1 donne les temps de génération et de transformation obtenus avec les algorithmes en taille approchée présentés ci-dessus. Ces mesures ont été effectuées avec un processeur Intel à 3.2GHz et 2Go de mémoire. Le générateur est écrit en Maple 10 et la procédure Algo_Pak en OCaml. Les figures 6 et 7 (en fin de chapitre) donnent des exemples de partitions planes, encadrées et tordues obtenues avec ces générateurs. Les dessins sont réalisés avec Gnuplot.

taille visée	10^3	10^4	10^5	10^6	10^7
$\Gamma\mathcal{M}$	0.052 sec.	0.27 sec.	1.25 sec.	5.88 sec.	39.23 sec.
bijection	0.001 sec.	0.016 sec.	0.401 sec.	7.766 sec.	2 min. 46 sec.
$\Gamma\mathcal{M}_{10,10}$	0.038 sec.	0.039 sec.	0.039 sec.	0.039 sec.	0.039 sec.
bijection	0.00 sec.	0.00 sec.	0.00 sec.	0.00 sec.	0.00 sec.
$\Gamma\mathcal{M}_{50,50}$	1.14 sec.	1.16 sec.	1.18 sec.	1.22 sec.	1.24 sec.
bijection	0.001 sec.	0.003 sec.	0.004 sec.	0.004 sec.	0.004 sec.
$\Gamma\mathcal{M}_{100,100}$	4.58 sec.	4.63 sec.	4.81 sec.	5.28 sec.	5.97 sec.
bijection	0.001 sec.	0.012 sec.	0.021 sec.	0.024 sec.	0.024 sec.

Tab. 1 – Temps de génération pour différentes tailles de partitions (planes ou encadrées).

3.2 Complexité

Pour analyser la complexité des générateurs en taille approchée et exacte, il faut dans un premier temps déterminer en fonction de x la complexité des générateurs libres. On pourra alors en déduire la complexité des générateurs de diagrammes paramétrés par ξ_n ou ξ_n^D. La complexité globale du générateur de partitions planes dépend également du nombre moyen de rejets effectués pour obtenir un diagramme de taille n (ou approchée). Enfin, s'ajoute à cela le coût de la procédure Algo_Pak sur un diagramme de taille n. En utilisant la notation $\Lambda\mathcal{A}$ pour la complexité d'un générateur de $\mathcal{A}$-structures, on a :

$$\Lambda\mathcal{P}[n] = \Lambda\mathcal{M}(\xi_n)\cdot\pi_n + \mathbb{E}_n(\text{Algo\_Pak}) \quad \text{et} \quad \Lambda\mathcal{P}[n,\varepsilon] = \Lambda\mathcal{M}_D(\xi_n)\cdot\pi_{n,\varepsilon} + \mathbb{E}(\text{Algo\_Pak}), \quad (11)$$

où π_n (resp. $\pi_{n,\varepsilon}$) est le nombre moyen d'essais avant d'obtenir un diagramme de taille n (resp. approximativement n) et $\mathbb{E}_n(\text{Algo\_Pak})$ est l'espérance de la complexité de l'algorithme Algo_Pak appliqué à un diagramme de taille n.

Pour calculer $\Lambda\mathcal{M}(x)$, on utilise le fait que le tirage d'un loi géométrique se fait avec un coût constant et on remarque que la complexité de la génération correspond alors à une somme de lois de Poisson de paramètre $\frac{x^i}{i(1-x^i)^2}$, pour $i \geq 1$. Enfin, en utilisant la transformée de Mellin, on obtient :

$$\Lambda\mathcal{M}(x) = \underset{x\to 1^-}{O}\left(\frac{1}{(1-x)^2}\right).$$

D'après (10), la complexité du générateur ciblé est donc :

$$\Lambda\mathcal{M}(\xi_n) = O(n^{\frac{4}{3}}).$$

Sous modèle de Boltzmann, les tailles des partitions planes ont une distribution concentrée. La figure 5 représente cette distribution pour différentes valeurs de x. Le rejet est donc relativement limité. En utilisant l'inégalité de Chebyshev pour calculer $\pi_{n,\varepsilon}$ et la transformée de Mellin

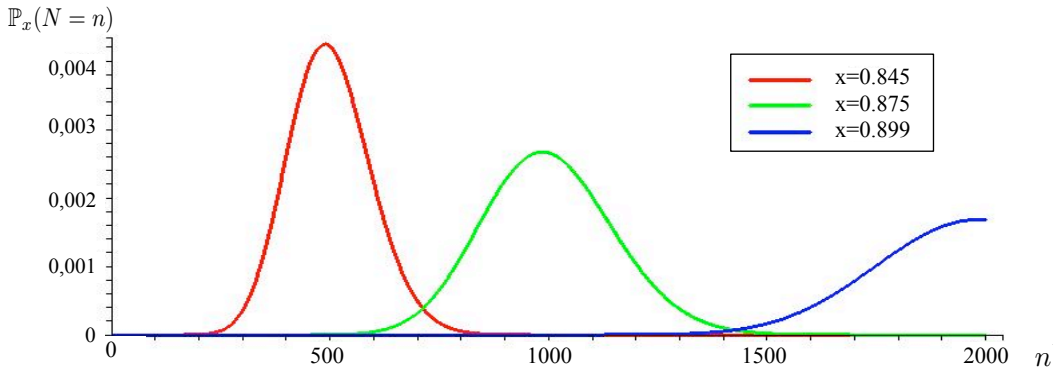


FIG. 5 – Distribution des tailles des partitions planes pour différentes valeurs de x.

ainsi que la méthode de col pour π_n, on obtient :

$$\pi_{n,\varepsilon} \underset{n\to\infty}{\to} 1, \qquad \pi_n \underset{n\to\infty}{\sim} n^{2/3}/c, \text{ avec } c \approx 0.1023.$$

Enfin, nous avons besoin de calculer l'espérance de la complexité de l'algorithme Algo_Pak. Nous avons vu que cet algorithme est cubique en la plus grande dimension M du rectangle enveloppant du diagramme passé en paramètre. Pour évaluer la complexité de la procédure Algo_Pak sur un diagramme de taille n, nous devons estimer la taille du rectangle enveloppant d'un diagramme de taille n. On va, en fait, analyser un paramètre plus simple : la longueur d'équerre maximale $h_{max}(n)$ d'un diagramme de taille n. Étant donné que pour un diagramme de longueur M, la longueur h_{max} vaut au moins M et au plus $2M$, la procédure Algo_Pak est également cubique en h_{max}. En utilisant la méthode de col, on sait analyser $h_{max}(n)$:

$$h_{max}(n) \underset{n\to\infty}{=} O(n^{\frac{1}{3}} \log n),$$

et on obtient :

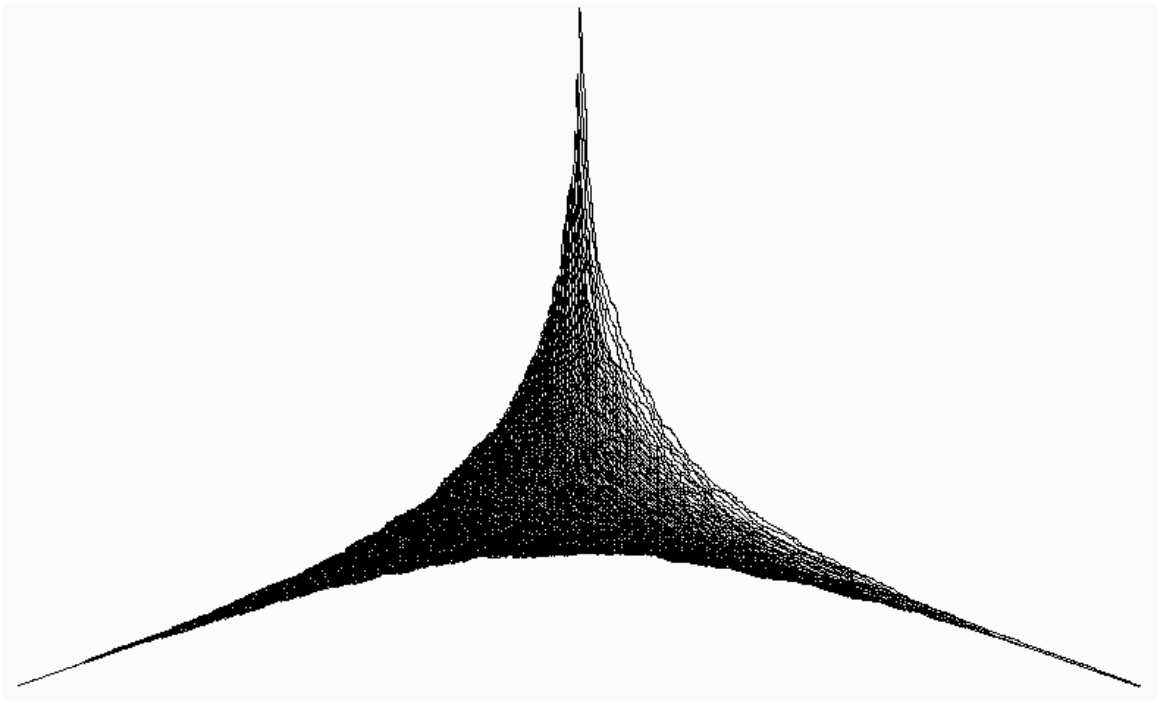
$$\mathbb{E}_n(\text{Algo\_Pak}) = O(n \log^3 n).$$

L'équation (11) nous permet alors d'obtenir les complexités du théorème 4.

Dans le cas des partitions tordues, l'analyse est simplifiée par le fait que la complexité du générateur ciblé et de la procédure Algo_Pak ne dépendent que de la taille du domaine choisi :

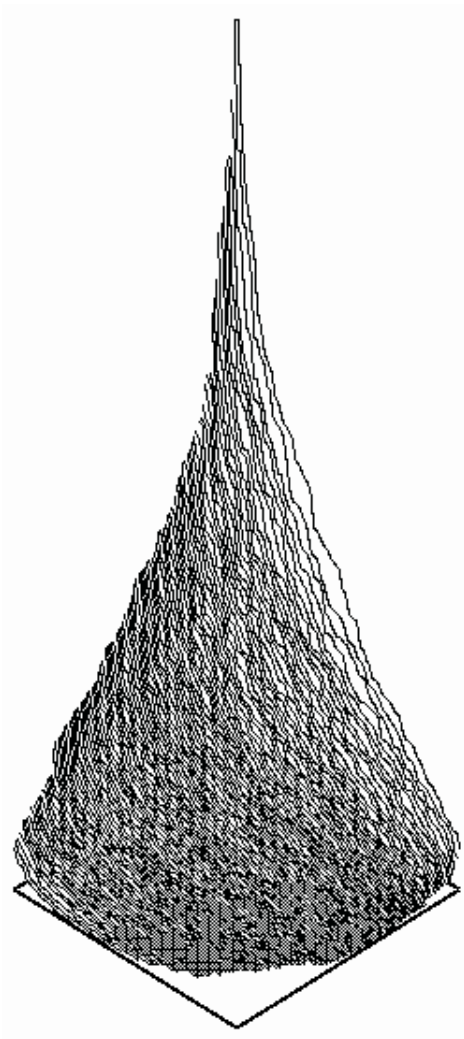
$$\Lambda\mathcal{M}_D(\xi_n^D) = O(1) \qquad \text{et} \qquad \mathbb{E}_n(\text{Algo\_Pak}) = O(1).$$

En moyenne, le nombre d'essais à faire avant d'obtenir un diagramme de taille n avec le générateur $\Gamma\mathcal{M}_D(\xi_n^D)$ est $O(1)$ en taille approchée et $O(n)$ en taille exacte.

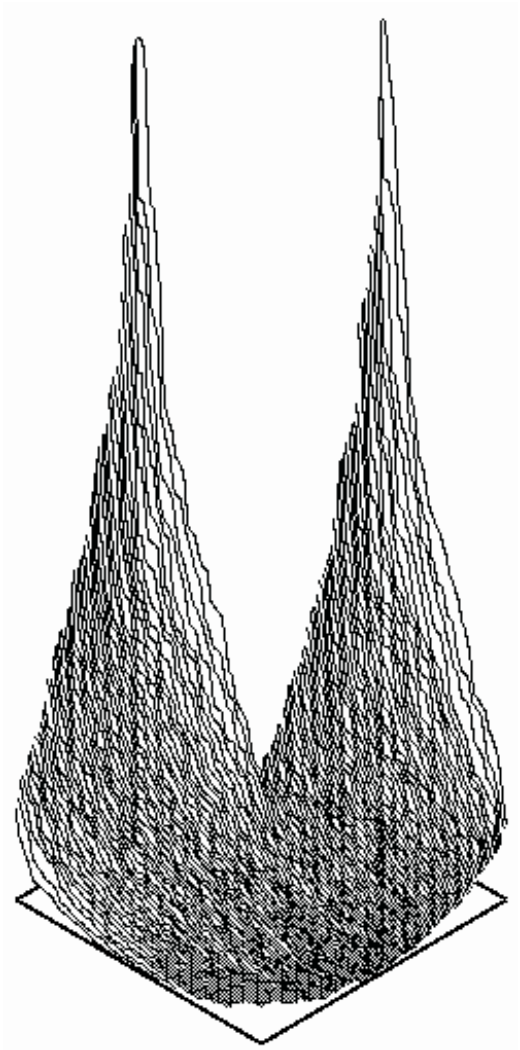


génération : ~ 7 sec. bijection : ~ 280 sec.

FIG. 6 – Une partition plane aléatoire de taille 1005749 engendrée avec $\Gamma\mathcal{P}(0.9866)$.



génération : ~ 5 sec.
bijection : ~ 0.7 sec.



génération : ~ 4 sec.
bijection : ~ 0.35 sec.

FIG. 7 – Une partition (100×100)-encadrée, de taille 999400, engendrée avec $\Gamma\mathcal{P}_{100,100}(0.9931)$ et une partition tordue dans le domaine $D = [0..99] \times [0..99] \backslash [0..49] \times [0..49]$, de taille 1005532, engendrée avec $\Gamma\mathcal{S}_D(0.9942)$.

Deuxième partie

Oracles pour les systèmes combinatoires

Chapitre 4

Éléments de théorie des espèces et itération combinatoire

Sommaire

Nous présentons ici une introduction succincte à la théorie des espèces de structures combinatoires de Joyal [Joy81], développée ensuite par Bergeron, Labelle et Leroux dans [BLL98], qui nous servira de base pour présenter les itérations combinatoires. La description récursive d'espèces arborescentes par le biais de systèmes d'équations fonctionnelles joue un rôle central dans cette théorie. Le théorème des espèces implicites de Joyal nous assure, sous des hypothèses très générales, l'existence et l'unicité des espèces décrites par ce type de systèmes. Nous donnons une preuve légèrement remaniée de ce théorème afin d'introduire les outils que nous utiliserons ensuite et notamment certaines propriétés suffisantes pour la convergence de suites d'espèces définies par itération. Ces itérations combinatoires sont le point de départ de notre méthode de calcul d'oracle. Nous concluons ce chapitre par des exemples d'espèces (arbres généraux planaires et graphes série-parallèle) produites par itération de point fixe. Le chapitre suivant sera consacré à des itérations dont la convergence est plus rapide.

1 Espèces de structures combinatoires

De façon informelle, une *espèce de structures* est une règle (ou une construction) $\mathcal{F}$ qui associe à chaque ensemble fini U un ensemble fini $\mathcal{F}[U]$ indépendant de la nature des éléments de U. Cela correspond à la notion de classe combinatoire que nous avons introduite au début de ce mémoire. Les éléments de $\mathcal{F}[U]$ sont des structures combinatoires étiquetées par les éléments de U et définies par la règle $\mathcal{F}$. L'indépendance entre la règle $\mathcal{F}$ et la nature des éléments de U traduit une invariance structurelle par réétiquetage.

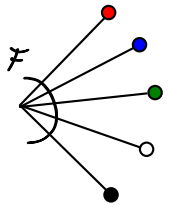


FIG. 1 – Exemple de $\mathcal{F}$-structure.

Dans cette première section, nous avons extrait de [BLL98][25] la définition précise d'une espèce de structures, ainsi que quelques notions complémentaires utiles pour la suite.

1.1 Définitions

Espèces de structures

Définition 14. *Une* espèce de structures *est une règle*[26] $\mathcal{F}$ *qui produit :*

1. *pour chaque ensemble fini U, un ensemble fini $\mathcal{F}[U]$,*
2. *pour chaque bijection $\sigma : U \to V$, une fonction $\mathcal{F}[\sigma] : \mathcal{F}[U] \to \mathcal{F}[V]$*

Les fonctions $\mathcal{F}[\sigma]$ doivent, de plus, satisfaire les propriétés suivantes :

- *pour toutes les bijections $\sigma : U \to V$ et $\tau : V \to W$,*

$$\mathcal{F}[\tau \circ \sigma] = \mathcal{F}[\tau] \circ \mathcal{F}[\sigma],$$

- *pour toute fonction identité $\mathrm{Id}_U : U \to U$,*

$$\mathcal{F}[\mathrm{Id}_U] = \mathrm{Id}_{\mathcal{F}[U]}.$$

Un élément γ appartenant à $\mathcal{F}[U]$ est appelé une $\mathcal{F}$-structure sur U (ou encore une structure de l'espèce $\mathcal{F}$ sur U). On dira que U est l'ensemble *sous-jacent* de γ, ou que γ est *portée* par U. La fonction $\mathcal{F}[\sigma]$ est appelée le *transport* des $\mathcal{F}$-structures suivant σ. Pour représenter une $\mathcal{F}$-structure générique, on utilisera souvent des schémas similaires à celui de la figure 1. Les ronds colorés représentent les différents éléments de l'ensemble sous-jacent U de la $\mathcal{F}$-structure. L'arc de cercle indique que les points de U portent la structure $\mathcal{F}$.

Exemple 14. L'espèce $\mathcal{G}$ des graphes simples (non orientés, sans boucle et sans arête multiple) et leurs isomorphismes est définie, pour un ensemble fini V, par

$$\mathcal{G}[V] = \{g \,|\, g = (V, E)\}, \text{ tel que } E \text{ est un sous-ensemble de } \{\langle \alpha, \beta \rangle \,|\, \alpha \neq \beta \text{ et } \alpha, \beta \in V\}.$$

Pour chaque ensemble V, les $\mathcal{G}$-structures sur V sont les graphes dont l'ensemble des sommets est V. Pour chaque bijection σ de V dans W, $\mathcal{G}[\sigma]$ transforme tout graphe de $\mathcal{G}[V]$ en un graphe de $\mathcal{G}[W]$ en réétiquetant les sommets suivant σ. La figure 2 donne un exemple γ de $\mathcal{G}$-structure sur $V = \{a, b, c, d, e, f\}$, ainsi que le transport de γ par la bijection σ décrite par la même figure. Le graphe obtenu est *isomorphe* à γ.

[25] Une introduction à la théorie des espèces est disponible en version électronique à l'adresse : `http://bergeron.math.uqam.ca/Site/bergeron_uqam_files/book.pdf`.

[26] On parle également de *foncteur*.

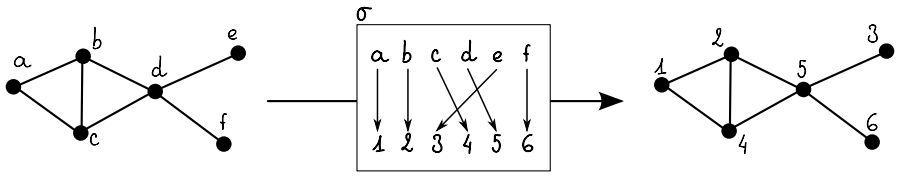


FIG. 2 – Transport d'une $\mathcal{G}$-structure.

Isomorphismes Intuitivement, deux structures isomorphes peuvent être considérées comme identiques si la nature des éléments de leurs ensembles sous-jacents est ignorée. Cette "forme générale" que les structures isomorphes ont en commun est leur *type d'isomorphisme*. On le représente par un schéma similaire à celui d'une $\mathcal{F}$-structure, mais les éléments de l'ensemble sous-jacent sont rendus indistinguables (représentés par des ronds noirs dans les schémas). La structure obtenue est *non étiquetée*.

Définition 15. *Soient deux $\mathcal{F}$-structures $\gamma_1 \in \mathcal{F}[U]$ et $\gamma_2 \in \mathcal{F}[V]$. Une bijection $\sigma : U \to V$ est appelée un* isomorphisme *de γ_1 vers γ_2 si $\gamma_2 = \mathcal{F}[\sigma](\gamma_1)$. On dit que γ_1 et γ_2 ont le même* type d'isomorphisme. *Un isomorphisme de γ vers γ est appelé un* automorphisme *de γ.*

Définition 16. *Soient $\mathcal{F}$ et $\mathcal{G}$ deux espèces de structures. Un* isomorphisme *de $\mathcal{F}$ vers $\mathcal{G}$ est une famille de bijections $\alpha_U : \mathcal{F}[U] \to \mathcal{G}[U]$ telle que, pour toute bijection $\sigma : U \to V$ et pour toute $\mathcal{F}$-structure γ, on a :*

$$\sigma(\alpha_U(\gamma)) = \alpha_V(\sigma(\gamma)).$$

En d'autre termes, le diagramme suivant commute :

$$\begin{array}{ccc} \mathcal{F}[U] & \xrightarrow{\alpha_U} & \mathcal{G}[U] \\ {\scriptstyle \mathcal{F}[\sigma]}\downarrow & & \downarrow{\scriptstyle \mathcal{G}[\sigma]} \\ \mathcal{F}[V] & \xrightarrow{\alpha_V} & \mathcal{G}[V] \end{array}$$

On dira dans ce cas que les espèces $\mathcal{F}$ et $\mathcal{G}$ sont isomorphes, ce que l'on note $\mathcal{F} \simeq \mathcal{G}$.

Espèces multi-sortes Une extension naturelle des espèces de structures est la notion d'espèce multi-sortes, analogue des fonctions à plusieurs variables. On peut, par exemple, décrire l'espèce des arbres comme une espèce à deux sortes : les nœuds internes et les feuilles. Le transport de structure doit alors conserver la distinction entre nœuds internes et feuilles.

Soit $\mathbf{U}$ un k-uplet $(U_1, \ldots, U_k)$. Un élément $u \in U_i$ est appelé un élément de $\mathbf{U}$ de sorte i. La multi-cardinalité de $\mathbf{U}$ est le k-uplet de cardinalités $|\mathbf{U}| = (|U_1|, \ldots, |U_k|)$. La cardinalité totale de $\mathbf{U}$ est la somme $||\mathbf{U}|| = |U_1| + \ldots + |U_k|$. Une multi-fonction f de $(U_1, \ldots, U_k)$ dans $(V_1, \ldots, V_k)$ est un k-uplet de fonctions $(f_1, \ldots, f_k)$ tel que $f_i : U_i \to V_i$ pour $i = 1, \ldots, k$. La composition de deux multi-fonctions est obtenue par composition des f_i.

Définition 17. *Soit $k \geq 1$ un entier. Une* espèce à m sortes *est une règle $\mathcal{F}$ qui produit :*

1. *pour chaque m-uplet fini $\mathbf{U} = (U_1, \ldots, U_m)$, un ensemble fini $\mathcal{F}[U_1, \ldots, U_m]$*
2. *pour chaque multi-fonction $\sigma = (\sigma_1, \ldots, \sigma_m) : (U_1, \ldots, U_m) \to (V_1, \ldots, V_m)$, une fonction :*

$$\mathcal{F}[\sigma] = \mathcal{F}[\sigma_1, \ldots, \sigma_m] : \mathcal{F}[U_1, \ldots, U_m] \to \mathcal{F}[V_1, \ldots, V_m],$$

telle que, quelles que soient σ et τ, $\mathcal{F}[\tau \circ \sigma] = \mathcal{F}[\tau] \circ \mathcal{F}[\sigma]$ et $\mathcal{F}[\mathrm{Id}_{\mathbf{U}}] = \mathrm{Id}_{\mathcal{F}[\mathbf{U}]}$.

Description des espèces Avec ces définitions des espèces et des espèces multi-sortes, il est possible de décrire la règle $\mathcal{F}$ et la fonction de transport $\mathcal{F}[\sigma]$ de différentes façons. Nous ne retenons ici que celles qui nous seront utiles par la suite.

Quand les structures d'une espèce sont particulièrement simples ou peu nombreuses, il peut être avantageux de la définir par une description explicite des ensembles $\mathcal{F}[U]$ et des fonctions[27] $\mathcal{F}[\sigma]$. Les exemples suivants appartiennent à cette catégorie :

- L'espèce E des ensembles est simplement définie par $E[U] = \{U\}$. Pour chaque ensemble U, il existe une unique E-structure, l'ensemble U lui-même.
- L'espèce $\mathcal{Z}$, caractéristique des singletons, définie par

  $$\mathcal{Z}[U] = \{U\} \text{ si } |U| = 1 \text{ et } \varnothing \text{ sinon.}$$

  C'est l'équivalent de l'atome $\mathcal{Z}$ utilisé pour construire les structures décomposables. Dans les schémas, nous identifions les $\mathcal{Z}$-structures à l'unique élément de leur ensemble sous-jacent (représenté par un rond noir).
- L'espèce 1, caractéristique de l'ensemble vide, définie par

  $$1[U] = \{U\} \text{ si } |U| = 0 \text{ et } \varnothing \text{ sinon.}$$

  C'est l'équivalent de l'atome $\mathcal{E}$ de taille 0 des structures décomposables.
- L'espèce vide, notée 0, définie par $0[U] = \varnothing$ pour tout U.

On peut également décrire les espèces à partir de constructions ensemblistes. C'est le cas de la description des graphes simples que nous avons donnée dans l'exemple 14. Ainsi, on peut notamment décrire l'espèce L des ordres linéaires[28] et l'espèce C des permutations cycliques qui sont respectivement des séquences et des cycles. Par la suite, nous utiliserons également l'espèce $\mathcal{T}$ des arbres enracinés ou encore l'espèce End des endofonctions[29] qui peuvent également être définies ainsi.

Une autre façon de produire des espèces de structures est d'appliquer des opération combinatoires à des espèces connues. Nous présentons ces opérations dans les sections suivantes. Enfin, nous verrons dans la section 2 qu'il est également possible de décrire des espèces de structures à l'aide de systèmes d'équations fonctionnelles. Cette façon de décrire les espèces va jouer un rôle essentiel dans les traitements combinatoires présentés au prochain chapitre.

1.2 Addition, multiplication

Définition 18. *Soient $\mathcal{F}$ et $\mathcal{G}$ deux espèces de structures, l'espèce $\mathcal{F} + \mathcal{G}$, appelée la* somme *de $\mathcal{F}$ et $\mathcal{G}$, est définie comme suit : une $(\mathcal{F} + \mathcal{G})$-structure sur U est soit une $\mathcal{F}$-structure, soit une $\mathcal{G}$-structure. En d'autres termes, pour tout ensemble fini U, on a :*

$$(\mathcal{F} + \mathcal{G})[U] = \mathcal{F}[U] + \mathcal{G}[U] \qquad \textit{où } + \textit{ désigne l'union disjointe.}$$

Le transport suivant la bijection $\sigma : U \to V$ est obtenu, pour toute $(\mathcal{F} + \mathcal{G})$-structure γ, par :

$$(\mathcal{F} + \mathcal{G})[\sigma](\gamma) = \begin{cases} \mathcal{F}[\sigma](\gamma) & \textit{si } \gamma \in \mathcal{F}[U], \\ \mathcal{G}[\sigma](\gamma) & \textit{si } \gamma \in \mathcal{G}[U]. \end{cases}$$

[27] Dans les exemples suivants, le transport des structures est trivial.

[28] On utilisera alternativement l'espèce isomorphe $\mathcal{S}$ des permutations, lorsque l'on veut représenter les séquences sous la forme d'ensembles de cycles.

[29] On les appelle également des graphes fonctionnels, voir exemple 18, page 84.

L'addition d'espèces est associative et commutative à isomorphisme près. De plus l'espèce vide 0 est un élément neutre pour l'addition : $\mathcal{F} + 0 = 0 + \mathcal{F} = \mathcal{F}$, quelle que soit l'espèce $\mathcal{F}$.

Exemple 15. L'espèce E des ensembles peut être définie comme la somme $E_0 + E_1$ où E_0 (resp. E_1) est l'espèce des ensembles ayant un nombre pair (resp. impair) d'éléments.

Une espèce $\mathcal{G}$ est une *sous-espèce* de $\mathcal{F}$, notée $\mathcal{G} \subseteq \mathcal{F}$, quand, pour tout ensemble U, on a $\mathcal{G}[U] \subseteq \mathcal{F}[U]$. La soustraction $\mathcal{C} = \mathcal{F} - \mathcal{G}$ est alors définie par l'équation $\mathcal{F} = \mathcal{G} + \mathcal{C}$.

Exemple 16. L'espèce E^+ des ensembles non vides est définie par $E^+ = E - 1$.

Définition 19. *Soient $\mathcal{F}$ et $\mathcal{G}$ deux espèces de structures ; l'espèce $\mathcal{F} \cdot \mathcal{G}$ (aussi notée $\mathcal{F} \times \mathcal{G}$ ou $\mathcal{FG}$), appelée le* produit *de $\mathcal{F}$ et $\mathcal{G}$ est définie comme suit : une $(\mathcal{F} \cdot \mathcal{G})$-structure sur U est une paire $\gamma = (\beta, \delta)$ telle que :*

1. *β est une $\mathcal{F}$-structure sur un sous-ensemble $U_1 \subseteq U$,*
2. *δ est une $\mathcal{G}$-structure sur un sous-ensemble $U_2 \subseteq U$,*
3. *$U_1 + U_2 = U$.*

En d'autres termes, pour tout ensemble fini U, on a :

$$(\mathcal{F} \cdot \mathcal{G})[U] = \sum_{U_1 + U_2 = U} \mathcal{F}[U_1] \times \mathcal{G}[U_2].$$

Le transport suivant la bijection $\sigma : U \to V$ est, pour toute $(\mathcal{F} \cdot \mathcal{G})$-structure $\gamma = (\beta, \delta)$:

$$(\mathcal{F} \cdot \mathcal{G})[\sigma](\gamma) = (\mathcal{F}[\sigma_1](\gamma), \mathcal{G}[\sigma_2](\gamma)) \quad \text{où } \sigma_i \text{ est la restriction de } \sigma \text{ à } U_i.$$

Le produit d'espèces est associatif et commutatif à isomorphisme près, mais en général $\mathcal{F} \cdot \mathcal{G}$ et $\mathcal{G} \cdot \mathcal{F}$ ne sont pas identiques. L'espèce 0 est un élément absorbant pour la multiplication et l'espèce 1 est un élément neutre : $\mathcal{F} \cdot 0 = 0 \cdot \mathcal{F} = 0$ et $\mathcal{F} \cdot 1 = 1 \cdot \mathcal{F} = \mathcal{F}$, pour toute espèce $\mathcal{F}$. Enfin, la multiplication est distributive par rapport à l'addition.

Exemple 17. L'espèce $\mathcal{D}$ des dérangements (permutations sans point fixe) peut être définie par l'équation $\mathcal{S} = E \cdot \mathcal{D}$, où $\mathcal{S}$ est l'espèce des permutations. Cette relation s'interprète par le fait que toute permutation peut être décomposée de façon canonique comme le produit d'un ensemble de points fixes et d'un ensemble de cycles de longueur ≥ 2.

Les notions usuelles de vecteurs (on parlera également de *familles* d'espèces) et de matrices s'étendent aux espèces de structures. On utilise, par convention, des lettres grasses pour les décrire. Une $\boldsymbol{\mathcal{F}}$-structure est une structure qui appartient à l'une des espèces qui composent la famille $\boldsymbol{\mathcal{F}} = (\mathcal{F}_1, \ldots, \mathcal{F}_m)$. Les sommes et les produits de vecteurs et de matrices d'espèces sont définis comme habituellement, en utilisant des additions et des produits d'espèces.

1.3 Substitution

Définition 20. *Soient $\mathcal{F}$ et $\mathcal{G}$ deux espèces de structures telles que $\mathcal{G}[\varnothing] = \varnothing$ (il n'y a pas de $\mathcal{G}$-structure sur un ensemble vide). L'espèce $\mathcal{F} \circ \mathcal{G}$, aussi notée $\mathcal{F}(\mathcal{G})$, appelée la* composée *de $\mathcal{G}$ par $\mathcal{F}$, est définie comme suit : une $(\mathcal{F} \circ \mathcal{G})$-structure sur U est un triplet $\gamma = (\pi, \varphi, \delta)$ tel que :*

1. *π est une partition de U,*
2. *φ est une $\mathcal{F}$-structure sur l'ensemble des classes de π,*
3. *$\delta = (\delta_p)_{p \in \pi}$ où, pour chaque partie p de π, δ_p est une $\mathcal{G}$-structure sur p.*

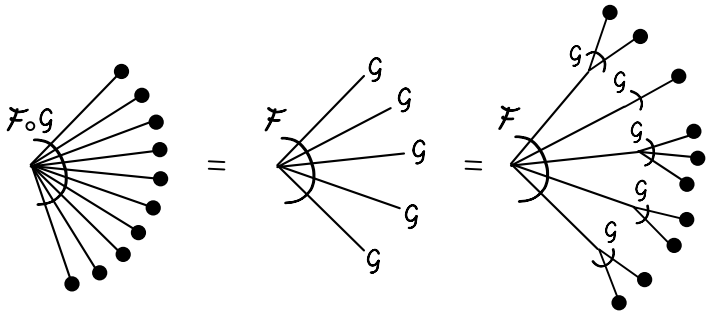


FIG. 3 – Exemple de $\mathcal{F} \circ \mathcal{G}$-structure.

En d'autres termes, pour tout ensemble fini U, on a :

$$(\mathcal{F} \circ \mathcal{G})[U] = \sum_{\pi \in \mathfrak{P}(U)} \mathcal{F}[\pi] \times \prod_{p \in \pi} \mathcal{G}[p] \qquad \text{avec } \mathfrak{P}(U) \text{ l'ensemble des partitions de } U.$$

Le transport suivant la bijection $\sigma : U \to V$ est, pour toute $(\mathcal{F} \circ \mathcal{G})$-structure $\gamma = (\pi, \varphi, \delta)$:

$$(\mathcal{F} \circ \mathcal{G})[\sigma](\gamma) = (\sigma(\pi), \mathcal{F}[\sigma](\varphi), (\mathcal{G}[\sigma](\delta_p))_{p \in \pi}).$$

L'espèce $\mathcal{F}(\mathcal{G})$ est le résultat de la substitution de $\mathcal{G}$ dans $\mathcal{F}$. On dira qu'une structure de l'espèce $\mathcal{F}(\mathcal{G})$ est une *$\mathcal{F}$-assemblée* de $\mathcal{G}$-structures. Ces $\mathcal{G}$-structures seront désignées comme les *membres* de la $\mathcal{F}$-assemblée. La figure 3 représente cette notion de façon schématique.

Exemple 18. L'espèce *End* des endofonctions est décrite de façon ensembliste par :

$$End[U] = \{\psi \mid \psi : U \to U\}.$$

Une endofonction ψ peut être représentée par un graphe dirigé fonctionnel. Ce graphe possède deux sortes de sommets : les points récurrents, *i.e.*, les éléments x pour lesquels il existe $k > 0$ tels $\psi(x)^k = x$, qui sont situés sur des cycles, et les points non récurrents. Une endofonction peut donc naturellement être identifiée à un ensemble de cycles d'arbres enracinés, c'est-à-dire, une permutation de $\mathcal{T}$-structures :

$$End = \mathcal{S} \circ \mathcal{T} = \mathcal{S}(\mathcal{T}).$$

Dans un contexte multi-sortes, la définition d'une $\mathcal{F}$-assemblée de $\mathcal{G}_1, \ldots, \mathcal{G}_k$-structures devient, pour $\mathbf{U} = (U_1, \ldots, U_d)$:

$$\mathcal{F}(\mathcal{G}_1, \ldots, \mathcal{G}_k)[\mathbf{U}] = \sum_{\substack{\pi \in \mathfrak{P}(\mathbf{U}) \\ \pi_1 + \cdots + \pi_k = \pi}} \mathcal{F}[\pi_1, \ldots, \pi_k] \times \prod_{\pi_i \in \pi} \prod_{s \in \pi_i} \mathcal{G}_i[s]$$

1.4 Dérivation

Définition 21. *Soit $\mathcal{F}$ une espèce de structures. L'espèce $\mathcal{F}'$ (également notée $\partial\mathcal{F}/\partial\mathcal{Z}$), appelée la* dérivée *de $\mathcal{F}$, est définie comme suit : une $\mathcal{F}'$-structure sur U est une $\mathcal{F}$-structure sur $U^+ = U \cup \{\star\}$, où $\star$ est un élément n'appartenant pas à U. En d'autre termes, pour tout ensemble fini U, on a :*

$$\mathcal{F}'[U] = \mathcal{F}[U^+], \qquad \text{avec } U^+ = U \cup \{\star\}.$$

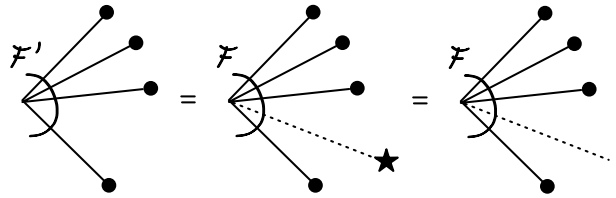


FIG. 4 – Exemple de $\mathcal{F}'$-structure.

Le transport suivant la bijection $\sigma : U \to V$ est obtenu, pour toute $(\mathcal{F}')$-structure γ, par :

$$(\mathcal{F}')[\sigma](\gamma) = (\mathcal{F})[\sigma^+](\gamma),$$

où $\sigma^+ : U^+ \to V^+$ est l'extension canonique de σ telle que $\sigma^+(u) = \sigma(u)$ si $u \in U$ et $\sigma^+(\star) = \star$.

L'élément $\star$ est appelé un *bourgeon*[30] et correspond à une *branche libre* dans une $\mathcal{F}$-structure. La figure 4 représente de façon schématique la dérivée d'une $\mathcal{F}$-structure. Dans les schémas, on oubliera le plus souvent le bourgeon $\star$ car la branche libre en pointillé suffit à marquer son emplacement. La composée $(\partial\mathcal{F}/\partial\mathcal{Z}) \circ \mathcal{G}$ de $\mathcal{G}$ par $\partial\mathcal{F}/\partial\mathcal{Z}$ est directement notée $\partial\mathcal{F}/\partial\mathcal{G}$. On interprète généralement la dérivée d'une espèce $\mathcal{F}$ par rapport à $\mathcal{G}$ comme l'ensemble des $\mathcal{F}$-assemblées dont l'un des membres appartenant à l'espèce $\mathcal{G}$ aurait été *effacé*.

Exemple 19. Intuitivement, si l'on efface un élément dans une permutation circulaire, les éléments qui restent forment un ordre linéaire. En effet, l'espèce des ordre linéaires L peut être définie comme la dérivée de l'espèce C des permutations circulaires : $L = C'$.

Pour toute espèce $\mathcal{C}$, le produit de $\partial\mathcal{F}/\partial\mathcal{G}$ et de $\mathcal{C}$ peut être interprété soit de façon classique, soit comme le remplacement du bourgeon de $\partial\mathcal{F}/\partial\mathcal{G}$ par une $\mathcal{C}$-structure. Cela se traduit par l'isomorphisme suivant :

$$\frac{\partial\mathcal{F}}{\partial\mathcal{G}} \times \mathcal{C} \simeq \left.\frac{\partial\mathcal{F}}{\partial\mathcal{G}}\right|_{\star\,=\,\mathcal{C}}.$$

Par la suite, c'est cette deuxième interprétation du produit par une espèce dérivée que nous utiliserons. Dans le cas particulier où $\mathcal{C}$ est une sous-espèce de $\mathcal{G}$, cette opération correspond au pointage d'une $\mathcal{G}$-structure dans une $\mathcal{F}$-assemblée. Par contre, lorsque $\mathcal{G}$ et $\mathcal{C}$ sont disjointes, la marque portée par la $\mathcal{C}$-structure substituée au bourgeon peut être oubliée, puisque ce marquage est le seul possible.

Exemple 20. Le carré $(\mathcal{F}')^2$ est l'espèce des $\mathcal{F}$-assemblées dont l'un des membres est une $\mathcal{F}'$-structure. Plus généralement, une structure de l'espèce $L(\mathcal{F}')$ est une arbre construit par itération de ce procédé. Ce phénomène est illustré par les figures 5 et 6.

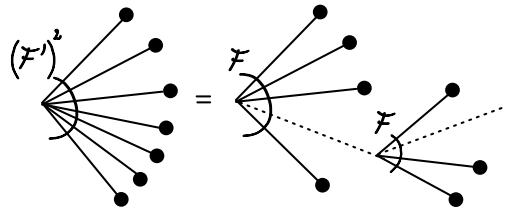


FIG. 5 – Exemple de $(\mathcal{F}')^2$-structure.

[30] Terminologie empruntée à [Lab85]

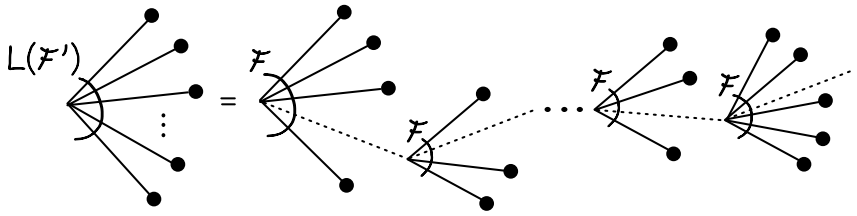


FIG. 6 – Exemple de $L(\mathcal{F}')$-structure.

Exemple 21. Quelles que soient les espèces $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{C}$ telles que $\mathcal{G} \cap \mathcal{C} = \varnothing$, on a l'inclusion suivante :

$$\mathcal{F}(\mathcal{G} + \mathcal{C}) \supset \mathcal{F}(\mathcal{G}) + \mathcal{F}'(\mathcal{G}) \times \mathcal{C}. \tag{1}$$

L'interprétation est la suivante : les structures à droite sont soit de l'espèce $\mathcal{F}(\mathcal{G})$, *i.e.*, des $\mathcal{F}$-assemblées de $\mathcal{G}$-structures uniquement ; soit de l'espèce $\mathcal{F}'(\mathcal{G}) \times \mathcal{C}$, c'est-à-dire des $\mathcal{F}$-assemblées de $\mathcal{G}$-structures dont l'une est remplacée par une $\mathcal{C}$-structure. Ce sont clairement des espèces disjointes, toutes deux contenues dans $\mathcal{F}(\mathcal{G}+\mathcal{C})$. Ce sont les premiers termes du développement de Taylor donné par Labelle dans [Lab90].

On peut étendre l'opération de dérivation aux espèces multi-sortes. La dérivée *partielle* de l'espèce $\mathcal{F}(\mathcal{Z}_1, \ldots, \mathcal{Z}_k)$ sur $\mathbf{U} = (U_1, \ldots, U_k)$ par rapport à $\mathcal{Z}_i$ est alors définie par :

$$\frac{\partial \mathcal{F}}{\partial \mathcal{Z}_i}(\mathcal{Z}_1, \ldots, \mathcal{Z}_k)[\mathbf{U}] = \mathcal{F}[U_1, \ldots, U_i^+, \ldots, U_k].$$

La *matrice jacobienne* d'une famille d'espèces $\boldsymbol{\mathcal{F}}$ par rapport au vecteur $\boldsymbol{\mathcal{Z}}$, notée $\boldsymbol{\partial \mathcal{F}/\partial \mathcal{Z}}$, est la matrice dont les entrées sont les dérivées partielles $\partial \mathcal{F}_i(\boldsymbol{\mathcal{Z}})/\partial \mathcal{Z}_i$.

Exemple 22. Soit une famille d'espèces définie par $\boldsymbol{\mathcal{F}}(\mathcal{Z}_1, \mathcal{Z}_2)$, avec

$$\boldsymbol{\mathcal{F}}(\mathcal{Z}_1, \mathcal{Z}_2) = \begin{pmatrix} \mathcal{Z}_1^2 \times L(\mathcal{Z}_2) \\ E(\mathcal{Z}_2) \end{pmatrix}.$$

Dériver une L-structure consiste à effacer un de ses éléments. Cela revient à la "couper" en deux. La structure obtenue est donc un couple d'ordres linéaires. De même, dériver un produit consiste à effacer l'un des deux membres du produit. Enfin, en effaçant un élément dans un ensemble, la structure obtenue reste un ensemble. On a donc :

$$L'(\mathcal{Z}) = L(\mathcal{Z})^2, \quad (\mathcal{Z}^2)' = 2\mathcal{Z}, \quad E'(\mathcal{Z}) = E(\mathcal{Z}).$$

La matrice jacobienne de $\boldsymbol{\mathcal{F}}$ par rapport à $\boldsymbol{\mathcal{Z}} = (\mathcal{Z}_1, \mathcal{Z}_2)$ est donc :

$$\frac{\boldsymbol{\partial \mathcal{F}}}{\boldsymbol{\partial \mathcal{Z}}} = \begin{pmatrix} 2\mathcal{Z}_1 L(\mathcal{Z}_2) & \mathcal{Z}_1^2 L(\mathcal{Z}_2)^2 \\ 0 & E(\mathcal{Z}_2) \end{pmatrix}.$$

2 Espèces implicites et itération combinatoire

Les arbres binaires se définissent naturellement de façon récursive : un arbre binaire est soit une feuille, soit le produit d'un nœud et de deux sous-arbres binaires. Cette définition de l'espèce $\mathcal{B}$ des arbres binaires se traduit par l'*équation fonctionnelle* suivante :

$$\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B}^2.$$

FIG. 7 – $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ (à gauche). Arborescence $\boldsymbol{\mathcal{H}}$-enrichie (à droite).

Plus généralement, on peut décrire de nombreuses familles d'espèces $\boldsymbol{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_m)$ au moyen de *systèmes d'équations fonctionnelles* de la forme[31] :

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}) \quad \equiv \quad \begin{cases} \mathcal{Y}_1 = \mathcal{H}_1(\boldsymbol{\mathcal{Z}}, \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_m), \\ \mathcal{Y}_2 = \mathcal{H}_2(\boldsymbol{\mathcal{Z}}, \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_m), \\ \qquad \vdots \\ \mathcal{Y}_m = \mathcal{H}_m(\boldsymbol{\mathcal{Z}}, \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_m), \end{cases} \tag{2}$$

où $\boldsymbol{\mathcal{Z}} = (\mathcal{Z}_1, \ldots, \mathcal{Z}_s)$ et chaque $\mathcal{H}_i$ est une espèce à $s + m$ sortes. La donnée d'un tel système suggère une description récursive des $\boldsymbol{\mathcal{Y}}$-structures : une $\boldsymbol{\mathcal{Y}}$-structure est une $\boldsymbol{\mathcal{H}}$-assemblée de singletons et de $\boldsymbol{\mathcal{Y}}$-structures. Par itération sur chaque membre de cette $\boldsymbol{\mathcal{H}}$-assemblée, on obtient une structure arborescente enracinée que l'on appelle *arborescence $\boldsymbol{\mathcal{H}}$-enrichie* (ou encore *$\boldsymbol{\mathcal{H}}$-arborescence*). Cette idée est illustrée par la figure 7. La *hauteur* d'une arborescence $\boldsymbol{\mathcal{H}}$-enrichie est le nombre maximal de $\boldsymbol{\mathcal{H}}$-assemblées sur une branche partant de la racine. Autrement dit, si γ est une $\boldsymbol{\mathcal{H}}$-arborescence, alors sa hauteur $h(\gamma)$ est 0 si γ est un singleton et $\max_i(h(\gamma_i)) + 1$, où les γ_i sont les membres de la $\boldsymbol{\mathcal{H}}$-assemblée à la racine de γ, sinon. L'arborescence de la figure 7, par exemple, a une hauteur égale à 3.

Pour qu'un tel système décrive effectivement une famille d'espèces de structures combinatoires, il est essentiel de donner des conditions précises assurant l'existence et l'unicité d'une famille d'espèces solution. Joyal [Joy81] nous fournit un théorème analogue au théorème des fonctions implicites classique en analyse, mais dans le cadre des espèces de structures.

Théorème 5 (Espèces implicites [Joy81])**.** *Si $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$ est un système implicite d'équations combinatoires satisfaisant les conditions :*

1. *pour tout i tel que $1 \leq i \leq m$, $\mathcal{H}_i(0, 0, \ldots, 0) = 0$,* i.e.*,*

$$\boldsymbol{\mathcal{H}}(\mathbf{0}, \mathbf{0}) = \mathbf{0}, \tag{3}$$

2. *la matrice jacobienne :*

$$\frac{\partial \boldsymbol{\mathcal{H}}}{\partial \boldsymbol{\mathcal{Y}}}(\mathbf{0}, \mathbf{0}) \text{ est nilpotente,} \tag{4}$$

alors il admet une solution $\boldsymbol{\mathcal{Y}}$ unique, à isomorphisme d'espèces près.

Comme en algèbre linéaire classique, une matrice d'espèces de structures $\boldsymbol{\mathcal{M}}$ est nilpotente si l'une de ses puissances est la matrice $\mathbf{0}$. On peut montrer que l'indice de nilpotence (la

[31] Cette façon de définir les espèces est l'analogue des spécifications combinatoires que nous avons étudiées dans la première partie de ce mémoire (voir définition 1, page 21).

puissance p minimale telle que $\boldsymbol{\mathcal{M}}^p = \mathbf{0}$) est borné par la dimension de la matrice. Nous verrons par la suite (lemme 8) que la nilpotence de la matrice jacobienne garantit qu'un tel système ne définit pas une infinité de structures portées par un même ensemble sous-jacent.

2.1 Suites d'espèces

Une preuve du théorème des espèces implicites peut être obtenue par construction d'une suite d'espèce qui converge vers la solution du système $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$. Ce type de suite joue un rôle prépondérant dans les traitements combinatoires que nous présenterons au prochain chapitre. Dans la présente section, nous donnons la preuve du théorème des espèces implicites de Joyal en isolant les propriétés qui nous seront utiles ensuite.

2.1.1 Contact et convergence

On utilisera la notation $\mathcal{F}_{\leq k}$ pour désigner l'espèce $\mathcal{F}$ réduite aux structures portées par des ensembles de cardinalité au plus k.

Définition 22. *Soient $\mathcal{F}$ et $\mathcal{G}$ deux espèces de structures et $k \geq 0$ un entier. On dit que $\mathcal{F}$ et $\mathcal{G}$ ont un* contact *d'ordre k, que l'on note*

$$\mathcal{F} =_k \mathcal{G},$$

s'il existe un isomorphisme d'espèces entre $\mathcal{F}_{\leq k}$ et $\mathcal{G}_{\leq k}$.

Deux vecteurs d'espèces $\boldsymbol{\mathcal{F}} = (\mathcal{F}_1, \dots \mathcal{F}_m)$ et $\boldsymbol{\mathcal{G}} = (\mathcal{G}_1, \dots \mathcal{G}_m)$ ont un contact d'ordre k si toutes les espèces qui les composent ont un contact d'ordre k :

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{G}} \quad \Leftrightarrow \quad \mathcal{F}_i =_k \mathcal{G}_i, \ \ \forall\, i = 1, \dots, m.$$

Définition 23. *Une suite d'espèces $(\mathcal{Y}^{[n]})_{n\in\mathbb{N}}$* converge *vers une espèce $\mathcal{Y}$, ce que l'on note :*

$$\lim_{n\to\infty} \mathcal{Y}^{[n]} = \mathcal{Y},$$

si pour tout $k \geq 0$, il existe $N \geq 0$ tel que, pour tout $n \geq N$, $\mathcal{Y}^{[n]} =_k \mathcal{Y}$.

Par exemple, pour toute espèce $\mathcal{Y}$, $\lim\limits_{n\to\infty} \mathcal{Y}_{\leq n} = \mathcal{Y}$.

Définition 24. *Une suite d'espèces $(\mathcal{Y}^{[n]})_{n\in\mathbb{N}}$ est* croissante *si, pour tout $n \geq 0$,*

$$\mathcal{Y}^{[n]} \subset \mathcal{Y}^{[n+1]}.$$

Comme le contact, la convergence et la croissance pour des vecteurs d'espèces sont définies composante par composante.

Lemme 5. *Soit $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ une suite croissante de familles d'espèces. Si elle possède une suite extraite $(\boldsymbol{\mathcal{Y}}^{[\varphi(n)]})$, avec $\varphi : \mathbb{N} \to \mathbb{N}$ strictement croissante, qui converge vers une limite $\boldsymbol{\mathcal{Y}}$ alors $(\boldsymbol{\mathcal{Y}}^{[n]})$ converge également vers $\boldsymbol{\mathcal{Y}}$.*

Démonstration. Par définition de la convergence d'une suite, on a

$$\forall\, k \geq 0, \quad \exists\, N \geq 0, \quad \forall\, n \geq 0, \quad \varphi(n) \geq N \;\Rightarrow\; \boldsymbol{\mathcal{Y}}^{[\varphi(n)]} =_k \boldsymbol{\mathcal{Y}}. \tag{5}$$

De plus, pour tout m tel que $m \geq \varphi(N)$, il existe $M \geq N$ tel que $\varphi(N) \leq m \leq \varphi(M)$; par croissance de $(\boldsymbol{\mathcal{Y}}^{[\varphi(n)]})$, on a :

$$\boldsymbol{\mathcal{Y}}^{[\varphi(N)]} \subset \boldsymbol{\mathcal{Y}}^{[m]} \subset \boldsymbol{\mathcal{Y}}^{[\varphi(M)]}.$$

En ne considérant que les structures portées par des cardinalités inférieures à k, on a donc :

$$\boldsymbol{\mathcal{Y}}^{[\varphi(N)]}_{\leq k} \subset \boldsymbol{\mathcal{Y}}^{[m]}_{\leq k} \subset \boldsymbol{\mathcal{Y}}^{[\varphi(M)]}_{\leq k}.$$

Or, d'après (5), on a $\boldsymbol{\mathcal{Y}}^{[\varphi(N)]}_{\leq k} = \boldsymbol{\mathcal{Y}}_{\leq k}$ et $\boldsymbol{\mathcal{Y}}^{[\varphi(M)]}_{\leq k} = \boldsymbol{\mathcal{Y}}{\leq k}$; donc on a également $\boldsymbol{\mathcal{Y}}^{[m]}_{\leq k} = \boldsymbol{\mathcal{Y}}_{\leq k}$ et finalement :

$$\boldsymbol{\mathcal{Y}}^{[m]} =_k \boldsymbol{\mathcal{Y}}. \qquad \square$$

La proposition suivante donne des conditions suffisantes pour la convergence d'une suite d'espèces.

Proposition 12. *Soit $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ une suite de familles d'espèces. Soit $(u_n)_{n\in\mathbb{N}}$ une suite d'entiers positifs telle que $\boldsymbol{\mathcal{Y}}^{[n]} =_{u_n} \boldsymbol{\mathcal{Y}}^{[n+1]}$. Si $\lim_{n\to\infty} u_n = \infty$, alors il existe une espèce $\boldsymbol{\mathcal{Y}}$ vers laquelle la suite $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ converge. Cette limite est unique, à isomorphisme près.*

Démonstration. Dans un premier temps, nous définissons $\boldsymbol{\mathcal{Y}}$ en donnant ses structures pour tous les ensembles finis et pour toutes les bijections entre des ensembles finis ; on a alors $\lim \boldsymbol{\mathcal{Y}}^{[n]} = \boldsymbol{\mathcal{Y}}$, par définition de la limite.

Soit k un entier positif. La condition sur (u_n) implique qu'il existe N tel que, pour tout $n \geq N$, on a $u_n \geq k$. Par conséquent, pour ces valeurs de n, $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n-1]} =_k \cdots =_k \boldsymbol{\mathcal{Y}}^{[N]}$, et donc, pour tout $n \geq N$, $\boldsymbol{\mathcal{Y}}^{[n]}$ et $\boldsymbol{\mathcal{Y}}^{[N]}$ coïncident pour tous les ensembles finis de taille k, de même que toutes les bijections entre eux. Nous définissons alors leurs structures communes comme celles de $\boldsymbol{\mathcal{Y}}$. Les propriétés de $\boldsymbol{\mathcal{Y}}[\tau \circ \sigma]$ et $\boldsymbol{\mathcal{Y}}[\mathrm{Id}_U]$ découlent des propriétés similaires pour $\boldsymbol{\mathcal{Y}}^{[N]}$.

L'existence de limites isomorphes est contenue dans la définition d'une limite : une famille d'espèces $\boldsymbol{\mathcal{F}}$ est une autre limite de $(\boldsymbol{\mathcal{Y}}^{[n]})$ si et seulement si $\boldsymbol{\mathcal{Y}} =_k \boldsymbol{\mathcal{F}}$ pour tout $k \geq 0$; ceci implique l'existence d'un isomorphisme d'espèces entre $\boldsymbol{\mathcal{Y}}_{\leq p}$ et $\boldsymbol{\mathcal{F}}_{\leq p}$ pour tout p, *i.e.*, d'un isomorphisme entre $\boldsymbol{\mathcal{Y}}$ et $\boldsymbol{\mathcal{F}}$. $\square$

2.1.2 Preuve du théorème des espèces implicites

La preuve du théorème des espèces implicites consiste à définir une suite d'espèces qui converge vers la solution du système $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$.

Lemme 6. *Soit $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ une suite de familles d'espèces convergeant vers $\boldsymbol{\mathcal{Y}}$. Si $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]})$ converge aussi vers $\boldsymbol{\mathcal{Y}}$, alors la famille d'espèces $\boldsymbol{\mathcal{Y}}$ est solution de $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$.*

Démonstration. Pour montrer que $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$, il suffit de prouver que les deux parties de cette équation coïncident sur les ensembles finis et leurs bijections, ce qui est une conséquence directe de leurs convergences. $\square$

Une *itération combinatoire* est un processus qui construit itérativement une espèce ou une famille d'espèces par substitutions successives. En général, une telle itération est définie par son point de départ que nous noterons $\boldsymbol{\mathcal{Y}}^{[0]}$ et une règle du type $\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]})$ qui permet,

à l'étape $n+1$, de construire une famille d'espèces $\boldsymbol{\mathcal{Y}}^{[n+1]}$ par composition de $\boldsymbol{\mathcal{Y}}^{[n]}$ par $\boldsymbol{\mathcal{F}}$. La preuve du théorème 5 repose sur la convergence de l'itération de point fixe définie par :

$$\boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0} \quad \text{et} \quad \boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) \quad n \geq 0. \tag{6}$$

Les deux lemmes suivants nous assurent que cette suite $(\boldsymbol{\mathcal{Y}}^{[n]})$ remplit les conditions de la proposition 12.

Lemme 7. *La suite de familles d'espèces $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ définie par* (6) *est une suite croissante.*

Démonstration. On utilise une récurrence sur n. Pour $n = 0$, cela vient de la définition de la famille d'espèce $\mathbf{0} = (0, \ldots, 0)$. Pour tout $n \geq 0$, l'inclusion $\boldsymbol{\mathcal{Y}}^{[n]} \subset \boldsymbol{\mathcal{Y}}^{[n+1]}$ est préservée par $\boldsymbol{\mathcal{H}}$-composition :

$$\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) \subset \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]}).$$

Par définition de la suite $(\boldsymbol{\mathcal{Y}}^{[n]})$, c'est équivalent à $\boldsymbol{\mathcal{Y}}^{[n+1]} \subset \boldsymbol{\mathcal{Y}}^{[n+2]}$. □

Pour le lemme 8, nous aurons besoin du corollaire suivant.

Corollaire 1. *Soit la suite $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ définie par* (6)*. Pour tout $n \geq 0$, s'il existe $k \geq 0$ tel que $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$ alors on a $\boldsymbol{\mathcal{Y}}^{[n+1]} =_k \boldsymbol{\mathcal{Y}}^{[n+2]}$.*

Démonstration. Il suffit de remarquer qu'une $\boldsymbol{\mathcal{Y}}^{[n+2]}$-structure α portée par une cardinalité $\leq k$ est, par définition, une $\boldsymbol{\mathcal{H}}$-assemblée de $\boldsymbol{\mathcal{Y}}^{[n+1]}$-structures portées par des cardinalités $\leq k$, c'est à dire des $\boldsymbol{\mathcal{Y}}^{[n]}$-structures. Par conséquent, α appartient à $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) = \boldsymbol{\mathcal{Y}}^{[n+1]}$. En itérant ce raisonnement, on obtient :

$$\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]} =_k \boldsymbol{\mathcal{Y}}^{[n+2]} =_k \ldots$$ □

Lemme 8. *Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$ un système où $\boldsymbol{\mathcal{H}}$ est une famille d'espèces fixée et $\boldsymbol{\mathcal{H}}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$. Soit $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ une suite d'espèces définie par :*

$$\boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0} \quad \textit{et} \quad \boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) \quad n \geq 0. \tag{7}$$

La matrice jacobienne $\frac{\partial \boldsymbol{\mathcal{H}}}{\partial \boldsymbol{\mathcal{Y}}}(\mathbf{0}, \mathbf{0})$ est nilpotente si et seulement si, pour tout $n \geq 0$ et tout $k > 0$, le contact $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$ implique qu'il existe $p > 0$ tel que $\boldsymbol{\mathcal{Y}}^{[n+p]} =_{k+1} \boldsymbol{\mathcal{Y}}^{[n+p+1]}$.

Démonstration. Nous montrons d'abord que la condition sur la matrice jacobienne implique la propriété sur $(\boldsymbol{\mathcal{Y}}^{[n]})$ et ensuite la réciproque.

1. Supposons que $\boldsymbol{\partial \mathcal{H}}/\boldsymbol{\partial \mathcal{Y}}(\mathbf{0}, \mathbf{0})$ est nilpotente d'indice $p \geq 1$. Nous allons montrer que :

$$\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]} \quad \Rightarrow \quad \boldsymbol{\mathcal{Y}}^{[n+p]} =_{k+1} \boldsymbol{\mathcal{Y}}^{[n+p+1]}.$$

Intuitivement, l'idée est que si une $\boldsymbol{\mathcal{H}}$-arborescence appartenant à l'espèce $\boldsymbol{\mathcal{Y}}^{[n+p+1]} - \boldsymbol{\mathcal{Y}}^{[n+p]}$ était portée par une cardinalité $\leq k+1$, alors l'une de ses branches contiendrait une structure appartenant à $(\boldsymbol{\partial \mathcal{H}}/\boldsymbol{\partial \mathcal{Y}}(\mathbf{0}, \mathbf{0}))^p = \mathbf{0}$.
Nous commençons par montrer que pour $i \geq 1$, toute structure γ appartenant à $\boldsymbol{\mathcal{Y}}^{[n+i+1]} - \boldsymbol{\mathcal{Y}}^{[n+i]}$ et portée par une cardinalité $\leq k+1$ se réécrit comme une structure de l'espèce

$$\boldsymbol{\partial \mathcal{H}}/\boldsymbol{\partial \mathcal{Y}}(\mathbf{0}, \mathbf{0})(\boldsymbol{\mathcal{Y}}^{[n+i]} - \boldsymbol{\mathcal{Y}}^{[n+i-1]}).$$

Par définition, la structure γ est une $\boldsymbol{\mathcal{H}}$-assemblée de $\boldsymbol{\mathcal{Y}}^{[n+i]}$-structures. Au moins l'une de ces structures, notée β, appartient à l'espèce $\boldsymbol{\mathcal{Y}}^{[n+i]} - \boldsymbol{\mathcal{Y}}^{[n+i-1]}$ sinon γ serait une $\boldsymbol{\mathcal{H}}$-assemblée

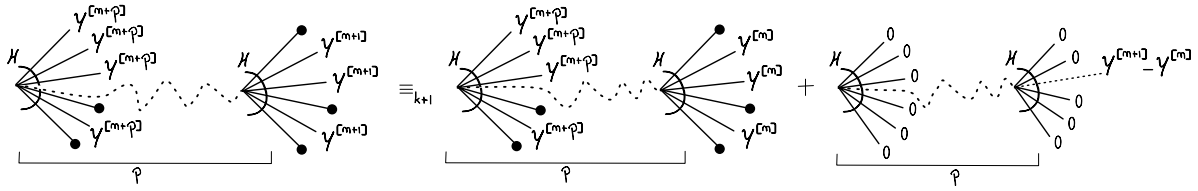


FIG. 8 – Preuve du lemme 8.

de $\boldsymbol{\mathcal{Y}}^{[n+i-1]}$-structures, c'est-à-dire une $\boldsymbol{\mathcal{Y}}^{[n+i]}$-structure. D'après le corollaire 1, un contact d'ordre k au rang n est conservé aux rangs suivants ; la structure β est donc nécessairement portée par une cardinalité $> k$, en l'occurrence par la cardinalité $k+1$ et toutes les autres structures qui composent γ sont portées par la cardinalité 0. Or, comme $\boldsymbol{\mathcal{H}}(\mathbf{0},\mathbf{0}) = \mathbf{0}$, il n'y a pas de $\boldsymbol{\mathcal{H}}$-arborescences portées par la cardinalité 0. Par conséquent, γ appartient bien à l'espèce $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0},\mathbf{0})(\boldsymbol{\mathcal{Y}}^{[n+i]} - \boldsymbol{\mathcal{Y}}^{[n+i-1]})$.

Par itération du résultat précédent, on déduit qu'une structure portée par une cardinalité $\leq k+1$ et appartenant à l'espèce $\boldsymbol{\mathcal{Y}}^{[n+p+1]} - \boldsymbol{\mathcal{Y}}^{[n+p]}$, peut aussi s'écrire comme une structure de l'espèce $(\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0},\mathbf{0}))^p(\boldsymbol{\mathcal{Y}}^{[n+1]} - \boldsymbol{\mathcal{Y}}^{[n]}) = \mathbf{0}$. Autrement dit :

$$\boldsymbol{\mathcal{Y}}^{[n+p+1]} - \boldsymbol{\mathcal{Y}}^{[n+p]} =_{k+1} \mathbf{0}.$$

La figure 8 donne une illustration schématique de cette preuve.

2. Réciproquement, supposons que la matrice jacobienne $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0},\mathbf{0})$ n'est pas nilpotente. Pour tout $q \geq 0$, il existe donc une structure appartenant à $(\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0},\mathbf{0}))^q$ et portée par la cardinalité 0. Soient $n \geq 0$ et $k \geq 1$ tels que $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$ et γ une $\boldsymbol{\mathcal{Y}}^{[n]}$-structure portée par un ensemble de cardinalité d telle que $0 < d \leq k$. Pour tout $q \in \mathbb{N}$, on peut construire une structure $\alpha_q = \beta \cdot \gamma$, portée par la cardinalité d, telle que β appartient à $(\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0},\mathbf{0}))^q$. Ainsi, quel que soit $q > 0$, α_q est une $\boldsymbol{\mathcal{Y}}^{[n+q]}$-structure portée par la cardinalité d mais qui n'est pas une $\boldsymbol{\mathcal{Y}}^{[n+q-1]}$-structure, ce qui rend impossible l'hypothèse sur $(\boldsymbol{\mathcal{Y}}^{[n]})$. □

> **Exemple 23.** L'équation $\mathcal{Y} = \mathcal{H}(\mathcal{Z},\mathcal{Y})$, avec $\mathcal{H}(\mathcal{Z},\mathcal{Y}) = \mathcal{Z} + \mathcal{Z}\mathcal{Y}$, définit les ordres linéaires ayant au moins un élément. Dans ce cas, $\partial\mathcal{H}/\partial\mathcal{Y}(0,0) = 0$, et la séquence définie par l'équation (7) converge bien vers l'espèce des ordres linéaires. En revanche, si $\mathcal{H}(\mathcal{Z},\mathcal{Y}) = \mathcal{Z} + \mathcal{Y}$, alors $\partial\mathcal{H}/\partial\mathcal{Y}(0,0) = 1$ n'est pas nilpotente. Cela se traduit par le fait que cette équation définit une infinité de structures portées par la cardinalité 1. Enfin, bien qu'il semble absurde d'essayer de résoudre l'équation $\mathcal{Y} = \mathcal{Z}\mathcal{Y}$ par itération, cette dernière rentre bien dans le cadre du théorème de espèces implicites, mais définit simplement l'espèce 0.

D'après le lemme précédent et la proposition 12, si p est l'indice de nilpotence de la matrice $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0},\mathbf{0})$, alors la suite $(\boldsymbol{\mathcal{Y}}^{[pn]})_{n\in\mathbb{N}}$ extraite de $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$, est convergente. D'après le lemme 5, la suite $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ est également convergente. La preuve du théorème des espèces implicites est alors complétée par l'invocation du lemme 6.

2.2 Caractérisation de l'itération de point fixe combinatoire

Nous venons de voir que l'itération "simple", qui consiste, à chaque étape, à composer le résultat de l'étape précédente par $\boldsymbol{\mathcal{H}}$, converge vers la solution du système $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}},\boldsymbol{\mathcal{Y}})$. Nous avons là un algorithme qui nous permet de déterminer la solution d'un système d'équations

combinatoires implicites. Par ailleurs, on peut caractériser les espèces successives engendrées par cette itération : les $\boldsymbol{\mathcal{H}}$-arborescences qui les composent ont une hauteur bornée.

Propriété 1. *Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$ un système tel que $\boldsymbol{\mathcal{H}}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$ et $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0}, \mathbf{0})$ est nilpotente. L'itération combinatoire*

$$\boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0} \quad \textit{et} \quad \boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) \quad n \geq 0$$

converge vers $\boldsymbol{\mathcal{Y}}$, solution de $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$. De plus, pour $i \geq 0$, l'espèce $\boldsymbol{\mathcal{Y}}^{[i]}$ est constituée de toutes les $\boldsymbol{\mathcal{H}}$-arborescences de hauteur h inférieure ou égale à i :

$$\forall \gamma, \quad \gamma \in \boldsymbol{\mathcal{Y}}^{[i]} \quad \Leftrightarrow \quad h(\gamma) \leq i.$$

Démonstration. La convergence est une conséquence directe de la preuve du théorème des espèces implicites. La caractérisation de l'espèce $\boldsymbol{\mathcal{Y}}^{[i]}$ par la hauteur de ses structures se vérifie par récurrence : pour $i = 0$, cela vient du fait que $\boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0}$. Pour $i > 0$, par définition de l'itération, une $\boldsymbol{\mathcal{Y}}^{[i]}$-structure est une $\boldsymbol{\mathcal{H}}$-assemblée dont les membres sont des singletons et/ou, par hypothèse de récurrence, des $\boldsymbol{\mathcal{H}}$-arborescences de hauteur inférieure ou égale à $i-1$; toute $\boldsymbol{\mathcal{Y}}^{[i]}$-structure est donc une $\boldsymbol{\mathcal{H}}$-arborescence de hauteur au plus i. Réciproquement, si γ est une $\boldsymbol{\mathcal{H}}$-arborescence de hauteur inférieure ou égale à i, pour $i \geq 0$, alors γ une $\boldsymbol{\mathcal{H}}$-assemblée dont les membres ont au plus une hauteur $i-1$. Ce sont donc des $\boldsymbol{\mathcal{Y}}^{[i-1]}$-structures, et par conséquent, γ est une $\boldsymbol{\mathcal{Y}}^{[i]}$-structure. □

Nous illustrons le déroulement de l'itération combinatoire sur deux exemples classiques. Nous commençons par des arbres planaires qui peuvent être définis à l'aide d'une seule équation. Nous traitons ensuite la famille des graphes série-parallèle.

2.2.1 Les arbres généraux planaires

Les graphes généraux planaires sont définis par l'équation suivante :

$$\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y}), \qquad \text{avec } \mathcal{H}(\mathcal{Z}, \mathcal{Y}) = \mathcal{Z} \times L(\mathcal{Y}).$$

Dans un premier temps, nous nous assurons que les conditions du corollaire 1 sont remplies. L'équation $\mathcal{H}(0,0) = 0$ se vérifie simplement. Quant à la seconde condition, on a :

$$\partial\mathcal{H}/\partial\mathcal{Y} = 0 \times L(\mathcal{Y}) + \mathcal{Z} \times L(\mathcal{Y})^2 \quad \Rightarrow \quad \partial\mathcal{H}/\partial\mathcal{Y}(0,0) = 0$$

L'itération combinatoire correspondante est :

$$\mathcal{Y}^{[n+1]} = \mathcal{Z} \times L(\mathcal{Y}^{[n]}). \tag{8}$$

Les six premières espèces $\mathcal{Y}^{[0]}, \ldots, \mathcal{Y}^{[5]}$ obtenues par cette itération sont représentées par la figure 9. Pour simplifier la lecture de ce dessin, nous n'avons représenté que les types d'isomorphisme des arbres. Les sous-arbres dessinés en bleu correspondent aux arbres hérités de l'itération précédente. Les rectangles qui entourent certaines structures représentent le contact avec la solution $\mathcal{Y}$, c'est-à-dire, l'espèce des arbres généraux planaires. Par exemple, l'espèce $\mathcal{Y}^{[4]}$ contient tous les arbres portés par des cardinalités inférieures ou égales à 4, *i.e.*, $\mathcal{Y}^{[4]} =_4 \mathcal{Y}$. L'espèce $\mathcal{Y}^{[i]}$ obtenue à la i-ième itération, pour tout $i \geq 0$, est l'espèce des arbres généraux de hauteur au plus i.

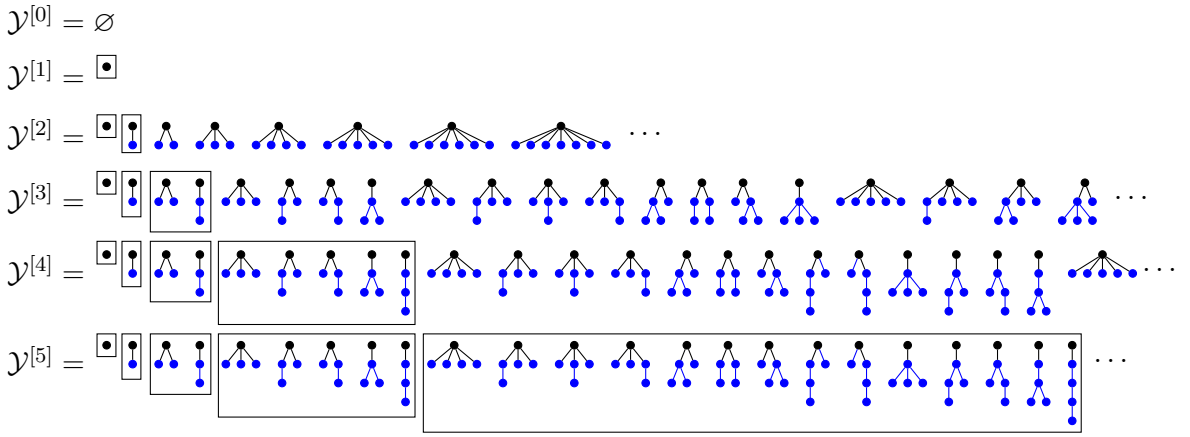


FIG. 9 – Premières itérations pour les arbres généraux planaires.

2.2.2 Les graphes série-parallèle

Les graphes série-parallèle sont définis par $(\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3) = \boldsymbol{\mathcal{SP}}(\mathcal{Z}, \mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3)$, avec

$$\boldsymbol{\mathcal{SP}}(\mathcal{Z}, \mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3) = \begin{pmatrix} \mathcal{Z} + \mathcal{Y}_2 + \mathcal{Y}_3 \\ L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3) \\ E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2) \end{pmatrix},$$

où $L_{\geq 2}$ et $E_{\geq 2}$ sont les espèces des ordres linéaires et des ensembles ayant au moins 2 éléments.

Nous avons bien les hypothèses du corollaire 1. En effet, la condition $\boldsymbol{\mathcal{H}}(0, \mathbf{0}) = \mathbf{0}$ se vérifie simplement. Pour la seconde condition, nous commençons par calculer la matrice jacobienne de $\boldsymbol{\mathcal{H}}$. Dériver une $L_{\geq 2}$-structure consiste à effacer l'un de ses éléments. Cela revient à la "couper" en deux. La structure obtenue est donc un couple d'ordres linéaires porté par une cardinalité globale ≥ 1. De même, en effaçant un élément dans un ensemble ayant au moins deux éléments, on obtient un ensemble avec au moins un élément. On a donc :

$$\partial L_{\geq 2}/\partial \mathcal{Y}(\mathcal{Y}) = L(\mathcal{Y})^2 - 1, \quad \partial E_{\geq k}/\partial \mathcal{Y}(\mathcal{Y}) = E^+(\mathcal{Y}).$$

La matrice jacobienne du système décrivant les graphes série-parallèle est donc :

$$\frac{\boldsymbol{\partial \mathcal{SP}}}{\boldsymbol{\partial \mathcal{Y}}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & L(\mathcal{Z} + \mathcal{Y}_3)^2 - 1 \\ 0 & E^+(\mathcal{Z} + \mathcal{Y}_2) & 0 \end{pmatrix},$$

et la matrice

$$\boldsymbol{\partial \mathcal{SP}}/\boldsymbol{\partial \mathcal{Y}}(0,0,0,0) = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

est nilpotente d'indice 3.
L'itération combinatoire suivante converge vers l'espèce des graphes série-parallèle :

$$\begin{cases} \mathcal{Y}_1^{[n+1]} &= \mathcal{Z} + \mathcal{Y}_2^{[n]} + \mathcal{Y}_3^{[n]} \\ \mathcal{Y}_2^{[n+1]} &= L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3^{[n]}) \\ \mathcal{Y}_3^{[n+1]} &= E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2^{[n]}) \end{cases}. \tag{9}$$

Les trois premières familles d'espèces $\mathcal{Y}^{[0]}$, $\mathcal{Y}^{[1]}$, $\mathcal{Y}^{2]}$ obtenues par cette itération sont représentées par la figure 10. Comme précédemment, nous n'avons représenté que les types d'isomorphisme des graphes. Les sous-graphes dessinés en bleu correspondent aux graphes hérités de l'itération précédente. Les rectangles rouges qui entourent certaines structures représentent le contact avec la solution $\mathcal{Y}$, Par exemple, $\mathcal{Y}^{[2]} =_2 \mathcal{Y}$.

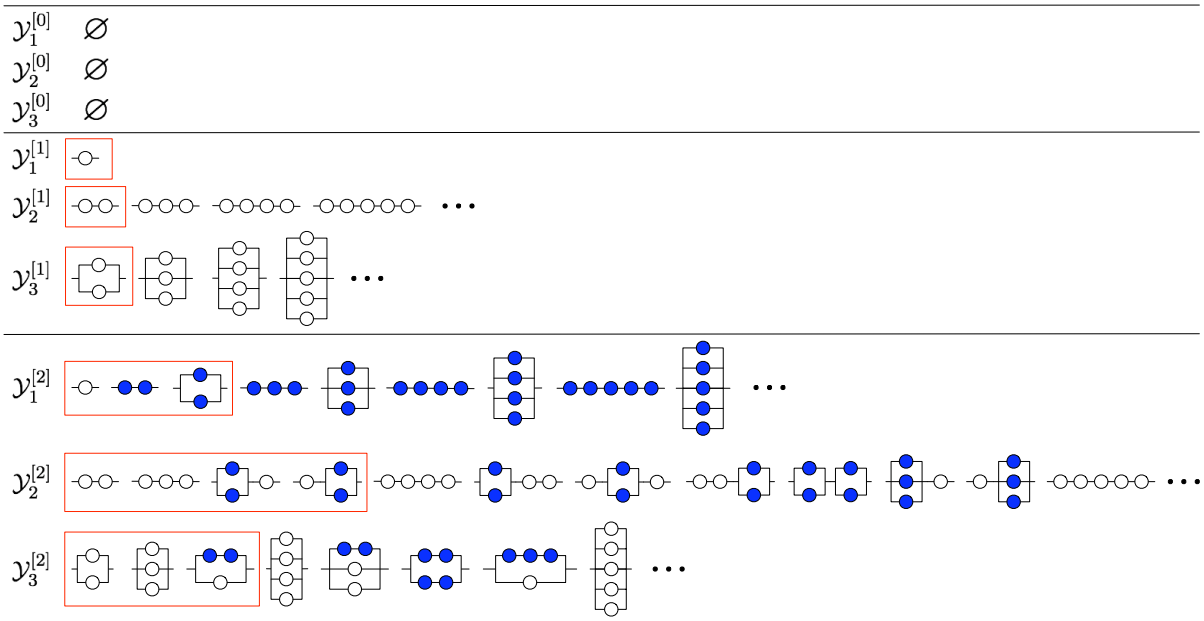


FIG. 10 – Premières itérations pour les graphes série-parallèle.

Chapitre 5

Itérations de Newton combinatoires

Sommaire

L'itération de point fixe présentée au chapitre précédent fournit un algorithme pour construire la solution d'un système d'équations combinatoires implicites. Nous verrons dans les chapitres suivants qu'une telle itération peut se traduire en un algorithme pour calculer les suites de dénombrement des espèces ainsi définies et les valeurs numériques correspondant à l'oracle des générateurs de Boltzmann. Dans ce chapitre, nous montrons comment accélérer la convergence en utilisant une itération de Newton.

L'idée d'une itération de Newton combinatoire pour résoudre une équation sur les espèces vient de Décoste, Labelle et Leroux [DLL82] et est reprise dans [BLL98]. Nous étendons tout d'abord cette idée aux systèmes combinatoires et définissons ensuite une itération optimisée qui converge vers l'espèce solution aussi rapidement que l'itération de Newton classique, mais en construisant moins de structures intermédiaires. Une part importante de ce chapitre est consacrée aux preuves de convergence de ces itérations. Le point essentiel de notre traitement est que les propriétés combinatoires que l'on dégage d'une itération se relèvent ensuite au niveau des séries de dénombrement et de leurs évaluations numériques, comme nous le verrons dans les prochains chapitres.

1 Itération de Newton combinatoire

Comme on peut le voir sur les exemples présentés à la fin du chapitre précédent, la convergence de l'itération simple est typiquement linéaire, c'est-à-dire que le contact entre la solution

recherchée et les espèces engendrées par l'itération augmente de 1 à chaque étape. Dans cette section, nous présentons une deuxième itération, plus rapide que la précédente ; il s'agit d'une itération de Newton combinatoire. Contrairement à l'itération simple, celle-ci a une convergence quadratique, autrement dit, le contact avec l'espèce solution double à chaque itération. L'itération de Newton pour une unique équation est présentée dans [DLL82, BLL98]. Nous présentons une extension de ce résultat aux familles d'espèces définies par des systèmes à plusieurs équations.

1.1 Itération de Newton pour une seule équation

L'itération de Newton combinatoire proposée dans [DLL82, BLL98] pour résoudre l'équation :

$$\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$$

est présentée comme la réponse à un problème d'approximation entre espèces :

> *« À partir d'une espèce $\mathcal{A}$ ayant un contact d'ordre k avec l'espèce $\mathcal{Y}$, construire combinatoirement et de façon canonique une meilleure approximation $\mathcal{A}^\star$ de $\mathcal{Y}$ au sens où, cette fois $\mathcal{A}^\star =_{2k+1} \mathcal{Y}$. »*

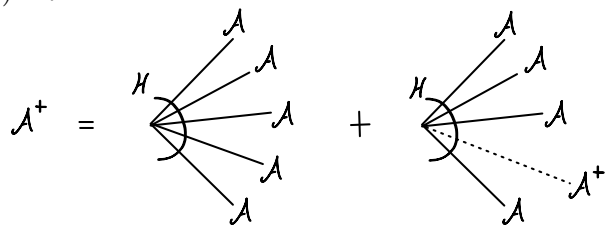
L'idée est de fabriquer l'espèce $\mathcal{A}^\star$ comme la somme de $\mathcal{A}$ et de suffisamment de *nouvelles* $\mathcal{H}$-arborescences pour obtenir le contact souhaité. On appelle $\mathcal{A}^+$ cette nouvelle espèce :

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

Les $\mathcal{A}^+$-structures doivent être portées par des cardinalités supérieures à k. Elles sont naturellement décrites comme des $\mathcal{H}$-assemblées de $\mathcal{Y}$-structures. Ces $\mathcal{Y}$-structures sont :

- soit toutes portées par des cardinalités inférieures à k ; ce sont donc des $\mathcal{A}$-structures ;
- soit des $\mathcal{A}$-structures et une unique $\mathcal{Y}$-structure portée par une cardinalité supérieure à k, c'est-à-dire une $\mathcal{A}^+$-structure (on ne considère pas les autres cas, étant donné qu'avec au moins deux $\mathcal{A}^+$-structures, la cardinalité totale serait supérieure à $2k+1$).

Dans le premier cas, les $\mathcal{A}^+$-structures appartiennent à l'espèce $\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A}$, et dans le second, à l'espèce $\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{A}) \times \mathcal{A}^+$:



On a donc $\mathcal{A}^+ = \partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{A}) \times \mathcal{A}^+ + (\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A})$ dont la solution est :

$$\mathcal{A}^+ = L(\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{A})) \times (\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A}).$$

On obtient ainsi une version purement combinatoire de l'itération de Newton :

$$\mathcal{A}^\star = \mathcal{A} + L(\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{A})) \times (\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A}).$$

Avec nos notations habituelles, cette itération de Newton combinatoire s'écrit :

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + L\left(\frac{\partial\mathcal{H}}{\partial\mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]})\right) \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}), \qquad \mathcal{Y}^{[0]} = 0.$$

Les structures engendrées par cette itération à l'étape $n+1$ sont des $\mathcal{H}$-arborescences de $\mathcal{Y}^{[n]}$-structures. On peut décomposer ces arborescences suivant la branche formée par les arêtes en pointillé correspondant aux bourgeons de la suite des dérivées. La figure 1 donne un exemple de structure engendrée à l'étape $n+1$ de l'itération de Newton.

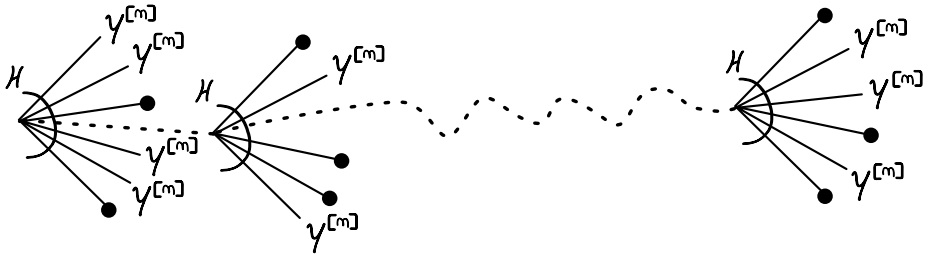


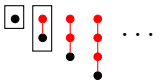
FIG. 1 – Exemple de $\mathcal{Y}^{[n+1]}$-structure obtenue par itération de Newton.

Exemple des arbres généraux Si l'on adapte ce résultat à l'exemple des arbres généraux planaires que nous avons traité à la section 2.2.1, on obtient l'itération :

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + L(\mathcal{Z} \times L(\mathcal{Y}^{[n]})^2) \times (\mathcal{Z} \times L(\mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}), \qquad \mathcal{Y}^{[0]} = 0. \tag{1}$$

La figure 2 représente le résultat des trois premières itérations. Les arbres bleus sont ceux qui proviennent directement de l'itération précédente. Pour les autres arbres, nous avons représenté en rouge les nœuds qui sont issus de $L(\mathcal{Z} \times L(\mathcal{Y}^{[n]})^2)$. Les rectangles noirs représentent les cardinalités pour lesquelles le contact avec la solution est atteint. Il est intéressant de comparer les espèces obtenues avec celles de la figure 9 (page 93). En effet, dès la troisième itération, on a un contact d'ordre 5 avec l'espèce des arbres, alors qu'avec l'itération simple, pour le même contact, il fallait aller jusqu'à la sixième itération.

$\mathcal{Y}^{[0]} = \varnothing$

$\mathcal{Y}^{[1]} =$ 

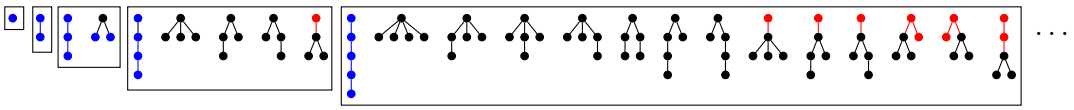
$\mathcal{Y}_{[2]} =$ 

FIG. 2 – Premières étapes de l'itération de Newton pour les arbres généraux planaires.

Caractérisation de l'itération de Newton Comme pour l'itération simple, on peut caractériser les espèces successivement produites par l'itération de Newton : l'espèce $\mathcal{Y}^{[i]}$ obtenue à la i-ème étape de l'itération de Newton est l'ensemble des $\mathcal{H}$-arborescences dont le nombre de Strahler est au plus i. Tout d'abord, on définit un nombre de Strahler pour les arborescences $\mathcal{H}$-enrichies[32]. Soit γ une $\mathcal{H}$-arborescence ; si $\gamma = 0$, alors son nombre de Strahler est 0, sinon,

[32]Il s'agit simplement d'une extension aux arbres généraux de la définition du nombre de Strahler pour les arbres binaires. C'est le nombre de registres nécessaires à l'évaluation d'un arbre d'expression arithmétique.

elle est composée des membres $(\gamma_1, \ldots, \gamma_q)$ et son nombre de Strahler vaut :

$$Sh(\gamma) = \begin{cases} 1 & \text{si } \gamma_1 = \cdots = \gamma_q = \mathcal{Z}, \\ Sh(\gamma_i) + 1 & \text{si } \exists\, i, j, \text{ tels que } i \neq j \text{ et } Sh(\gamma_i) = Sh(\gamma_j) = \max_{1 \leq \ell \leq q}(Sh(\gamma_\ell)), \\ \max_{1 \leq \ell \leq q}(Sh(\gamma_\ell)) & \text{sinon.} \end{cases}$$

Propriété 2. *Soit $(\mathcal{Y}^{[i]})_{i \in \mathbb{N}}$ la suite des espèces obtenues par itération de Newton pour le système combinatoire $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$; pour toute $\mathcal{H}$-arborescence γ, on a :*

$$\gamma \in \mathcal{Y}^{[i]} \;\Leftrightarrow\; Sh(\gamma) \leq i.$$

Démonstration. Par récurrence sur l'indice i des espèces $\mathcal{Y}^{[i]}$ successives. Pour $i = 0$, cela vient du fait que $\mathcal{Y}^{[0]} = 0$. Supposons à présent que la propriété est vérifiée jusqu'à l'étape i.

1. Si $\gamma \in \mathcal{Y}^{[i+1]}$, alors elle peut se décomposer sous la forme d'une séquence $\gamma = \beta_1 \cdots \beta_\ell \cdot \delta$, telle que $\beta_j \in \partial\mathcal{H}/\partial Y(\mathcal{Z}, \mathcal{Y}^{[i]})$ pour tout j tel que $1 \leq j \leq \ell$ et $\delta \in \mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[i]}) - \mathcal{Y}^{[i]}$. Par hypothèse de récurrence, $Sh(\delta) \leq i+1$, donc $\beta_\ell \cdot \delta$ est une $\mathcal{H}$-assemblée dont au plus un membre a un nombre de Strahler $\leq i+1$ et les autres ont un nombre de Strahler $\leq i$; par conséquent, $Sh(\beta_\ell \cdot \delta) \leq i+1$. En itérant ce raisonnement sur les β_j jusqu'à $j = 1$, on a $Sh(\gamma) \leq i + 1$.

2. Si $Sh(\gamma) \leq i+1$, alors la structure γ est une $\mathcal{H}$-assemblée qui est soit de la forme $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[i]})$, si toutes les $\mathcal{H}$-arborescences qui la composent ont un nombre de Strahler $\leq i$; soit de la forme $\partial\mathcal{H}/\partial Y(\mathcal{Z}, \mathcal{Y}^{[i]}) \cdot \delta$, avec $Sh(\delta) = i + 1$. En itérant ce raisonnement jusqu'à épuisement de γ, on obtient $\gamma \in \mathcal{Y}^{[i+1]}$. □

1.2 Cas général

Dans cette section, nous étendons le résultat de [BLL98] à l'ensemble des systèmes d'équations combinatoires. L'itération précédente se traduit en une itération similaire sur des vecteurs d'espèces, à condition de donner un sens à l'espèce $L\left(\frac{\partial\mathcal{H}}{\partial\mathcal{Y}}(\mathcal{Z}, \mathcal{Y})\right)$ dans le cas multi-dimensionnel.

Proposition 13. *Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$ un système tel que $\boldsymbol{\mathcal{H}}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$ et $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\mathbf{0}, \mathbf{0})$ est nilpotente. L'itération combinatoire :*

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{Y}}^{[n]} + \left(\mathbf{1} - \frac{\boldsymbol{\partial\mathcal{H}}}{\boldsymbol{\partial\mathcal{Y}}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]})\right)^{-1} (\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}), \quad \boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0}, \tag{2}$$

converge vers $\boldsymbol{\mathcal{Y}}$, solution de $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$. De plus, cette convergence est quadratique :

$$\forall\, n, k \geq 0, \quad \boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}} \;\Rightarrow\; \boldsymbol{\mathcal{Y}}^{[n+1]} =_{2k+1} \boldsymbol{\mathcal{Y}}.$$

L'interprétation combinatoire du quasi-inverse de la matrice jacobienne est donnée par Labelle dans [Lab85], en termes d'éclosions combinatoires. Pour mieux comprendre quelles sont les structures qui le composent, on peut le réécrire sous la forme :

$$\left(\mathbf{1} - \frac{\boldsymbol{\partial\mathcal{H}}}{\boldsymbol{\partial\mathcal{Y}}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})\right)^{-1} = \sum_{k \geq 0} \left(\frac{\boldsymbol{\partial\mathcal{H}}}{\boldsymbol{\partial\mathcal{Y}}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})\right)^k.$$

C'est l'équivalent multi-sortes de l'espèce des ordres linéaires de dérivées de $\boldsymbol{\mathcal{H}}$-structures. En effet, les structures qui le composent sont des séquences de $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$-structures. Plus précisément, chaque entrée (i, j) de cette matrice est une séquence commençant par une $\partial\mathcal{H}_i/\boldsymbol{\partial\mathcal{Y}}$-structure et se terminant par une $\boldsymbol{\partial\mathcal{H}}/\partial\mathcal{Y}_j$-structure. Les structures qui se trouvent entre ces deux extrémités sont ordonnées de telle façon qu'une $\boldsymbol{\partial\mathcal{H}}/\partial\mathcal{Y}_k$-structure est toujours suivie d'une $\partial\mathcal{H}_k/\boldsymbol{\partial\mathcal{Y}}$-structure, avec $1 \leq k \leq m$. La figure 3 illustre ce phénomène.

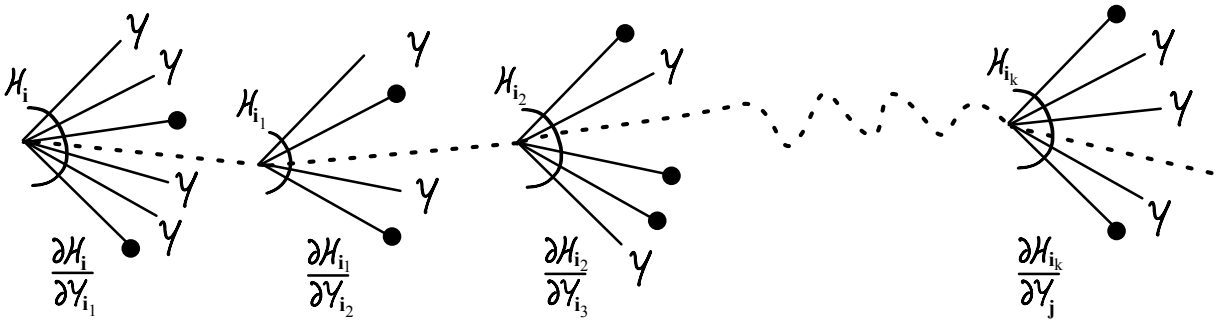


FIG. 3 – Exemple de structure appartenant à l'entrée (i, j) de la matrice $(\mathbf{1} - \partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{Y}))^{-1}$.

1.3 Convergence

Cette section a pour but de prouver la proposition 13. Dans un premier temps, on vérifie que l'itération (2) est bien définie, avant de montrer qu'elle converge bien vers la solution du système combinatoire $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$.

L'itération est croissante L'itération (2) n'a de sens que si la soustraction $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}$ est possible, c'est-à-dire si pour tout $n \geq 0$, l'espèce $\mathcal{Y}^{[n]}$ est une sous-espèce de $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]})$. Dans ce cas l'itération est également croissante, puisque par définition, l'espèce $\mathcal{Y}^{[n]}$ est contenue dans $\mathcal{Y}^{[n+1]}$.

Propriété 3. *L'itération de Newton combinatoire donnée par l'équation* (2) *est bien définie :*

$$\mathcal{Y}^{[n]} \subset \mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}).$$

Démonstration. La preuve de cette inclusion se fait par récurrence. Pour $n = 0$, c'est une conséquence de $\mathcal{Y}^{[0]} = \mathbf{0}$. Supposons maintenant que la propriété est satisfaite pour n. Remarquons que par définition de l'itération, cela implique :

$$\mathcal{Y}^{[n]} \subset \mathcal{Y}^{[n+1]}.$$

Nous utilisons alors les premiers termes du développement de Taylor pour les espèces (voir exemple 21, page 86) pour réécrire $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n+1]})$. La soustraction $\mathcal{Y}^{[n+1]} - \mathcal{Y}^{[n]}$ est justifiée puisque $\mathcal{Y}^{[n]} \subset \mathcal{Y}^{[n+1]}$. Nous avons donc :

$$\begin{aligned}
\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n+1]}) &\supset \mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) + \frac{\partial\mathcal{H}}{\partial\mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]}) \times (\mathcal{Y}^{[n+1]} - \mathcal{Y}^{[n]}), \\
&\supset \mathcal{Y}^{[n]} + (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}) \\
&\qquad + \frac{\partial\mathcal{H}}{\partial\mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]}) \times \sum_{k\geq 0} \left(\frac{\partial\mathcal{H}}{\partial\mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]})\right)^k \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}), \\
&\supset \mathcal{Y}^{[n]} + \sum_{k\geq 0} \left(\frac{\partial\mathcal{H}}{\partial\mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]})\right)^k \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}) = \mathcal{Y}^{[n+1]}.
\end{aligned}$$

À la seconde ligne, nous utilisons l'hypothèse de récurrence pour réécrire $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]})$ et la définition de l'itération pour réécrire $\mathcal{Y}^{[n+1]}$. □

Notons que la démonstration précédente implique que la suite d'espèces définie par l'itération de Newton est croissante. On a donc le corollaire suivant.

Corollaire 2. *Étant donné une espèce $\mathcal{H}$, la suite $(\mathcal{Y}^{[n]})_{n\in\mathbb{N}}$ définie par l'itération de Newton* (2) *est une suite d'espèces croissante : $\mathcal{Y}^{[n]} \subset \mathcal{Y}^{[n+1]}$.*

L'itération n'est pas ambiguë Pour nous assurer que l'itération (2) ne produit pas plusieurs fois les mêmes structures, nous vérifions que toutes les structures de $\mathcal{Y}^{[n+1]}$ sont distinctes.

Propriété 4. *Étant donné une espèce $\mathcal{H}$, l'itération de Newton* (2) *n'est pas ambiguë : pour tout $n \geq 0$ et tout $k \geq 0$:*

$$\mathcal{Y}^{[n]} \cap \left(\frac{\partial \mathcal{H}}{\partial \mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]})\right)^k \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}) = \varnothing$$

Démonstration. Pour cette preuve, on utilise une récurrence sur k. Pour $k = 0$, l'intersection de $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}$ avec l'espèce $\mathcal{Y}^{[n]}$ est effectivement vide, quel que soit n. Si $k \geq 1$, on considère un structure γ quelconque de l'espèce

$$(\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{Y}^{[n]}))^k \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}).$$

On peut décomposer γ comme le produit d'une structure $\beta \in \partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{Y}^{[n]})$ et d'une structure $\delta \in (\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{Y}^{[n]}))^{k-1} \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]})$. Par hypothèse de récurrence, δ n'est pas une $\mathcal{Y}^{[n]}$-structure. Comme γ est une $\mathcal{H}$-assemblée dont l'un des membres est δ, alors γ n'appartient pas à $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]})$. Or, pour tout $n \geq 0$, l'espèce $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]})$ contient $\mathcal{Y}^{[n]}$, donc γ ne peut pas être une $\mathcal{Y}^{[n]}$-structure. □

L'itération est convergente Maintenant que nous sommes assurés de la validité de l'itération de Newton pour les systèmes combinatoires, nous pouvons vérifier sa convergence.

Proposition 14. *Soit $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ un système tel que $\mathcal{H}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$ et $\partial\mathcal{H}/\partial\mathcal{Y}(\mathbf{0}, \mathbf{0})$ est nilpotente. L'itération combinatoire définie par :*

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \left(\mathbf{1} - \frac{\partial \mathcal{H}}{\partial \mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]})\right)^{-1} (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}), \quad \mathcal{Y}^{[0]} = \mathbf{0}$$

converge vers $\mathcal{Y}$, solution de $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$. De plus, cette convergence est quadratique.

Démonstration. Dans un premier temps, on vérifie la convergence (et sa vitesse) de l'itération et ensuite, on montre que la limite est bien la solution recherchée.

Supposons que $\mathcal{Y}^{[n-1]} =_k \mathcal{Y}^{[n]}$ et considérons une structure quelconque γ appartenant à l'espèce $\mathcal{Y}^{[n+1]} - \mathcal{Y}^{[n]}$ et portée par une cardinalité au plus $2k+1$. Par définition, la structure γ appartient à l'espèce

$$(\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z}, \mathcal{Y}^{[n]}))^\ell \times (\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}) \quad \text{avec } \ell \in \mathbb{N}.$$

Parmi les $\mathcal{Y}^{[n]}$-structures qui composent γ, il y en a une et une seule qui est portée par une cardinalité plus grande que k : si elles étaient deux, la structure γ serait portée par une cardinalité supérieure à $2k+1$ et s'il n'y en avait aucune, toutes ces $\mathcal{Y}^{[n]}$-structures seraient en fait des $\mathcal{Y}^{[n-1]}$-structures et γ appartiendrait à l'espèce $\mathcal{Y}^{[n]}$. De plus, cette structure, que l'on

appelle β, appartient à la $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]})$-structure finale, puisque sinon, cette structure finale δ, appartiendrait à $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n-1]}) \subset \boldsymbol{\mathcal{Y}}^{[n]}$. Par définition, la structure β appartient elle-même à l'espèce

$$(\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n-1]}))^{\ell'} \times (\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n-1]}) - \boldsymbol{\mathcal{Y}}^{[n-1]}) \quad \text{avec } \ell' \in \mathbb{N}.$$

Par conséquent δ appartient à $(\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n-1]}))^{\ell'+1}(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n-1]}) - \boldsymbol{\mathcal{Y}}^{[n-1]})$ qui est une sous-espèce de $\boldsymbol{\mathcal{Y}}^{[n]}$; or, c'est impossible. On en déduit qu'il n'existe pas de structure portée par une cardinalité inférieure à $2k+1$ dans $\boldsymbol{\mathcal{Y}}^{[n+1]} - \boldsymbol{\mathcal{Y}}^{[n]}$. Autrement dit, $\boldsymbol{\mathcal{Y}}^{[n]} =_{2k+1} \boldsymbol{\mathcal{Y}}^{[n+1]}$.

La preuve est conclue par invocation de la proposition 12 et du lemme 6 (page 89), en montrant que la limite de la suite d'espèces définie par l'équation (2) est la solution de $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$. Cela découle directement de la réécriture de l'itération sous la forme :

$$\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]} + \frac{\boldsymbol{\partial\mathcal{H}}}{\boldsymbol{\partial\mathcal{Y}}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) \cdot (\boldsymbol{\mathcal{Y}}^{[n+1]} - \boldsymbol{\mathcal{Y}}^{[n]}) = \boldsymbol{\mathcal{Y}}^{[n+1]} - \boldsymbol{\mathcal{Y}}^{[n]}$$

et de l'observation que puisque $\boldsymbol{\mathcal{Y}}^{[n+1]} - \boldsymbol{\mathcal{Y}}^{[n]}$ converge vers $\mathbf{0}$, alors $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}$ aussi. □

1.4 Exemple des graphes série-parallèle

Les graphes série-parallèle sont définis par[33] :

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{SP}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}}) \equiv \begin{cases} \mathcal{Y}_2 = L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3) \\ \mathcal{Y}_3 = E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2) \end{cases}.$$

L'itération de Newton combinatoire associée est la suivante :

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{Y}}^{[n]} + (\mathbf{1} - \boldsymbol{\partial\mathcal{SP}}/\boldsymbol{\partial\mathcal{Y}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}}))^{-1} \begin{pmatrix} L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3^{[n]}) - \mathcal{Y}_2^{[n]} \\ E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - \mathcal{Y}_3^{[n]} \end{pmatrix}, \tag{3}$$

où l'inverse de la matrice $\mathbf{1} - \boldsymbol{\partial\mathcal{SP}}/\boldsymbol{\partial\mathcal{Y}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ se réécrit directement sous la forme

$$\begin{pmatrix} 1 & 1 - L(\mathcal{Z} + \mathcal{Y}_3^{[n]})^2 \\ 1 - E(\mathcal{Z} + \mathcal{Y}_2^{[n]}) & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \mathcal{L}_1 & \left(L(\mathcal{Z} + \mathcal{Y}_3^{[n]})^2 - 1\right)\mathcal{L}_2 \\ \left(E(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - 1\right)\mathcal{L}_1 & \mathcal{L}_2 \end{pmatrix}$$

avec

$$\begin{aligned} \mathcal{L}_1 &= L\left(\left(L(\mathcal{Z} + \mathcal{Y}_3^{[n]})^2 - 1\right)\left(E(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - 1\right)\right), \\ \mathcal{L}_2 &= L\left(\left(E(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - 1\right)\left(L(\mathcal{Z} + \mathcal{Y}_3^{[n]})^2 - 1\right)\right), \end{aligned}$$

ce qui donne l'itération :

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{Y}}^{[n]} + \begin{pmatrix} \mathcal{L}_1\left(L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3^{[n]}) - \mathcal{Y}_2^{[n]}\right) + \left(L(\mathcal{Z} + \mathcal{Y}_3^{[n]})^2 - 1\right)\mathcal{L}_2\left(E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - \mathcal{Y}_3^{[n]}\right) \\ \left(E(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - 1\right)\mathcal{L}_1\left(L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3^{[n]}) - \mathcal{Y}_2^{[n]}\right) + \mathcal{L}_2\left(E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - \mathcal{Y}_3^{[n]}\right) \end{pmatrix}$$

La figure 4 représente le résultat des trois premières itérations. Les graphes bleus sont ceux qui proviennent directement de l'itération précédente. Pour les autres graphes, nous avons

[33] Nous avons supprimé la première équation du système donné au chapitre précédent (page 93), pour alléger l'écriture de l'itération.

représenté en rouge les nœuds qui sont issus de $(\mathbf{1} - \boldsymbol{\partial}\boldsymbol{\mathcal{S}}\boldsymbol{P}/\boldsymbol{\partial}\boldsymbol{\mathcal{Y}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}}))^{-1}$. Les rectangles rouges correspondent aux cardinalités pour lesquelles le contact avec la solution est atteint. Il est intéressant de comparer les espèces obtenues avec celles de la figure 10 du chapitre 4 (page 94). En effet, dès la troisième itération, on a un contact d'ordre 7 avec l'espèce des graphes série-parallèle, alors qu'avec l'itération simple, en trois étapes, on obtient seulement un contact d'ordre 3.

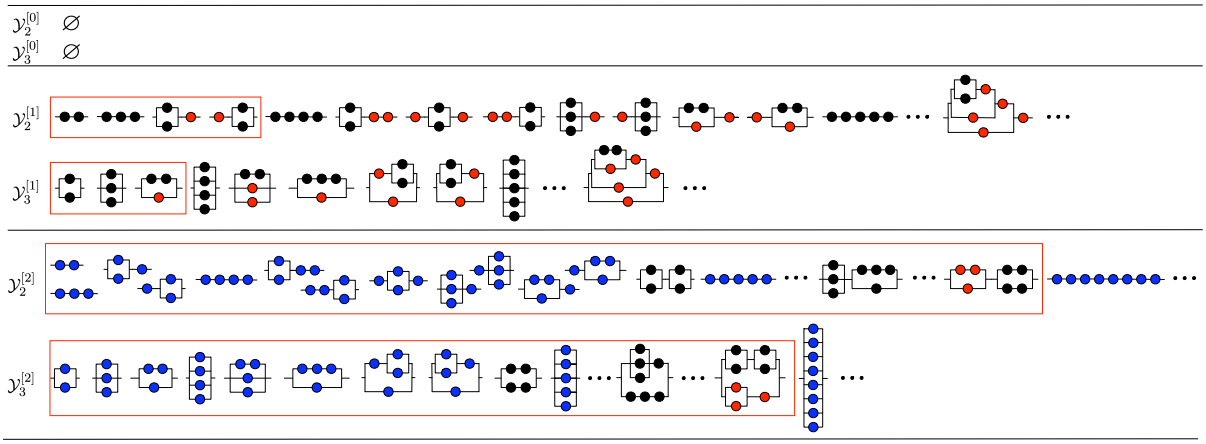


FIG. 4 – Premières itérations pour les graphes série-parallèle.

2 Itération de Newton optimisée

Dans cette section, nous présentons une optimisation de l'itération de Newton précédente. L'idée est d'éviter de passer trop de temps à calculer la totalité de l'inverse de la matrice jacobienne, à chaque étape de l'itération de Newton. On produira ainsi moins de structures dont la taille dépasse le contact obtenu. Dans toute cette section nous utiliserons la notation compacte suivante :

$$\boldsymbol{\mathcal{A}}^{[n]} = \boldsymbol{\partial}\boldsymbol{\mathcal{H}}/\boldsymbol{\partial}\boldsymbol{\mathcal{Y}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}}^{[n]}).$$

2.1 Itération

Pour engendrer l'espèce $\boldsymbol{\mathcal{Y}}^{[n+1]}$, il est nécessaire d'engendrer l'inverse de la matrice $\mathbf{1} - \boldsymbol{\mathcal{A}}^{[n]}$. Or calculer l'inverse $\boldsymbol{\mathcal{U}}$ d'une matrice $\boldsymbol{\mathcal{M}}$ revient à résoudre l'équation :

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}) = \boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{U}}^{-1} = \mathbf{0}.$$

Pour ce faire, on utilise l'itération Newton[34] :

$$\boldsymbol{\mathcal{U}}^{[i+1]} = \boldsymbol{\mathcal{U}}^{[i]} - \frac{\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{[i]})}{\boldsymbol{\mathcal{F}}'(\boldsymbol{\mathcal{U}}^{[i]})} = \boldsymbol{\mathcal{U}}^{[i]} - \frac{\boldsymbol{\mathcal{M}} - (\boldsymbol{\mathcal{U}}^{[i]})^{-1}}{(\boldsymbol{\mathcal{U}}^{[i]})^{-2}} = \boldsymbol{\mathcal{U}}^{[i]} - \boldsymbol{\mathcal{M}} \cdot (\boldsymbol{\mathcal{U}}^{[i]})^2 + \boldsymbol{\mathcal{U}}^{[i]}.$$

Pour $\boldsymbol{\mathcal{M}} = \mathbf{1} - \boldsymbol{\mathcal{A}}^{[n]}$, on obtient :

$$\boldsymbol{\mathcal{U}}^{[i+1]} = \boldsymbol{\mathcal{U}}^{[i]} + \boldsymbol{\mathcal{U}}^{[i]}\Big(\boldsymbol{\mathcal{A}}^{[n]} \cdot \boldsymbol{\mathcal{U}}^{[i]} - (\boldsymbol{\mathcal{U}}^{[i]} - \mathbf{1})\Big).$$

[34] L'idée d'utiliser l'itération de Newton pour calculer l'inverse d'une matrice remonte au moins à [Sch33].

Notons, en passant, que le second terme est symétrique ; en le développant, on obtient :

$$\mathcal{U}^{[i]}\Big(\mathcal{A}^{[n]}\cdot\mathcal{U}^{[i]}-(\mathcal{U}^{[i]}-\mathbf{1})\Big)=\Big(\mathcal{A}^{[n]}\cdot\mathcal{U}^{[i]}-(\mathcal{U}^{[i]}-\mathbf{1})\Big)\mathcal{U}^{[i]}. \tag{4}$$

Une autre remarque importante est qu'à l'étape $n+1$ de l'itération sur les $\mathcal{Y}$-structures, on ne construit pas l'intégralité des structures de $(\mathbf{1}-\mathcal{A}^{[n]})^{-1}$, mais uniquement celles qui sont nécessaires au doublement du contact entre les approximations de $\mathcal{Y}$. De plus la croissance de l'itération sur les $\mathcal{Y}$-structures, implique que l'espèce $\mathcal{A}^{[n-1]}$ est une sous-espèce de $\mathcal{A}^{[n]}$. Par conséquent, l'inverse calculé à chaque étape a déjà été partiellement engendré à l'étape précédente :

$$(\mathbf{1}-\mathcal{A}^{[n-1]})^{-1}\subset(\mathbf{1}-\mathcal{A}^{[n]})^{-1}.$$

Pour ne pas calculer à nouveau des structures déjà engendrées, nous proposons une itération optimisée qui consiste à initialiser l'itération pour engendrer $(\mathbf{1}-\mathcal{A}^{[n]})^{-1}$ avec les structures de $(\mathbf{1}-\mathcal{A}^{[n-1]})^{-1}$, construites lors du calcul de $\mathcal{Y}^{[n]}$. L'idée est donc de faire ces deux itérations en parallèle, ce qui nous donne l'itération de la proposition suivante.

Proposition 15. *Soit $\mathcal{Y}=\mathcal{H}(\mathcal{Z},\mathcal{Y})$ un système tel que $\mathcal{H}(\mathbf{0},\mathbf{0})=\mathbf{0}$ et $\partial\mathcal{H}/\partial\mathcal{Y}(\mathbf{0},\mathbf{0})$ est nilpotente. L'itération combinatoire définie par :*

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]}+\mathcal{U}^{[n+1]}\Big(\mathcal{H}(\mathcal{Z},\mathcal{Y}^{[n]})-\mathcal{Y}^{[n]}\Big), \tag{5}$$

$$\mathcal{U}^{[n+1]} = \mathcal{U}^{[n]}+\mathcal{U}^{[n]}\Big(\mathcal{A}^{[n]}\mathcal{U}^{[n]}-(\mathcal{U}^{[n]}-\mathbf{1})\Big), \tag{6}$$

avec $\mathcal{A}^{[n]}=\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z},\mathcal{Y}^{[n]})$, $\mathcal{Y}^{[0]}=\mathbf{0}$ et $\mathcal{U}^{[0]}=\mathbf{1}$, converge vers l'espèce $\mathcal{Y}$, solution du système combinatoire $\mathcal{Y}=\mathcal{H}(\mathcal{Z},\mathcal{Y})$. De plus, cette convergence est quadratique.

Intuitivement, les structures de $\mathcal{U}^{[n+1]}$ sont :

- soit des structures que nous avons déjà construites à l'itération précédente ; elles proviennent du terme $\mathcal{U}^{[n]}$ et sont uniquement composées de $\mathcal{Y}^{[n-1]}$-structures,
- soit des "nouvelles" structures, qui sont éventuellement composées de $\mathcal{Y}^{[n]}$-structures et qui proviennent du terme $\mathcal{U}^{[n]}\big(\mathcal{A}^{[n]}\mathcal{U}^{[n]}-(\mathcal{U}^{[n]}-\mathbf{1})\big)$.

Nous verrons dans ce qui suit que, de cette façon, on construit effectivement, à chaque étape, suffisamment de structures appartenant à $(\mathbf{1}-\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z},\mathcal{Y}))^{-1}$ pour que l'itération globale converge quadratiquement vers la solution combinatoire du système $\mathcal{Y}=\mathcal{H}(\mathcal{Z},\mathcal{Y})$. La preuve de cette proposition est constituée par les sections 2.2 à 2.4.

2.2 Croissance

Avant tout, nous devons vérifier que la double itération de la proposition 15 est bien définie, c'est-à-dire que les soustractions qu'elle comporte définissent bien des espèces.

Lemme 9. *L'itération donnée par les équations* (5) *et* (6) *est bien définie : quel que soit $n\geq 0$,*

$$\begin{cases}\mathcal{U}^{[n]}\subset\mathbf{1}+\mathcal{A}^{[n]}\mathcal{U}^{[n]},\\ \mathcal{Y}^{[n]}\subset\mathcal{H}(\mathcal{Z},\mathcal{Y}^{[n]}).\end{cases}$$

Démonstration. On démontre les deux inclusions en parallèle, par récurrence sur n.
1. Si $n=0$, $\mathcal{U}^{[0]}=\mathbf{1}$ et la propriété est vérifiée.

Sinon, supposons que la propriété est vraie au rang n. Cela implique notamment que :

$$\boldsymbol{\mathcal{U}}^{[n]} \subset \boldsymbol{\mathcal{U}}^{[n+1]} \quad \text{et} \quad \boldsymbol{\mathcal{Y}}^{[n]} \subset \boldsymbol{\mathcal{Y}}^{[n+1]}.$$

Nous montrons maintenant que toute $\boldsymbol{\mathcal{U}}^{[n+1]}$-structure appartient à l'espèce $\mathbf{1} + \boldsymbol{\mathcal{A}}^{[n+1]}\boldsymbol{\mathcal{U}}^{[n+1]}$. Par définition, $\boldsymbol{\mathcal{U}}^{[n+1]} = \boldsymbol{\mathcal{U}}^{[n]}(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[n+1]})$ avec $\boldsymbol{\mathcal{T}}^{[n+1]} = \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} - (\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1})$. Par hypothèse de récurrence, $\boldsymbol{\mathcal{U}}^{[n]} \subset \mathbf{1} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]}$; on a donc :

$$\boldsymbol{\mathcal{U}}^{[n+1]} \subset \big(\mathbf{1} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]}\big)\big(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[n+1]}\big) = \mathbf{1} + \boldsymbol{\mathcal{T}}^{[n+1]} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]}(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[n+1]}).$$

Or, d'une part, d'après la définition de l'itération (6), $\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]}(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[n+1]}) \subset \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]}$; et d'autre part, l'espèce $\boldsymbol{\mathcal{T}}^{[n+1]}$ est contenue dans $\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} \subset \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]}$. On a donc :

$$\boldsymbol{\mathcal{U}}^{[n+1]} \subset \mathbf{1} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]}. \tag{7}$$

De plus, $\boldsymbol{\mathcal{A}}^{[n]} \subset \boldsymbol{\mathcal{A}}^{[n+1]}$, car $\boldsymbol{\mathcal{Y}}^{[n]} \subset \boldsymbol{\mathcal{Y}}^{[n+1]}$, et donc finalement :

$$\boldsymbol{\mathcal{U}}^{[n+1]} \subset \mathbf{1} + \boldsymbol{\mathcal{A}}^{[n+1]}\boldsymbol{\mathcal{U}}^{[n+1]}.$$

2. Pour $n = 0$, $\boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0}$ est trivialement contenue dans $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \mathbf{0})$.
Sinon, supposons que la propriété est vraie au rang n. Une structure de l'espèce $\boldsymbol{\mathcal{Y}}^{[n+1]}$ est, par définition, soit une $\boldsymbol{\mathcal{Y}}^{[n]}$-structure qui, par hypothèse de récurrence, appartient à $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]})$, elle-même contenue dans $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]})$, soit une structure de l'espèce

$$\boldsymbol{\mathcal{U}}^{[n+1]}\Big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\Big).$$

D'après l'équation (7), cette espèce est contenue dans :

$$(\mathbf{1} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]})\Big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\Big) = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]}\Big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\Big).$$

Par définition de l'itération (5), on a $\boldsymbol{\mathcal{U}}^{[n+1]}\Big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\Big) \subset \boldsymbol{\mathcal{Y}}^{[n+1]}$, donc :

$$\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]}\Big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\Big) \subset \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{Y}}^{[n+1]}.$$

Or, une structure appartenant à l'espèce $\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{Y}}^{[n+1]}$ est une $\boldsymbol{\mathcal{H}}$-assemblée dont les membres sont des $\boldsymbol{\mathcal{Y}}^{[n]}$-structures, sauf un, qui est une $\boldsymbol{\mathcal{Y}}^{[n+1]}$-structure ; par conséquent, c'est une structure de l'espèce $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]})$, ce qui signifie que :

$$\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{Y}}^{[n+1]} \subset \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]}).$$

Par ailleurs, on a aussi $\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) \subset \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]})$; on a donc finalement l'inclusion suivante :

$$\boldsymbol{\mathcal{Y}}^{[n+1]} \subset \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]}). \qquad \square$$

La démonstration précédente implique que les suites d'espèces définies par l'itération des équations (5) et (6) sont croissantes. On a donc le corollaire suivant.

Corollaire 3. *Étant donné une espèce $\boldsymbol{\mathcal{H}}$, les suites $(\boldsymbol{\mathcal{Y}}^{[n]})_{n\in\mathbb{N}}$ et $(\boldsymbol{\mathcal{U}}^{[n]})_{n\in\mathbb{N}}$ définies par l'itération des équations* (5) *et* (6) *sont croissantes.*

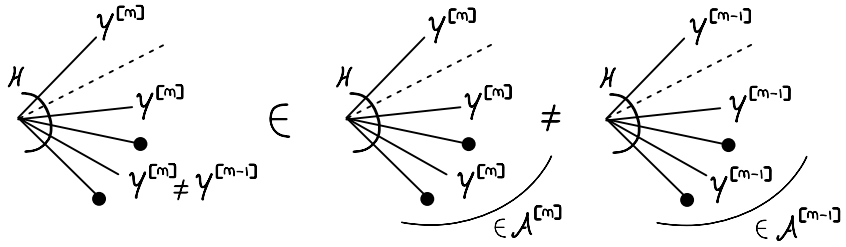


FIG. 5 – Exemple de $\mathcal{B}^{[n]}$-structure.

2.3 Non ambiguïté

Nous devons également nous assurer que l'itération de la proposition 15 n'est pas ambiguë, c'est-à-dire qu'elle n'engendre pas de doublons. La seule ambiguïté possible se trouve au niveau des $\mathcal{U}^{[n+1]}$-structures. En décomposant l'équation (6), on obtient :

$$\begin{aligned}\mathcal{U}^{[n+1]} &= \mathcal{U}^{[n]} + \mathcal{U}^{[n]}\mathcal{T}^{[n+1]} \\ \mathcal{T}^{[n+1]} &= \mathcal{A}^{[n]}\mathcal{U}^{[n]} - (\mathcal{U}^{[n]} - \mathbf{1}).\end{aligned}$$

On peut réécrire l'équation définissant $\mathcal{T}^{[n+1]}$ en introduisant $\mathcal{A}^{[n-1]}$, puis en remplaçant $\mathcal{U}^{[n]}$ par $\mathcal{U}^{[n-1]} + \mathcal{U}^{[n-1]}\mathcal{T}^{[n]}$. Pour faciliter la lecture, on notera $\mathcal{B}^{[n]} = \mathcal{A}^{[n]} - \mathcal{A}^{[n-1]}$ l'espèce des $\partial\mathcal{H}/\partial\mathcal{Y}$-assemblées de $\mathcal{Y}^{[n]}$-structures dont l'un des membres, au moins, n'appartient pas à l'espèce $\mathcal{Y}^{[n-1]}$ (voir figure 5).

$$\begin{aligned}\mathcal{T}^{[n+1]} &= \mathcal{A}^{[n-1]}\mathcal{U}^{[n]} - (\mathcal{U}^{[n]} - \mathbf{1}) + (\mathcal{A}^{[n]} - \mathcal{A}^{[n-1]})\mathcal{U}^{[n]} \\ &= \mathcal{A}^{[n-1]}(\mathcal{U}^{[n-1]} + \mathcal{U}^{[n-1]}\mathcal{T}^{[n]}) - (\mathcal{U}^{[n-1]} + \mathcal{U}^{[n-1]}\mathcal{T}^{[n]} - \mathbf{1}) + \mathcal{B}^{[n]}\mathcal{U}^{[n]}.\end{aligned}$$

En regroupant différemment les termes et en utilisant la définition de $\mathcal{T}^{[n]}$, on obtient :

$$\begin{aligned}\mathcal{T}^{[n+1]} &= \mathcal{T}^{[n]} + \mathcal{A}^{[n-1]}\mathcal{U}^{[n-1]}\mathcal{T}^{[n]} - \mathcal{U}^{[n-1]}\mathcal{T}^{[n]} + \mathcal{B}^{[n]}\mathcal{U}^{[n]} \\ &= \mathcal{B}^{[n]}\mathcal{U}^{[n]} + (\mathcal{T}^{[n]})^2.\end{aligned}$$

La suite de cette section a pour but de démontrer que l'itération suivante, qui est, comme nous venons de le voir, équivalente à l'itération de la proposition 15, n'est pas ambiguë :

$$\mathcal{U}^{[n+1]} = \mathcal{U}^{[n]} + \mathcal{U}^{[n]}\mathcal{T}^{[n+1]} \tag{8}$$

$$\mathcal{T}^{[n+1]} = \mathcal{B}^{[n]}\mathcal{U}^{[n]} + (\mathcal{T}^{[n]})^2 \tag{9}$$

$$\mathcal{B}^{[n]} = \mathcal{A}^{[n]} - \mathcal{A}^{[n-1]} \tag{10}$$

avec

$$\mathcal{Y}^{[0]} = \mathbf{0}, \quad \mathcal{U}^{[0]} = \mathbf{1}, \quad \mathcal{T}^{[0]} = 0, \quad \mathcal{B}^{[0]} = \mathcal{A}^{[0]}.$$

Les structures définies par les équations (8) à (10) sont des $\mathcal{H}$-arborescences qui peuvent être décomposées comme des suites de la forme $\beta_1 \cdots \beta_\ell$, telles que $\beta_j \in (\mathcal{B}^{[0]} + \cdots + \mathcal{B}^{[i]})$ pour $1 \leq j \leq \ell$. Par définition, les espèces $\mathcal{B}^{[i]}$ et $\mathcal{B}^{[j]}$ sont disjointes pour $i \neq j$. Montrer que l'itération de Newton optimisée n'est pas ambiguë revient donc à montrer que toute suite β de la forme $\beta_1 \cdots \beta_\ell$ est engendrée de façon unique par cette itération. Dans un premier temps, on montre que la suite β se décompose canoniquement comme une suite de $\mathcal{T}^{[i]}$-structures d'indices croissants (proposition 16), et ensuite, on montre que cela implique que toute suite de cette forme est engendrée de façon unique comme une $\mathcal{U}^{[k]}$-structure, pour un certain entier k minimal (corollaire 4).

Proposition 16. *Soit β une structure de la forme $\beta_1 \cdots \beta_\ell$ et i l'indice tel que :*

$$i := \max\{p \mid \beta_j \in \boldsymbol{\mathcal{B}}^{[p]},\ 1 \leq j \leq \ell\}.$$

La structure β se décompose de façon unique comme une structure de l'espèce

$$(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]})(\boldsymbol{\mathcal{T}}^{[i+1]})^q, \quad \textit{avec } q \in \mathbb{N}.$$

Démonstration. On utilise une récurrence sur l'indice i. Pour $i = 0$, la structure β est uniquement composée de $\boldsymbol{\mathcal{B}}^{[0]}$-structures, donc $\beta \in (\boldsymbol{\mathcal{B}}^{[0]})^\ell$. Or $\boldsymbol{\mathcal{B}}^{[0]} = \boldsymbol{\mathcal{T}}^{[1]}$, donc β se décompose de façon unique comme une $(\boldsymbol{\mathcal{T}}^{[1]})^\ell$-structure.

Pour $i \geq 1$, on "découpe" la structure β au niveau des β_j qui sont des $\boldsymbol{\mathcal{B}}^{[i]}$-structures ; elle s'écrit alors de façon unique :

$$\beta = \omega_0 \beta_{i_1} \omega_1 \beta_{i_2} \omega_2 \beta_{i_3} \cdots \beta_{i_\ell} \omega_\ell \,,$$

avec $\beta_{i_j} \in \boldsymbol{\mathcal{B}}^{[i]}$ et $\omega_j \in (\boldsymbol{\mathcal{B}}^{[0]} + \cdots + \boldsymbol{\mathcal{B}}^{[i-1]})^{r_j}$, avec $r_j \in \mathbb{N}$, pour $1 \leq j \leq \ell$. Par hypothèse de récurrence, chaque ω_j s'écrit de façon unique sous la forme :

$$\omega_j = \mu_j \nu_j \,, \tag{11}$$

avec $\mu_j \in (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \dots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[k_j]})$, pour $1 \leq k_j \leq i-1$ et $\nu_j \in (\boldsymbol{\mathcal{T}}^{[k_j+1]})^{q_j}$, pour $q_j \in \mathbb{N}$. En décomposant q_j en base $i+1$ (cette décomposition est unique), on peut réécrire ν_j d'une unique façon, sous la forme :

$$\nu_j = \nu_{j_0} \nu_{j_1} \,, \tag{12}$$

avec $\nu_{j_0} \in (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[k_j+1]}) \dots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]})$, $\nu_{j_1} \in (\boldsymbol{\mathcal{T}}^{[i+1]})^{p_j}$ et $p_j = \lfloor \frac{q_j}{i+1} \rfloor$. Finalement, $\omega_j = \mu_j \nu_{j_0} \nu_{j_1}$ se décompose comme une structure de

$$(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]})(\boldsymbol{\mathcal{T}}^{[i+1]})^{p_j}.$$

Cette décomposition est unique, sinon, celles des équation (11) et (12) ne le seraient pas.

Par ailleurs, par itération des équations (8) à (10), on a $\boldsymbol{\mathcal{U}}^{[i]} = (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]})$, donc

$$\boldsymbol{\mathcal{B}}^{[i]}(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]}) = \boldsymbol{\mathcal{B}}^{[i]} \boldsymbol{\mathcal{U}}^{[i]} \subset \boldsymbol{\mathcal{T}}^{[i+1]}.$$

Par conséquent, $\beta_{i_j} \mu_j \nu_{j_0} \in \boldsymbol{\mathcal{T}}^{[i+1]}$. On en déduit que $\beta_{i_j} \omega_j = \beta_{i_j} \mu_j \nu_{j_0} \nu_{j_1}$ se décompose canoniquement comme une structure de $(\boldsymbol{\mathcal{T}}^{[i+1]})^{p_j+1}$. Là encore, cette décomposition est unique, sinon cela contredirait l'unicité de (11) et (12). Enfin, comme les $\boldsymbol{\mathcal{B}}^{[i]}$-structures sont nécessairement en tête des $\boldsymbol{\mathcal{T}}^{[i+1]}$-structures, on décompose $\beta = \omega_0 \prod_{j=1}^{\ell} \beta_{i_j} \omega_j$ de façon unique comme une structure de l'espèce :

$$(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]})(\boldsymbol{\mathcal{T}}^{[i+1]})^q, \quad \text{avec } q = \sum_{j=0}^{\ell} (p_j + 1). \qquad \square$$

Exemple 24. Nous illustrons la décomposition de cette preuve pour la structure ω :

$$\omega = \beta_0\, \beta_2\, \beta_0\, \beta_0\, \beta_1\, \beta_1\, \beta_0\, \beta_0\, \beta_1.$$

On décompose ω suivant les β_2 ; on a alors $\omega = \omega_0\, \beta_2\, \omega_1$, avec $\omega_0 = \beta_0 \in \boldsymbol{\mathcal{T}}^{[1]}$ et

$$\omega_1 = \beta_0\, \beta_0\, \beta_1\, \beta_1\, \beta_0\, \beta_0\, \beta_1.$$

On décompose ensuite ω_1 suivant les β_1, ce qui donne $\omega_1 = \omega_{1_0}\,\beta_1\,\beta_1\,\omega_{1_1}\,\beta_1$, avec

$$\omega_{1_0} = \omega_{1_1} = \beta_0\beta_0 \in (\boldsymbol{\mathcal{T}}^{[1]})^2 \subset \boldsymbol{\mathcal{T}}^{[2]}.$$

Par conséquent, $\omega_1 \in (\boldsymbol{\mathcal{T}}^{[2]})^5 \subset \boldsymbol{\mathcal{T}}^{[2]}(\boldsymbol{\mathcal{T}}^{[3]})^2$ et donc :

$$\beta_2\omega_1 \in \boldsymbol{\mathcal{B}}^{[2]}\boldsymbol{\mathcal{U}}^{[2]}\boldsymbol{\mathcal{T}}^{[2]}(\boldsymbol{\mathcal{T}}^{[3]})^2 \subset (\boldsymbol{\mathcal{T}}^{[3]})^3.$$

Finalement, ω se décompose de façon unique comme une structure de l'espèce $\boldsymbol{\mathcal{T}}^{[1]}(\boldsymbol{\mathcal{T}}^{[3]})^3$.

Corollaire 4. *Soit β une structure de la forme $\beta_1 \cdots \beta_\ell$ et i l'indice tel que :*

$$i := \max\{p \mid \beta_j \in \boldsymbol{\mathcal{B}}^{[p]},\ 1 \leq j \leq \ell\}.$$

Il existe $k \in \mathbb{N}$ minimal tel que β se décompose de façon unique comme une structure de $\boldsymbol{\mathcal{U}}^{[k]}$.

Démonstration. D'après la proposition 16, la structure β se décompose de façon unique sous la forme $\beta = \mu\nu$, telle que :

$$\mu \in (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[1]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i]}) \quad \text{et} \quad \nu \in (\boldsymbol{\mathcal{T}}^{[i+1]})^q, \quad \text{avec } q \in \mathbb{N}.$$

Si $\sum_{i=0}^{p} q_i 2^i = q$ est la décomposition de q en base 2, avec $p = \lfloor \log_2 q \rfloor$, $q_i \in \{0, 1\}$ et $q_p = 1$, alors ν se décompose de façon unique sous la forme $\nu = \nu_0 \ldots \nu_p$, avec $\nu_\ell \in (\boldsymbol{\mathcal{T}}^{[i+1]})^{2^\ell q_\ell}$. Or, comme les β_j sont au plus des $\boldsymbol{\mathcal{B}}^{[i]}$-structures, on a :

$$\nu_\ell \in (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i+\ell+1]}), \quad 0 \leq \ell \leq p,$$

et donc :

$$\nu \in (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i+1]})(\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i+2]}) \cdots (\mathbf{1} + \boldsymbol{\mathcal{T}}^{[i+p]})\boldsymbol{\mathcal{T}}^{[i+p+1]}$$

De l'unicité de dérivation de β et de l'unicité de la décomposition de q en base 2, on déduit l'unique décomposition de β comme une structure de l'espèce $\boldsymbol{\mathcal{U}}^{[k]} - \boldsymbol{\mathcal{U}}^{[k-1]}$, avec $k = i+p+1$. □

Exemple 25. La structure ω de l'exemple 24 se décompose de façon unique comme une structure de l'espèce $\boldsymbol{\mathcal{T}}^{[1]}\boldsymbol{\mathcal{T}}^{[3]}\boldsymbol{\mathcal{T}}^{[4]}$.

Le corollaire précédent nous assure que, quel que soit $i \geq 0$, toute suite formée de structures appartenant à l'espèce $\boldsymbol{\mathcal{B}}^{[i]} + \cdots + \boldsymbol{\mathcal{B}}^{[0]}$ est dérivée de façon unique par l'équation (8), ce qui implique que l'itération de Newton optimisée n'est pas ambiguë.

2.4 Convergence

Le lemme suivant, associé à la croissance de l'itération de Newton optimisée, nous assure, d'une part, que nous sommes bien dans les conditions de la proposition 12 du chapitre précédent (page 89), et que, par conséquent, cette itération est convergente, et d'autre part qu'elle converge quadratiquement vers la solution de $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}})$, ce qui conclut la preuve de la proposition 15.

Lemme 10. *Soit l'itération combinatoire définie par :*

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{Y}}^{[n]} + \boldsymbol{\mathcal{U}}^{[n+1]}\Big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\Big)$$
$$\boldsymbol{\mathcal{U}}^{[n+1]} = \boldsymbol{\mathcal{U}}^{[n]} + \boldsymbol{\mathcal{U}}^{[n]}\Big(\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} - (\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1})\Big),$$

avec $\boldsymbol{\mathcal{A}}^{[n]} = \boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]})$, $\boldsymbol{\mathcal{Y}}^{[0]} = 0$ et $\boldsymbol{\mathcal{U}}^{[0]} = \mathbf{1}$.

Si, pour tout n et tout k positifs, on a $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$ et $\boldsymbol{\mathcal{U}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n+1]}$, alors :

$$\boldsymbol{\mathcal{Y}}^{[n+1]} =_{2k+1} \boldsymbol{\mathcal{Y}}^{[n+2]} \quad \text{et} \quad \boldsymbol{\mathcal{U}}^{[n+1]} =_k \boldsymbol{\mathcal{U}}^{[n+2]}.$$

Démonstration. Au cours de cette preuve, nous utiliserons l'hypothèse $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$ sous différentes formes équivalentes :

$$\begin{aligned}\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]} \;\Leftrightarrow\; \boldsymbol{\mathcal{Y}}^{[n+1]} - \boldsymbol{\mathcal{Y}}^{[n]} =_k \mathbf{0} \;&\Leftrightarrow\; \boldsymbol{\mathcal{U}}^{[n+1]}\big(\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]}\big) =_k \mathbf{0}\\ &\Leftrightarrow\; \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}) - \boldsymbol{\mathcal{Y}}^{[n]} =_k \mathbf{0},\end{aligned}$$

et de même pour l'hypothèse $\boldsymbol{\mathcal{U}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n+1]}$:

$$\boldsymbol{\mathcal{U}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n+1]} \;\Leftrightarrow\; \boldsymbol{\mathcal{U}}^{[n]}\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n]}(\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1}) \;\Leftrightarrow\; \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} - (\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1}) =_{\lfloor k/2 \rfloor} \mathbf{0}.$$

Dans les deux cas, la dernière équivalence est due au fait que l'espèce $\mathbf{1}$ est contenue dans l'espèce $\boldsymbol{\mathcal{U}}^{[i]}$, quel que soit $i \geq 0$. La preuve se fait en deux temps : en utilisant conjointement les hypothèses $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$ et $\boldsymbol{\mathcal{U}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n+1]}$, on commence par montrer l'accroissement du contact pour les $\boldsymbol{\mathcal{U}}$-structures et ensuite, on utilise ce résultat pour donner le contact sur les $\boldsymbol{\mathcal{Y}}$-structures.

1. La première étape est de montrer que les hypothèses de récurrence impliquent :

$$\boldsymbol{\mathcal{U}}^{[n+1]}\boldsymbol{\mathcal{A}}^{[n+1]}\boldsymbol{\mathcal{U}}^{[n+1]} =_k \boldsymbol{\mathcal{U}}^{[n+1]}(\boldsymbol{\mathcal{U}}^{[n+1]} - \mathbf{1}). \tag{13}$$

Soit une structure γ appartenant à l'espèce $\boldsymbol{\mathcal{U}}^{[n+1]}\boldsymbol{\mathcal{A}}^{[n+1]}\boldsymbol{\mathcal{U}}^{[n+1]}$ et portée par une cardinalité inférieure à k. La $\boldsymbol{\mathcal{A}}^{[n+1]}$-structure centrale de γ est elle-même portée par une cardinalité $\leq k$; comme $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}^{[n+1]}$, c'est donc une $\boldsymbol{\mathcal{A}}^{[n]}$-structure. On peut alors décomposer γ en $\gamma_1 \cdot \gamma_2 \cdot \gamma_3$, avec γ_1, $\gamma_3 \in \boldsymbol{\mathcal{U}}^{[n+1]}$ et $\gamma_2 \in \boldsymbol{\mathcal{A}}^{[n]}$.

On distingue plusieurs cas selon les cardinalités des ensembles sous-jacents de γ_1 et γ_3. On s'intéresse tout d'abord au cas où γ_1 et γ_3 sont portées par des cardinalités $\leq k/2$; γ appartient alors à l'espèce $\boldsymbol{\mathcal{U}}^{[n]}\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]}$. Or, par réécriture de la définition de $\boldsymbol{\mathcal{U}}^{[n+1]}$, on a :

$$\boldsymbol{\mathcal{U}}^{[n]}\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} \;=\; \boldsymbol{\mathcal{U}}^{[n]}(\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1}) + \boldsymbol{\mathcal{U}}^{[n+1]} - \boldsymbol{\mathcal{U}}^{[n]} \;=_{\lfloor k/2 \rfloor}\; \boldsymbol{\mathcal{U}}^{[n]}(\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1}).$$

De plus, l'itération est croissante, *i.e.*, $\boldsymbol{\mathcal{U}}^{[n]} \subset \boldsymbol{\mathcal{U}}^{[n+1]}$, donc γ appartient à $\boldsymbol{\mathcal{U}}^{[n+1]}(\boldsymbol{\mathcal{U}}^{[n+1]} - \mathbf{1})$.

Il nous reste à étudier le cas où l'une des structures γ_1 ou γ_3 est portée par une cardinalité $> k/2$. Comme γ est portée par une cardinalité $\leq k$, cela implique que soit $\gamma_1 \cdot \gamma_2$, soit $\gamma_2 \cdot \gamma_3$ est portée par une cardinalité $\leq k/2$. Cela entraîne que $\gamma_1 \cdot \gamma_2$ appartient à $\boldsymbol{\mathcal{U}}^{[n]}\boldsymbol{\mathcal{A}}^{[n]}$ ou, symétriquement, que $\gamma_2 \cdot \gamma_3$ appartient à $\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]}$. Or, par hypothèse de récurrence, on a :

$$\boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1}.$$

D'après la remarque (4), on a symétriquement :

$$\boldsymbol{\mathcal{U}}^{[n]}\boldsymbol{\mathcal{A}}^{[n]} =_{\lfloor k/2 \rfloor} \boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1},$$

et donc γ appartient soit à $\boldsymbol{\mathcal{U}}^{[n+1]}(\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1})$, soit à $(\boldsymbol{\mathcal{U}}^{[n]} - \mathbf{1})\boldsymbol{\mathcal{U}}^{[n+1]}$; comme $\boldsymbol{\mathcal{U}}^{[n]} \subset \boldsymbol{\mathcal{U}}^{[n+1]}$, dans les deux cas, γ appartient à $\boldsymbol{\mathcal{U}}^{[n+1]}(\boldsymbol{\mathcal{U}}^{[n+1]} - \mathbf{1})$. Ainsi, nous avons démontré le contact donné par l'équation (13), ce qui implique notamment que l'espèce $\boldsymbol{\mathcal{U}}^{[n+1]}$ contient toutes les séquences de $\boldsymbol{\mathcal{A}}^{[n]}$-structures portées par une cardinalité $\leq k$:

$$\mathbf{1} + \boldsymbol{\mathcal{A}}^{[n]}\boldsymbol{\mathcal{U}}^{[n+1]} =_k \boldsymbol{\mathcal{U}}^{[n+1]}. \tag{14}$$

2. La seconde étape de cette démonstration est de montrer que les hypothèses de récurrence impliquent :

$$\boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n+1]}) - \boldsymbol{\mathcal{Y}}^{[n+1]} =_{2k+1} \mathbf{0}.$$

Pour cela, on considère une $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n+1]})$-structure quelconque que l'on nomme γ, portée par une cardinalité $\leq 2k+1$. Par définition, γ est une $\mathcal{H}$-assemblée de $\mathcal{Y}^{[n+1]}$-structures. Si toutes ces structures sont portées par des cardinalités $\leq k$, alors ce sont des $\mathcal{Y}^{[n]}$-structures et dans ce cas, γ est une $\mathcal{Y}^{[n+1]}$-structure. Dans le cas contraire, une seule des $\mathcal{Y}^{[n+1]}$-structures qui composent γ peut être portée par une cardinalité $> k$, sinon, γ serait portée par une cardinalité totale supérieure à $2k+1$. La structure γ appartient alors à l'espèce :

$$\mathcal{A}^{[n]}(\mathcal{Y}^{[n+1]} - \mathcal{Y}^{[n]}) = \mathcal{A}^{[n]}\mathcal{U}^{[n+1]}\Big(\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}\Big).$$

Par hypothèse, $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]} =_k \mathbf{0}$, donc γ se décompose en $\gamma = \beta \cdot \delta$, de telle façon que la structure $\delta \in \mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}$ est portée par une cardinalité $> k$ et la structure β appartenant à $\mathcal{A}^{[n]}\mathcal{U}^{[n+1]}$ est portée par une cardinalité $\leq k$. D'après (14), β appartient donc à l'espèce $\mathcal{U}^{[n+1]}$ et γ appartient à :

$$\mathcal{U}^{[n+1]}\Big(\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) - \mathcal{Y}^{[n]}\Big) \subset \mathcal{Y}^{[n+1]}.$$

Nous avons ainsi montré que toute $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n+1]})$-structure portée par une cardinalité $\leq 2k+1$ est une $\mathcal{Y}^{[n+1]}$-structure. □

2.5 Caractérisation de l'itération de Newton optimisée

Contrairement à l'itération de Newton classique, nous ne savons pas, a priori, caractériser de façon précise les structures produites à chaque étape de cette itération optimisée. Néanmoins, on peut dire qu'elles ont un nombre de Strahler et une hauteur bornés. En effet, les $\mathcal{H}$-arborescences produites à la n-ième étape de l'itération optimisée ont un nombre de Strahler inférieur à n, puisqu'elles sont contenues dans l'espèce $\mathcal{Y}^{[n]}$ obtenue par itération de Newton classique. Nous montrons maintenant que pour l'itération optimisée, la hauteur des $\mathcal{H}$-arborescences produites est également bornée.

Propriété 5. *Soit $(\mathcal{Y}^{[n]})_{n\in\mathbb{N}}$ la suite des espèces obtenues par itération de Newton optimisée pour le système $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$; la hauteur maximale d'une $\mathcal{Y}^{[n]}$-structure est $2^{n+1} - 2$.*

Démonstration. Une $\mathcal{Y}^{[n]}$-structure est une $\mathcal{U}^{[n]}$-structure suivie d'une $\mathcal{H}$-assemblée dont les membres sont des $\mathcal{Y}^{[n-1]}$-structures. Or, une $\mathcal{U}^{[n]}$-structure est formée d'un chemin se terminant par une branche libre et sur lequel sont greffées des $\mathcal{Y}^{[n-1]}$-structures (voir figure 6). La hauteur maximale $h_Y(n)$ d'une $\mathcal{Y}^{[n]}$-structure est donc la longueur maximale du chemin de la racine vers la branche libre d'une $\mathcal{U}^{[n]}$-structure à laquelle s'ajoute la hauteur maximale d'une structure de l'espèce $\mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n-1]})$. On obtient finalement la récurrence suivante :

$$h_Y(n) = \ell_U(n) + h_Y(n-1) + 1, \quad \text{avec} \;\; h_Y(0) = 0$$

où $\ell_U(n)$ est la longueur maximale d'une séquence de $\mathcal{A}^{[n-1]}$-structures appartenant à $\mathcal{U}^{[n]}$ (c'est la longueur du chemin de la racine à la dernière branche libre, ou plus visuellement, du chemin en pointillé dans nos schémas) ; par construction, on a :

$$\ell_U(n) = 2\ell_U(n-1) + 1, \quad \text{et} \quad \ell_U(0) = 0.$$

En résolvant la récurrence précédente, on obtient $\ell_U(n) = 2^n - 1$ et finalement on trouve que la hauteur maximale d'une $\mathcal{Y}^{[n]}$-structure est $h_Y(n) = 2^{n+1} - 2$. □

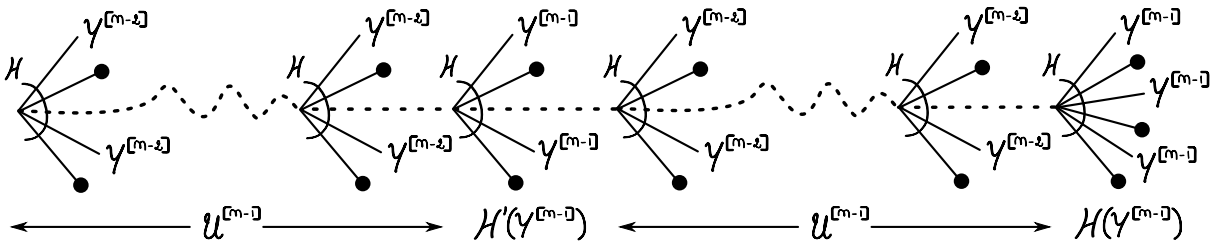


FIG. 6 – Décomposition d'une $\mathcal{Y}^{[n]}$-structure.

Ainsi, la principale différence entre les deux itérations de Newton est que, pour obtenir un contact d'ordre k avec l'espèce visée, la version optimisée produit beaucoup moins de structures inutiles, c'est-à-dire portées par une cardinalité supérieure à k. Il est difficile d'observer ce phénomène sur les exemples des arbres généraux et des graphes série-parallèle, car le contact augmente très vite. Nous serons mieux à même de l'illustrer au niveau des itérations sur les séries que nous verrons dans le prochain chapitre. Cette optimisation permettra en particulier d'améliorer d'un facteur constant l'efficacité du calcul des coefficients des séries et également des valeurs numériques (chapitre 7).

Chapitre 6

Itérations sur les séries génératrices de dénombrement

Sommaire

Comme pour les classes combinatoires présentées dans la première partie de ce mémoire, on peut associer à chaque espèce de structures $\mathcal{F}$, une série génératrice exponentielle pour l'énumération des structures étiquetées et une série ordinaire pour les types d'isomorphisme (structures non étiquetées). Une troisième série est associée à $\mathcal{F}$: la série indicatrice des cycles qui est un outil plus général contenant à elle seule plus d'information que les séries ordinaires et exponentielles. Les opérations combinatoires définies sur les espèces de structures peuvent être relevées au niveau des séries de dénombrement. Ainsi les systèmes d'équations combinatoires qui décrivent récursivement certaines espèces de structures se traduisent en systèmes d'équations définissant leurs séries génératrices associées. Les itérations combinatoires des chapitres précédents sont transposées au niveau des séries formelles et donnent lieu à des algorithmes efficaces pour calculer les suites de dénombrement. Cette approche est validée par le fait que ces itérations "héritent" des propriétés que nous avons démontrées au niveau des structures combinatoires, notamment la croissance et la vitesse de convergence.

La figure 1 représente les approximations des séries successivement obtenues par itération simple et par itération de Newton, pour les arbres généraux planaires (dont la série $Y(z)$ est indiquée en bleu). Dans les deux cas, la croissance (le nombre de coefficients corrects augmente à chaque étape) se traduit par le fait que les séries se rapprochent progressivement de la solution. La vitesse de convergence est également visible : après dix étapes, l'itération simple n'atteint même pas la précision obtenue par l'itération de Newton en trois étapes.

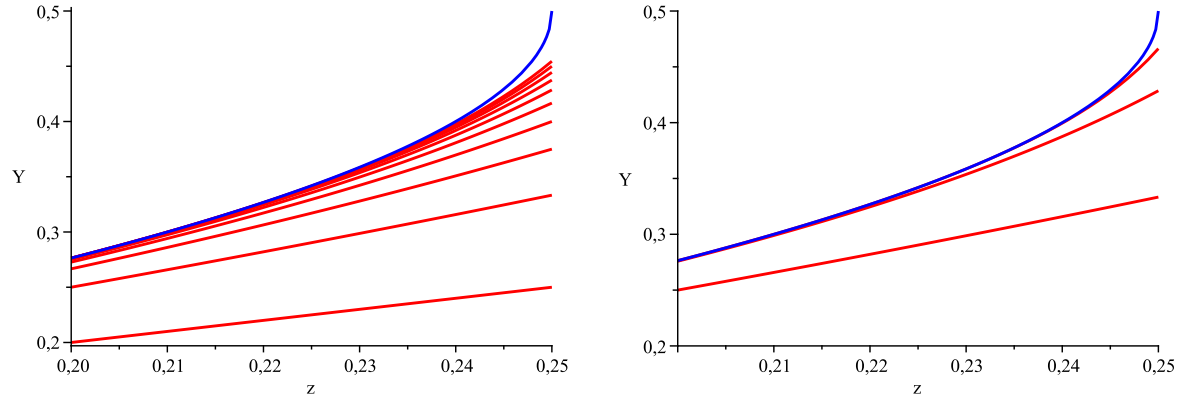


FIG. 1 – Itérations (simple à gauche, et de Newton à droite) sur les séries formelles des arbres généraux planaires.

Dans l'optique de notre problématique initiale qu'est la génération aléatoire de structures combinatoires, on peut se ramener au cadre des structures décomposables (voir définition 1, page 21), et alors garantir une bonne complexité pour les itérations de Newton sur les séries génératrices. Dans ce chapitre, nous démontrons et illustrons le résultat suivant :
Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ *une spécification combinatoire satisfaisant les hypothèses du théorème des espèces implicites de Joyal (voir page 87). L'itération suivante converge quadratiquement vers la solution du système fonctionnel* $\boldsymbol{Y}(z) = \boldsymbol{H}(z, \boldsymbol{Y}(z))$ *:*

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{Y}^{[n]}(z) + \boldsymbol{U}^{[n+1]}(z)\Big(\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)\Big), \qquad \boldsymbol{Y}^{[0]}(z) = \mathbf{0},$$

$$\boldsymbol{U}^{[n+1]}(z) = \boldsymbol{U}^{[n]}(z) + \boldsymbol{U}^{[n]}(z)\Big(\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\boldsymbol{U}^{[n]}(z) - (\boldsymbol{U}^{[n]}(z) - \mathbf{Id})\Big), \quad \boldsymbol{U}^{[0]}(z) = \mathbf{Id}.$$

Les algorithmes qui en résultent permettent de calculer les N *premiers termes du vecteur de séries* $\boldsymbol{Y}(z)$ *avec une complexité quasi-optimale, aussi bien en nombre d'opérations arithmétiques, qu'en termes d'opérations binaires.*

Plus qu'une simple étape dans notre démonstration de la validité de l'oracle de Boltzmann, l'itération de Newton pour les séries génératrices est donc également un algorithme efficace pour calculer les coefficients de ces séries ; cela permet notamment d'améliorer la complexité du pré-traitement pour la méthode récursive (voir page 25).

1 Séries formelles associées aux espèces

1.1 Séries exponentielles et ordinaires

Dans cette section, nous présentons les définitions des séries de dénombrement qui sont données dans [BLL98], en modifiant légèrement les notations pour qu'elles soient cohérentes avec celles utilisées dans les chapitres 1 à 3.

Définition 25. *La* série génératrice (exponentielle) *de l'espèce de structures* $\mathcal{F}$ *est :*

$$\widehat{F}(z) = \sum_{n=0}^{\infty} \hat{f}_n \frac{z^n}{n!},$$

où $\hat{f}_n = n![z^n]\widehat{F}(z)$ *est le nombre de* $\mathcal{F}$*-structures portées par un ensemble à* n *éléments.*

Une $\mathcal{F}$-structure est étiquetée par les éléments de son ensemble sous-jacent. Par opposition, une $\mathcal{F}$-structure γ est *non étiquetée* si l'on oublie la nature des éléments qui composent son ensemble sous-jacent ; autrement dit, γ est un type d'isomorphisme de $\mathcal{F}$-structure. On dira qu'une structure non étiquetée est d'*ordre* n si elle est le type d'isomorphisme d'une structure portée par un ensemble de cardinalité n. Les séries génératrices des types permettent de dénombrer les structures non étiquetées.

Définition 26. *La* série génératrice (ordinaire) des types d'isomorphisme *de l'espèce $\mathcal{F}$ est :*

$$F(z) = \sum_{n=0}^{\infty} f_n z^n,$$

où $f_n = [z^n]F(z)$ est le nombre de $\mathcal{F}$-structures non étiquetées d'ordre n.

1.2 Séries indicatrices des cycles

En général, calculer explicitement ou récursivement la série génératrice des types d'une espèce $\mathcal{F}$ est un problème difficile. En plus des opérations combinatoires sur les espèces, cela requiert l'utilisation d'un troisième genre de série associée à $\mathcal{F}$, sa série indicatrice des cycles, notée $Z_{\mathcal{F}}$. C'est une série formelle en une infinité de variables $z_1, z_2, z_3, \dots$; elle contient à elle seule plus d'information que les séries $\widehat{F}(z)$ et $F(z)$ réunies. Pour pouvoir la définir, on commence par introduire le *type de cycle* d'une permutation.

Définition 27. *Soit U un ensemble fini et σ une permutation de U. Le* type de cycle *de la permutation σ est une suite $(\sigma_1, \sigma_2, \sigma_3, \dots)$ telle que pour $k \geq 1$, σ_k est le nombre de cycles de longueur k dans la décomposition en cycles de σ.*

Définition 28. *La* série indicatrice des cycles *d'une espèce $\mathcal{F}$ est la série formelle en une infinité de variables $z_1, z_2, z_3, \dots$:*

$$Z_{\mathcal{F}}(z_1, z_2, z_3, \dots) = \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{\sigma \in \mathcal{S}[n]} f_\sigma z_1^{\sigma_1} z_2^{\sigma_2} z_3^{\sigma_3} \cdots$$

où $\mathcal{S}[n]$ est le groupe des permutations de $\{1, 2, \dots, n\}$, la suite $(\sigma_1, \sigma_2, \sigma_3, \dots)$ est le type de cycle de σ et f_σ est le nombre de $\mathcal{F}$-structures portées par la cardinalité n et qui sont inchangées par $\mathcal{F}[\sigma]$, i.e., *pour lesquelles σ est un automorphisme.*

Exemple 26. Pour les espèces 0, 1, $\mathcal{Z}$, L (ordres linéaires), E (ensembles) et C (ordres cycliques) que nous avons vues au cours des chapitres précédents, on a :

$$\begin{aligned}
Z_0(z_1, z_2, z_3, \dots) &= 0, \quad Z_1(z_1, z_2, z_3, \dots) = 1, \quad Z_{\mathcal{Z}}(z_1, z_2, z_3, \dots) = z_1, \\
Z_L(z_1, z_2, z_3, \dots) &= \frac{1}{1 - z_1} \\
Z_E(z_1, z_2, z_3, \dots) &= \exp(z_1 + \frac{z_2}{2} + \frac{z_3}{3} + \cdots) \\
Z_C(z_1, z_2, z_3, \dots) &= \sum_{k \geq 1} \frac{\varphi(k)}{k} \log \frac{1}{1 - z_k}
\end{aligned}$$

Le théorème suivant nous permet de retrouver les séries exponentielles et ordinaires d'une espèce $\mathcal{F}$ à partir de sa série indicatrice des cycles.

Théorème 6 ([BLL98])**.** *Quelle que soit l'espèce de structure $\mathcal{F}$, on a :*

$$\widehat{F}(z) = Z_{\mathcal{F}}(z, 0, 0, \dots) \quad \text{et} \quad F(z) = Z_{\mathcal{F}}(z, z^2, z^3, \dots).$$

1.3 Opérations sur les séries

Les opérations combinatoires sur les espèces qui ont été définies au chapitre 4 se traduisent en opérations sur les trois genres de séries que nous venons de décrire. Dans le cas des séries exponentielles, l'addition, la multiplication, la substitution et la dérivation des espèces correspondent aux mêmes opérations sur les séries. Le dénombrement des types de structures est un problème plus difficile ; les opérations de substitution et de dérivation sur les séries génératrices de type se font par le biais des séries indicatrices des cycles. Ces opérations sont résumées par la table 1. De ces opérations et du théorème 6, on peut déduire les séries génératrices présentées dans la table 1 du chapitre 1 (page 22).

	$\widehat{H}(z)$	$H(z)$	$Z_{\mathcal{H}}(z_1, z_2, \dots)$
$\mathcal{H} = \mathcal{F} + \mathcal{G}$	$\widehat{F}(z) + \widehat{G}(z)$	$F(z) + G(z)$	$Z_{\mathcal{F}}(z_1, z_2, \dots) + Z_{\mathcal{G}}(z_1, z_2, \dots)$
$\mathcal{H} = \mathcal{F} \cdot \mathcal{G}$	$\widehat{F}(z) \cdot \widehat{G}(z)$	$F(z) \cdot G(z)$	$Z_{\mathcal{F}}(z_1, z_2, \dots) \cdot Z_{\mathcal{G}}(z_1, z_2, \dots)$
$\mathcal{H} = \mathcal{F} \circ \mathcal{G}$	$\widehat{F}(\widehat{G}(z))$	$Z_{\mathcal{F}}(G(z), G(z^2), \dots)$	$Z_{\mathcal{F}}(Z_{\mathcal{G}}(z_1, z_2, \dots), Z_{\mathcal{G}}(z_2, z_4, \dots), \dots)$
$\mathcal{H} = \partial\mathcal{F}/\partial\mathcal{Z}$	$\frac{d}{dz}\widehat{F}(z)$	$\frac{\partial Z_{\mathcal{F}}}{\partial z_1}(z, z^2, \dots)$	$\frac{\partial Z_{\mathcal{F}}}{\partial z_1}(z_1, z_2, \dots)$

TAB. 1 – Opérations sur les séries formelles associées aux espèces de structures.

2 Transfert de convergence

Nous établissons ici la première partie du transfert de convergence qui relève les itérations combinatoires au niveau des séries formelles.

2.1 Convergence des itérations

La valuation d'une série formelle $S(z)$, notée $\mathrm{val}(S(z))$ est l'indice de son premier terme non nul. Classiquement, on définit la distance entre deux séries formelles $F(z)$ et $G(z)$ par :

$$d\big(F(z), G(z)\big) = 2^{-\mathrm{val}(F(z)-G(z))}.$$

De cette notion de distance, découle la définition classique de convergence des suites de séries formelles. La convergence des vecteurs de séries, ainsi que la distance et la valuation, sont définies composante par composante.

Définition 29. *On dit qu'une suite de séries formelles* $(Y^{[n]}(z))_{n\in\mathbb{N}}$ converge *vers la série* $Y(z)$*, ce que l'on note :*

$$\lim_{n\to\infty} Y^{[n]}(z) = Y(z),$$

si et seulement si

$$\forall\, \varepsilon > 0, \quad \exists N \geq 0, \quad \forall\, n \in \mathbb{N}, \quad n \geq N \;\Rightarrow\; d\big(Y^{[n]}(z), Y(z)\big) < \varepsilon.$$

Comme les coefficients des séries génératrices sont les suites de dénombrement des structures combinatoires associées, on peut traduire la notion de contact combinatoire au niveau des séries.

Définition 30. *Étant données deux séries formelles $F(z) = \sum_{n\geq 0} f_n z^n$ et $G(x) = \sum_{n\geq 0} g_n z^n$, on dit que $F(z)$ et $G(z)$ ont un* contact *d'ordre k, que l'on note $F(z) =_k G(z)$, si pour tout $i \leq k$, on a $[z^i]F(z) = [z^i]G(z)$. En d'autres termes, en notant $F_{\leq k}(z) = \sum_{n=0}^{k} f_n z^n$, on a :*

$$F(z) =_k G(z) \quad \Leftrightarrow \quad F_{\leq k}(z) = G_{\leq k}(z).$$

Le lemme suivant fait le lien entre le contact des espèces et le contact de leurs séries génératrices associées.

Lemme 11. *Soient $\widehat{F}(z)$ et $\widehat{G}(z)$ les séries génératrices exponentielles et $F(z)$ et $G(z)$ les séries ordinaires des espèces $\mathcal{F}$ et $\mathcal{G}$. Quel que soit l'entier positif k, un contact d'ordre k entre $\mathcal{F}$ et $\mathcal{G}$ implique un contact d'ordre k entre leurs séries génératrices associées :*

$$\mathcal{F} =_k \mathcal{G} \quad \Rightarrow \quad \widehat{F}(z) =_k \widehat{G}(z) \quad \text{et} \quad F(z) =_k G(z).$$

La valuation de la différence des séries est alors supérieure à k et leur distance inférieure à 2^{-k}.

Démonstration. L'implication est une conséquence directe de la définition de contact entre les séries. On en déduit que, pour tout i tel que $0 \leq i \leq k$,

$$[z^i]F(z) = [z^i]G(z) \quad \Leftrightarrow \quad [z^i](F(z) - G(z)) = 0,$$

ce qui signifie que $\mathrm{val}(\widehat{F}(z) - \widehat{G}(z)) > k$, et de même pour les séries exponentielles. La distance s'ensuit. □

Nous pouvons maintenant établir le transfert de convergence des itérations combinatoires vers les itérations sur les séries.

Lemme 12 (Transfert, première partie)**.** *Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ un système tel que $\boldsymbol{\mathcal{H}}(0, \mathbf{0}) = \mathbf{0}$ et $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(0, \mathbf{0})$ est nilpotente. Soit $\boldsymbol{\mathcal{F}}$ une famille d'espèces ayant pour séries génératrices exponentielles $\widehat{\boldsymbol{F}}$ et pour séries ordinaires $\boldsymbol{F}$. Si l'itération combinatoire*

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}), \qquad \text{avec } \boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0},$$

converge vers la famille d'espèces $\boldsymbol{\mathcal{Y}}$, solution de $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$, alors l'itération :

$$\widehat{\boldsymbol{Y}}^{[n+1]}(z) = \widehat{\boldsymbol{F}}(z, \widehat{\boldsymbol{Y}}^{[n]}(z)), \qquad \text{avec } \widehat{\boldsymbol{Y}}^{[0]}(z) = \mathbf{0},$$

converge vers le vecteur de séries génératrices exponentielles $\widehat{\boldsymbol{Y}}(z)$ de la famille d'espèces $\boldsymbol{\mathcal{Y}}$. De même, l'itération :

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{F}(z, \boldsymbol{Y}^{[n]}(z)), \qquad \text{avec } \boldsymbol{Y}^{[0]}(z) = \mathbf{0},$$

converge vers le vecteur de séries génératrices ordinaires $\boldsymbol{Y}(z)$ de la famille d'espèces $\boldsymbol{\mathcal{Y}}$.

Démonstration. Par hypothèse, pour tout $k \geq 0$, il existe $N \geq 0$ tel que, pour tout $n \geq N$, on a $\boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}}$; d'après le lemme 11, cela se traduit au niveau des séries par :

$$\forall\, k \geq 0, \quad \exists N \geq 0, \quad \forall\, n \geq N, \quad d\big(\boldsymbol{Y}^{[n]}(z), \boldsymbol{Y}(z)\big) < 2^{-k} \text{ et } d\big(\widehat{\boldsymbol{Y}}^{[n]}(z), \widehat{\boldsymbol{Y}}(z)\big) < 2^{-k}. \quad \square$$

On peut désormais traduire les itérations combinatoires des chapitres précédents en itérations pour calculer les séries génératrices associées aux espèces définies de façon implicite. La proposition 17 résulte de l'application du lemme de transfert 12 à l'itération de Newton optimisée présentée au chapitre précédent. D'après ce qui précède, tout comme l'itération combinatoire, l'itération sur les séries formelles a une vitesse de convergence quadratique.

Proposition 17. *Soit* $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ *une spécification combinatoire telle que* $\boldsymbol{\mathcal{H}}(0, \mathbf{0}) = \mathbf{0}$ *et la matrice* $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(0, \mathbf{0})$ *est nilpotente. Soit* $\boldsymbol{Y}(z) = \boldsymbol{H}(z, \boldsymbol{Y}(z))$ *le système fonctionnel associé sur les séries génératrices (ordinaires ou exponentielles). L'itération suivante converge quadratiquement vers* $\boldsymbol{Y}(z)$ *:*

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{Y}^{[n]}(z) + \boldsymbol{U}^{[n+1]}(z)\Big(\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)\Big) \tag{1}$$

$$\boldsymbol{U}^{[n+1]}(z) = \boldsymbol{U}^{[n]}(z) + \boldsymbol{U}^{[n]}(z)\Big(\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\boldsymbol{U}^{[n]}(z) - (\boldsymbol{U}^{[n]}(z) - \mathbf{Id})\Big), \tag{2}$$

avec $\boldsymbol{Y}^{[0]}(z) = \mathbf{0}$ *et* $\boldsymbol{U}^{[0]}(z) = \mathbf{Id}$.

Démonstration. D'après le lemme 12, la convergence de l'itération combinatoire entraîne la convergence de l'itération sur les séries. Nous avons montré, au chapitre précédent, la convergence quadratique de cette itération :

$$\forall\, k, n \geq 0, \quad \boldsymbol{\mathcal{Y}}^{[n]} =_k \boldsymbol{\mathcal{Y}} \;\Rightarrow\; \boldsymbol{\mathcal{Y}}^{[n+1]} =_{2k+1} \boldsymbol{\mathcal{Y}}.$$

D'après le lemme 11, cela implique que la distance entre les séries est doublée à chaque itération :

$$\forall\, k, n \geq 0, \quad d\big(\boldsymbol{Y}^{[n]}(z), \boldsymbol{Y}(z)\big) < 2^{-k} \Rightarrow d\big(\boldsymbol{Y}^{[n+1]}(z), \boldsymbol{Y}(z)\big) < 2^{-2k-1}. \qquad \square$$

Le même résultat est obtenu avec l'itération de Newton classique. Dans les sections suivantes, nous illustrons le transfert de convergence des itérations combinatoires pour les arbres et les graphes série-parallèle vers les itérations permettant de calculer leurs séries.

2.2 Exemple des arbres généraux planaires

L'espèce des arbres généraux planaires est définie par l'équation :

$$\mathcal{Y} = \mathcal{Z} \times L(\mathcal{Y}).$$

Nous avons vu au chapitre 4, que par itération de cette équation combinatoire, avec comme point de départ l'espèce 0, on peut engendrer l'espèce des arbres généraux. En relevant cette itération au niveau des séries formelles, à l'aide des opérations définies par la table 1 et de la série des ordres linéaires (voir exemple 26), on obtient[35] :

$$Y^{[n+1]}(z) = \frac{z}{1 - Y^{[n]}(z)}, \qquad Y^{[0]}(z) = 0.$$

De la même façon, on peut transposer au niveau des séries l'itération de Newton combinatoire présentée à la section 1.1 du chapitre précédent (page 96) :

$$Y^{[n+1]}(z) = Y^{[n]}(z) + \frac{z/\big(1 - Y^{[n]}(z)\big) - Y^{[n]}(z)}{1 - z/\big(1 - Y^{[n]}(z)\big)^2}, \qquad Y^{[0]}(z) = 0,$$

[35] L'itération pour la série exponentielle est la même, car pour tout n, il y a exactement $n!$ arbres étiquetés ayant le même type d'isomorphisme.

ainsi que l'itération optimisée :

$$\begin{aligned} Y^{[n+1]}(z) &= Y^{[n]}(z) + U^{[n+1]}(z)\big(z/(1-Y^{[n]}(z)) - Y^{[n]}(z)\big), & Y^{[0]}(z) &= 0,\\ U^{[n+1]}(z) &= U^{[n]}(z) + U^{[n]}(z)\big(A^{[n]}(z)U^{[n]}(z) - U^{[n]}(z) + 1\big), & U^{[0]}(z) &= 1,\\ A^{[n]}(z) &= z/(1-Y^{[n]}(z))^2. \end{aligned}$$

Les figures 2 et 3 représentent les cinq premières étapes de chacune de ces itérations. Les termes en gras sont ceux qui coïncident avec les termes de la série génératrice des arbres. Sur ces exemples, on peut observer la convergence linéaire de l'itération simple alors que les itérations de Newton ont une convergence quadratique. Ces séries correspondent aux courbes données par la figure 1, en introduction de ce chapitre. On remarque également que les deux itérations de Newton convergent exactement à la même vitesse : les termes corrects à chaque étape sont les mêmes ; mais les autres coefficients sont plus petits pour l'itération optimisée. C'est précisément la conséquence de notre optimisation ; on passe ainsi moins de temps à calculer des termes qui ne font pas augmenter le contact avec la solution. En comparant ces résultats avec les itérations combinatoires correspondantes, on peut observer qu'à chaque étape, le coefficient $[z^i]Y^{[n]}(z)$ correspond effectivement au nombre d'arbres portés par un ensemble de cardinalité i engendrés à la n-ième étape de l'itération combinatoire.

Itération simple :

$$\begin{aligned} Y^{[0]}(z) &= \mathbf{0} \quad Y^{[1]}(z) = \boldsymbol{z}\\ Y^{[2]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + z^3 + z^4 + z^5 + z^6 + z^7 + z^8 + z^9 + \cdots\\ Y^{[3]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + 4\,z^4 + 8\,z^5 + 16\,z^6 + 32\,z^7 + 64\,z^8 + 128\,z^9 + \cdots\\ Y^{[4]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + 13\,z^5 + 34\,z^6 + 89\,z^7 + 233\,z^8 + 610\,z^9 + \cdots \end{aligned}$$

Fig. 2 – Itération simple pour l'énumération des arbres planaires.

Itération de Newton :

$$\begin{aligned} Y^{[0]}(z) &= \mathbf{0}\\ Y^{[1]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + z^3 + z^4 + z^5 + z^6 + z^7 + z^8 + z^9 + \cdots\\ Y^{[2]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + \mathbf{14}\,\boldsymbol{z^5} + \mathbf{42}\,\boldsymbol{z^6} + 131\,z^7 + 417\,z^8 + 1341\,z^9 \cdots\\ Y^{[3]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + \mathbf{14}\,\boldsymbol{z^5} + \mathbf{42}\,\boldsymbol{z^6} + \mathbf{132}\,\boldsymbol{z^7} + \cdots + \mathbf{742900}\,\boldsymbol{z^{14}} + \cdots\\ Y^{[4]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + \mathbf{14}\,\boldsymbol{z^5} + \mathbf{42}\,\boldsymbol{z^6} + \cdots + \mathbf{1002242216651368}\,\boldsymbol{z^{30}} + \cdots \end{aligned}$$

Itération de Newton optimisée :

$$\begin{aligned} Y^{[0]}(z) &= \mathbf{0}\\ Y^{[1]}(z) &= \boldsymbol{z} + \boldsymbol{z^2}\\ Y^{[2]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + \mathbf{14}\,\boldsymbol{z^5} + \mathbf{42}\,\boldsymbol{z^6} + 110\,z^7 + 264\,z^8 \cdots\\ Y^{[3]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + \mathbf{14}\,\boldsymbol{z^5} + \mathbf{42}\,\boldsymbol{z^6} + \mathbf{132}\,\boldsymbol{z^7} + \cdots + \mathbf{742900}\,\boldsymbol{z^{14}} + \cdots\\ Y^{[4]}(z) &= \boldsymbol{z} + \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{5}\,\boldsymbol{z^4} + \mathbf{14}\,\boldsymbol{z^5} + \mathbf{42}\,\boldsymbol{z^6} + \cdots + \mathbf{1002242216651368}\,\boldsymbol{z^{30}} + \cdots \end{aligned}$$

Fig. 3 – Itérations de Newton pour l'énumération des arbres planaires.

2.3 Exemple des graphes série-parallèle

On rappelle que la famille des graphes série-parallèle est définie par :

$$\begin{cases} \mathcal{Y}_2 &= L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3) \\ \mathcal{Y}_3 &= E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2) \end{cases}.$$

Comme pour les arbres, il est possible de résoudre ce système combinatoire par itération, et toute itération qui remplit les conditions du lemme de transfert 12 peut se traduire en itération sur les séries de dénombrement des graphes série-parallèle. La première étape est de réécrire ces itérations en termes d'opérations sur les séries. Si $F(z)$ est la série ordinaire de l'espèce $\mathcal{F}$, alors la série de l'espèce des ensembles de $\mathcal{F}$-structures est :

$$\overline{\mathrm{Exp}}\,(F(z)) = \exp\left(\sum_{k\geq 1} \frac{F(z^k)}{k}\right).$$

En utilisant ces notations, on peut définir l'itération simple sur les graphes séries-parallèle non étiquetés à partir de l'itération combinatoire présentée au chapitre 4 (page 93) :

$$\begin{cases} Y_2^{[n+1]}(z) &= \frac{(z+Y_3^{[n]}(z))^2}{1-z-Y_3^{[n]}(z)} \\ Y_3^{[n+1]}(z) &= \overline{\mathrm{Exp}}\left(z + Y_2^{[n]}(z)\right) - z - Y_2^{[n]}(z) - 1. \end{cases} \tag{3}$$

Tout comme pour les arbres, cette itération a une convergence typiquement linéaire. Là encore, on peut obtenir une convergence plus rapide avec l'itération de Newton. Cette dernière est obtenue en traduisant directement l'itération combinatoire suivante :

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{Y}}^{[n]} + \begin{pmatrix} 1 & 1 - L(\mathcal{Z} + \mathcal{Y}_3^{[n]})^2 \\ 1 - E(\mathcal{Z} + \mathcal{Y}_2^{[n]}) & 1 \end{pmatrix}^{-1} \begin{pmatrix} L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3^{[n]}) - \mathcal{Y}_2^{[n]} \\ E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2^{[n]}) - \mathcal{Y}_3^{[n]} \end{pmatrix}$$

en une itération sur les séries formelles :

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{Y}^{[n]}(z) + \Big(\mathbf{Id} - \boldsymbol{A}\big(\boldsymbol{Y}^{[n]}(z)\big)\Big)^{-1} \boldsymbol{C}\big(\boldsymbol{Y}(z)^{[n]}\big), \tag{4}$$

avec

$$\boldsymbol{A}\big(\boldsymbol{Y}(z)\big) = \begin{pmatrix} 0 & 1/(1-z-Y_3(z))^2 - 1 \\ \overline{\mathrm{Exp}}\,(z + Y_2(z)) - 1 & 0 \end{pmatrix}, \tag{5}$$

et

$$\boldsymbol{C}\big(\boldsymbol{Y}(z)\big) = \begin{pmatrix} (z + Y_3(z))^2/(1 - z - Y_3(z)) - Y_2(z) \\ \overline{\mathrm{Exp}}\,(z + Y_2(z)) - z - Y_2(z) - 1 - Y_3(z) \end{pmatrix}.$$

Pour améliorer l'efficacité de cette itération, la matrice inverse se réécrit sous la forme

$$(\mathbf{Id} - \boldsymbol{A}(z))^{-1} = u\big(\boldsymbol{Y}(z)\big)^{-1} \boldsymbol{M}\big(\boldsymbol{Y}(z)\big),$$

avec $u\big(\boldsymbol{Y}(z)\big) = 1 + \overline{\mathrm{Exp}}(z + Y_2(z))\,(z + Y_3(z))\,(z - 2 + Y_3(z))$, et

$$\boldsymbol{M}\big(\boldsymbol{Y}(z)\big) = \begin{pmatrix} (-1 + z + Y_3(z))^2 & -(z + Y_3(z))(z - 2 + Y_3(z)) \\ \big(\overline{\mathrm{Exp}}(z + Y_2(z)) - 1\big)\,(-1 + z + Y_3(z))^2 & (-1 + z + Y_3(z))^2 \end{pmatrix},$$

Itération de Newton, séries exponentielles :

$$\widehat{Y}_2^{[0]}(z) = \mathbf{0} \quad \widehat{Y}_3^{[0]}(z) = \mathbf{0}$$
$$\widehat{Y}_2^{[1]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \frac{29}{6}\,z^4 + \frac{139}{12}\,z^5 + \frac{3337}{120}\,z^6 + \frac{601}{9}\,z^7 + \frac{808243}{5040}\,z^8 + \cdots$$
$$\widehat{Y}_3^{[1]}(z) = \frac{\mathbf{1}}{\mathbf{2}}\,\boldsymbol{z^2} + \frac{\mathbf{7}}{\mathbf{6}}\,\boldsymbol{z^3} + \frac{61}{24}\,z^4 + \frac{721}{120}\,z^5 + \frac{10351}{720}\,z^6 + \frac{173867}{5040}\,z^7 + \frac{667957}{8064}\,z^8 + \cdots$$
$$\widehat{Y}_2^{[2]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \frac{\mathbf{61}}{\mathbf{12}}\,\boldsymbol{z^4} + \frac{\mathbf{29}}{\mathbf{2}}\,\boldsymbol{z^5} + \frac{\mathbf{15961}}{\mathbf{360}}\,\boldsymbol{z^6} + \frac{\mathbf{2841}}{\mathbf{20}}\,\boldsymbol{z^7} + \frac{9484021}{20160}\,z^8 + \cdots$$
$$\widehat{Y}_3^{[2]}(z) = \frac{\mathbf{1}}{\mathbf{2}}\,\boldsymbol{z^2} + \frac{\mathbf{7}}{\mathbf{6}}\,\boldsymbol{z^3} + \frac{\mathbf{73}}{\mathbf{24}}\,\boldsymbol{z^4} + \frac{\mathbf{1051}}{\mathbf{120}}\,\boldsymbol{z^5} + \frac{\mathbf{19381}}{\mathbf{720}}\,\boldsymbol{z^6} + \frac{\mathbf{436087}}{\mathbf{5040}}\,\boldsymbol{z^7} + \frac{11584693}{40320}\,z^8 + \cdots$$
$$\widehat{Y}_2^{[3]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \frac{\mathbf{61}}{\mathbf{12}}\,\boldsymbol{z^4} + \frac{\mathbf{29}}{\mathbf{2}}\,\boldsymbol{z^5} + \frac{\mathbf{15961}}{\mathbf{360}}\,\boldsymbol{z^6} + \cdots + \frac{\mathbf{366558482492939101}}{\mathbf{108972864000}}\,\boldsymbol{z^{15}} + \cdots$$
$$\widehat{Y}_3^{[3]}(z) = \frac{\mathbf{1}}{\mathbf{2}}\,\boldsymbol{z^2} + \frac{\mathbf{7}}{\mathbf{6}}\,\boldsymbol{z^3} + \frac{\mathbf{73}}{\mathbf{24}}\,\boldsymbol{z^4} + \frac{\mathbf{1051}}{\mathbf{120}}\,\boldsymbol{z^5} + \frac{\mathbf{19381}}{\mathbf{720}}\,\boldsymbol{z^6} + \frac{\mathbf{386081655546862081}}{\mathbf{186810624000}}\,\boldsymbol{z^{15}} + \cdots$$

Itération de Newton, séries ordinaires :

$$Y_2^{[0]}(z) = \mathbf{0} \quad Y_3^{[0]}(z) = \mathbf{0}$$
$$Y_2^{[1]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + 8\,z^4 + 21\,z^5 + 55\,z^6 + 144\,z^7 + 377\,z^8 + 987\,z^9 + 2584\,z^{10} + \cdots$$
$$Y_3^{[1]}(z) = \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + 5\,z^4 + 13\,z^5 + 34\,z^6 + 8\,z^7 + 233\,z^8 + 610\,z^9 + 1597\,z^{10} + \cdots$$
$$Y_2^{[2]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \mathbf{9}\,\boldsymbol{z^4} + \mathbf{30}\,\boldsymbol{z^5} + \mathbf{103}\,\boldsymbol{z^6} + \mathbf{375}\,\boldsymbol{z^7} + 1399\,z^8 + 5365\,z^9 + 20922\,z^{10} + \cdots$$
$$Y_3^{[2]}(z) = \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{6}\,\boldsymbol{z^4} + \mathbf{18}\,\boldsymbol{z^5} + \mathbf{64}\,\boldsymbol{z^6} + \mathbf{227}\,\boldsymbol{z^7} + 855\,z^8 + 3269\,z^9 + 12764\,z^{10} + \cdots$$
$$Y_2^{[3]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \mathbf{9}\,\boldsymbol{z^4} + \mathbf{30}\,\boldsymbol{z^5} + \mathbf{103}\,\boldsymbol{z^6} + \mathbf{375}\,\boldsymbol{z^7} + \mathbf{1400}\,\boldsymbol{z^8} + \cdots + \mathbf{23696081}\,\boldsymbol{z^{15}} + \cdots$$
$$Y_3^{[3]}(z) = \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{6}\,\boldsymbol{z^4} + \mathbf{18}\,\boldsymbol{z^5} + \mathbf{64}\,\boldsymbol{z^6} + \mathbf{227}\,\boldsymbol{z^7} + \mathbf{856}\,\boldsymbol{z^8} + \cdots + \mathbf{14541132}\,\boldsymbol{z^{15}} + \cdots$$

Fig. 4 – Itérations de Newton pour les séries associées aux graphes série-parallèle.

Itération de Newton optimisée, séries ordinaires :

$$Y_2^{[0]}(z) = \mathbf{0}$$
$$Y_3^{[0]}(z) = \mathbf{0}$$
$$Y_2^{[1]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + 6\,z^4 + 10\,z^5 + 15\,z^6 + 21\,z^7 + 28\,z^8 + 36\,z^9 + 45\,z^{10} + \cdots$$
$$Y_3^{[1]}(z) = \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + 3\,z^4 + 4\,z^5 + 5\,z^6 + 6\,z^7 + 7\,z^8 + 8\,z^9 + 9\,z^{10} + \cdots$$
$$Y_2^{[2]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \mathbf{9}\,\boldsymbol{z^4} + \mathbf{30}\,\boldsymbol{z^5} + \mathbf{103}\,\boldsymbol{z^6} + \mathbf{375}\,\boldsymbol{z^7} + 1349\,z^8 + 4683\,z^9 + 15404\,z^{10} + \cdots$$
$$Y_3^{[2]}(z) = \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{6}\,\boldsymbol{z^4} + \mathbf{18}\,\boldsymbol{z^5} + \mathbf{64}\,\boldsymbol{z^6} + \mathbf{227}\,\boldsymbol{z^7} + 820\,z^8 + 2813\,z^9 + 9141\,z^{10} + \cdots$$
$$Y_2^{[3]}(z) = \boldsymbol{z^2} + \mathbf{3}\,\boldsymbol{z^3} + \mathbf{9}\,\boldsymbol{z^4} + \mathbf{30}\,\boldsymbol{z^5} + \mathbf{103}\,\boldsymbol{z^6} + \mathbf{375}\,\boldsymbol{z^7} + \mathbf{1400}\,\boldsymbol{z^8} + \cdots + \mathbf{23696081}\,\boldsymbol{z^{15}} + \cdots$$
$$Y_3^{[3]}(z) = \boldsymbol{z^2} + \mathbf{2}\,\boldsymbol{z^3} + \mathbf{6}\,\boldsymbol{z^4} + \mathbf{18}\,\boldsymbol{z^5} + \mathbf{64}\,\boldsymbol{z^6} + \mathbf{227}\,\boldsymbol{z^7} + \mathbf{856}\,\boldsymbol{z^8} + \cdots + \mathbf{14541132}\,\boldsymbol{z^{15}} + \cdots$$

Fig. 5 – Itération de Newton optimisée pour les séries ordinaires des graphes série-parallèle.

ainsi, il n'est plus nécessaire d'inverser une matrice de séries mais seulement la série $u\big(\boldsymbol{Y}(z)\big)$. En utilisant cette même astuce, on obtient directement l'itération de Newton optimisée suivante :

$$\begin{aligned} \boldsymbol{Y}^{[n+1]}(z) &= \boldsymbol{Y}^{[n]}(z) + U^{[n+1]}(z)\boldsymbol{M}\big(\boldsymbol{Y}^{[n]}(z)\big)\boldsymbol{C}^{[n+1]}(z) && \boldsymbol{Y}^{[0]}(z) = \mathbf{0} \\ U^{[n+1]}(z) &= U^{[n]}(z) + U^{[n]}(z)\big(u\big(\boldsymbol{Y}^{[n]}(z)\big)U^{[n]}(z) - U^{[n]}(z) + 1\big), && U^{[0]}(z) = 1 \end{aligned}$$

Les itérations pour les graphes série-parallèle étiquetés sont identiques, à ceci près que l'on remplace $\overline{\text{Exp}}$ par exp. Le passage entre les deux types de séries est possible car l'itération sur les séries est une traduction de l'itération sur les espèces. On peut donc indifféremment transformer l'itération combinatoire en une itération sur des séries exponentielles ou ordinaires.

Les figures 4 et 5 représentent les quatre premières étapes de l'itération de Newton pour ces graphes (étiquetés ou non), ainsi que l'itération de Newton optimisée, pour les graphes non étiquetés. Comme précédemment, les termes en gras indiquent le contact avec la solution.

3 Des espèces de structures aux classes combinatoires spécifiables

Nous avons prouvé et illustré le transfert de convergence qui s'opère des itérations combinatoires vers les séries génératrices et nous utiliserons ce résultat au chapitre suivant pour valider notre oracle numérique.

Nous allons maintenant terminer ce chapitre en utilisant l'itération de Newton optimisée de la proposition 17 comme un algorithme efficace pour le calcul des *coefficients* des séries génératrices. En tout généralité, cette itération est effectivement un algorithme, dès que l'on sait traduire les systèmes d'équations sur les espèces en systèmes sur leurs séries. Cependant, pour calculer précisément la complexité d'un tel algorithme, il faut fixer l'ensemble des opérations effectuées sur les séries ; c'est pourquoi, nous repassons désormais dans l'univers des classes combinatoires spécifiables (voir chapitre 1, page 21) considérées dans la première partie de ce mémoire.

Le résultat de convergence sur les séries (lemme 12) s'applique en particulier dans le cas des séries génératrices associées aux classes décomposables qui sont des espèces de structures définies par des systèmes d'équations combinatoires :

- remplissant les conditions du théorème des espèces implicites,
- n'utilisant que les opérations d'addition, de multiplication et de substitution,
- et ne faisant intervenir que les espèces 1, $\mathcal{Z}$, L, E et C.

Avec les notations utilisées pour les classes spécifiables, on a :

$$1 \equiv \mathcal{E}, \quad L \equiv \textsc{Seq}(\mathcal{Z}), \quad E \equiv \textsc{Set}(\mathcal{Z}) \quad \text{et} \quad C \equiv \textsc{Cyc}(\mathcal{Z}).$$

La table 2 rappelle les séries génératrices associées aux différentes constructions combinatoires admissibles (c'est une version simplifiée de la table 1, page 22). La construction d'ensemble étiqueté ne permet pas les répétitions (puisque tous les atomes sont différents), par contre, lorsque l'on oublie les étiquettes, on obtient un multi-ensemble. On peut également ajouter à cette table la construction d'ensemble non étiqueté sans répétitions, notée PSet. La série ordinaire associée à la classe $\textsc{PSet}(\mathcal{A})$ est :

$$\exp\Big(\sum_{k\geq 0}(-1)^{k-1}\frac{A(z^k)}{k}\Big).$$

$\mathcal{F}$	$\mathcal{E}$	$\mathcal{Z}$	$\mathcal{A}+\mathcal{B}$	$\mathcal{AB}$	Seq($\mathcal{A}$)	Set($\mathcal{A}$) (MSet)	Cyc($\mathcal{A}$)
$\widehat{F}(z)$	1	z	$\hat{A}(z)+\widehat{B}(z)$	$\hat{A}(z)\widehat{B}(z)$	$\dfrac{1}{1-\hat{A}(z)}$	$\exp(\hat{A}(z))$	$\log\dfrac{1}{1-\hat{A}(z)}$
$F(z)$	1	z	$A(z)+B(z)$	$A(z)B(z)$	$\dfrac{1}{1-A(z)}$	$\exp\left(\displaystyle\sum_{k\geq 0}\frac{A(z^k)}{k}\right)$	$\displaystyle\sum_{k\geq 0}\frac{\varphi(k)}{k}\log\frac{1}{1-A(z^k)}$

Tab. 2 – Séries génératrices associées aux constructions combinatoires.

Les itérations de Newton sur les séries formelles sont obtenues par traduction des itérations combinatoires. La dérivation qui apparaît dans l'opérateur de Newton peut être traitée au niveau des classes combinatoires – et non pas au niveau des séries – puisque ces classes s'expriment elles-mêmes en termes de constructions admissibles ; la table 3 résume l'ensemble des classes dérivées que l'on obtient à partir des constructions admissibles. La liste des séries données par la table 2 nous permet donc de traduire les itérations de Newton combinatoires associées à l'ensemble des classes spécifiables, en itérations sur leurs séries génératrices. Ce dictionnaire nous a, par exemple, permis de traduire les itérations combinatoires pour les arbres planaires et les circuits série-parallèle de la section précédente. Cela évite en particulier d'avoir à calculer les séries indicatrices des cycles (voir le calcul de la matrice jacobienne (5), par exemple).

$\mathcal{F}$	$\mathcal{E}$	$\mathcal{Z}$	$\mathcal{Y}$	$\mathcal{A}+\mathcal{B}$	$\mathcal{AB}$	Seq($\mathcal{A}$)	Set($\mathcal{A}$)	Cyc($\mathcal{A}$)
$\mathcal{F}'=\partial\mathcal{F}/\partial\mathcal{Y}$	0	0	1	$\mathcal{A}'+\mathcal{B}'$	$\mathcal{A}'\mathcal{B}+\mathcal{A}\mathcal{B}'$	Seq($\mathcal{A}$) $\mathcal{A}'$ Seq($\mathcal{A}$)	$\mathcal{A}'$ Set($\mathcal{A}$)	$\mathcal{A}'$ Seq($\mathcal{A}$)

Tab. 3 – Règles de dérivation des constructions combinatoires.

4 Complexités

Cette section est consacrée à l'analyse de la complexité de l'itération de Newton comme algorithme permettant d'évaluer les N premiers termes des séries génératrices associées aux systèmes combinatoires implicites.

Dans un premier temps, nous considérons la complexité arithmétique. Nous établissons les équations de récurrence qui gouvernent la complexité des itérations de Newton (classique et optimisée) et en déduisons que leur complexité est quasi-optimale, c'est-à-dire $O(N\log N)$ en utilisant une méthode à base de FFT pour multiplier les séries tronquées (cf .[vzGG99]). Bien que les complexités de ces deux itérations soient du même ordre, nous montrons ensuite que notre optimisation permet de gagner un facteur constant sur le calcul des séries.

Nous étudions enfin la complexité binaire de l'itération de Newton optimisée, en prenant en compte la taille des coefficients des séries, afin de montrer qu'elle est également quasi-optimale.

4.1 Complexité arithmétique

L'étude de la complexité arithmétique de l'itération de Newton se fait en deux temps. Une partie de cette complexité est propre au système considéré : c'est le coût de l'évaluation, à la $(n+1)$-ième étape de l'itération, de $\boldsymbol{H}(z,\boldsymbol{Y}^{[n]}(z))$ et $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z,\boldsymbol{Y}^{[n]}(z))$.

Dans un deuxième temps, nous analysons la complexité issue de l'itération de Newton proprement dite, en utilisant des récurrences de type "diviser pour régner".

La complexité du calcul de $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ et $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ dépend de l'algorithme choisi pour multiplier des séries à une précision donnée, c'est pourquoi nous donnons les estimations de complexité en fonction de $\mathsf{M}(N)$, une fonction de multiplication abstraite qui donne une borne supérieure sur le nombre d'opérations arithmétiques utilisées pour le produit de deux polynômes de degré N (ou de séries tronquées à l'ordre N). Par exemple, l'algorithme de Karatsuba a une complexité $O(N^{\log_2 3})$ et la FFT est en $O(N \log N)$.

Par ailleurs, le calcul de l'itération de Newton fait intervenir des produits de matrices de séries tronquées ; nous utiliserons donc également la fonction $\mathsf{MM}(m, N)$ qui correspond au nombre de multiplications nécessaires au produit de matrices $m \times m$ dont les éléments sont des polynômes de degré N. Classiquement, on suppose que :

$$\mathsf{MM}(m, 2N) \geq 2\mathsf{MM}(m, N). \tag{6}$$

D'après [BS05], on a :

$$\mathsf{MM}(m, N) = O(m^\omega N + m^2 \mathsf{M}(N)), \quad \text{avec } \omega \leq 3.$$

Dans ce qui suit, nous estimons la complexité des itérations de Newton en fonction de la précision des séries et non de la taille des systèmes considérés. Nous pourrons donc supposer que les matrices manipulées sont de taille constante et que par conséquent $\mathsf{MM}(m, N) = O(\mathsf{M}(N))$.

Les notations, ainsi que certaines techniques de preuve utilisées dans cette section et la suivante, sont empruntées à [BCG+07]. On pourra se rapporter à ce document pour des références plus précises.

4.1.1 Calcul de $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ et $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$

En toute généralité, la composition de séries n'a pas une complexité quasi-optimale (voir [BK78]). En revanche, dans le cas particulier des séries génératrices associées aux systèmes combinatoires, les constructions utilisées sur les structures conduisent à des compositions de séries génératrices qui ne font intervenir que des opérations dont la complexité est en $O(\mathsf{M}(N))$. Ce gain se répercute ensuite sur l'itération de Newton : sa complexité est alors quasi-optimale.

Proposition 18. *Soient $\mathcal{A}$ et $\mathcal{B}$ des classes combinatoires. Le calcul des N premiers termes des séries génératrices ordinaires et exponentielles associées aux classes :*

$$\mathcal{A} + \mathcal{B}, \quad \mathcal{A} \times \mathcal{B}, \quad \textsc{Seq}(\mathcal{A}), \quad \textsc{MSet}(\mathcal{A}), \quad \textsc{PSet}(\mathcal{A}) \quad \textit{et} \quad \textsc{Cyc}(\mathcal{A}),$$

à partir des séries de $\mathcal{A}$ et $\mathcal{B}$, nécessite $O(\mathsf{M}(N))$ opérations dans le pire cas.

Démonstration. Dans le cas des structures étiquetées, les seules opérations utilisées sur les séries sont, pour $f(z)$ et $g(z)$ fixées :

$$f(z) + g(z), \quad f(z) \cdot g(z), \quad \exp(f(z)), \quad 1/f(z) \quad \text{et} \quad \log(f(z)).$$

La complexité pour une approximation à précision N est alors $O(\mathsf{M}(N))$ pour chacune de ces opérations (cf. [BCS97]).

Dans le cas des structures non étiquetées, les constructions d'ensemble et de cycle impliquent des opérateurs de Pólya :

$$\exp\left(\sum_{k=1}^{\infty} \frac{1}{k} A(z^k)\right), \quad \exp\left(\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} B(z^k)\right) \quad \text{et} \quad \sum_{k=1}^{\infty} \frac{\varphi(k)}{k} \log \frac{1}{1 - A(z^k)}.$$

Pour calculer N termes d'une série correspondant à un opérateur de Pólya, nous devons sommer des séries tronquées ayant respectivement N, $N/2$, $N/3, \ldots$ termes. Le calcul de ces séries à précision N nécessite donc $O(\mathsf{M}(N) + N \log N) = O(\mathsf{M}(N))$ opérations. □

On en déduit que la complexité arithmétique pour calculer les N premiers termes du vecteur de séries $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ est $O(\mathsf{M}(N))$. De même, le calcul des N premiers termes de la matrice $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ a une complexité $O(\mathsf{M}(N))$, puisque par dérivation, on se ramène au même ensemble d'opérations que pour $\boldsymbol{H}$.

4.1.2 Complexité des itérations de Newton

Dans cette section, on montre que le calcul par itération de Newton des N premiers termes des séries génératrices ordinaires issues de systèmes combinatoires a une complexité $O(\mathsf{M}(N))$ pour N grand[36] ; le même résultat reste valable dans le cas des séries exponentielles. Avec une multiplication de séries tronquées à base de FFT par exemple, on peut calculer les N premiers termes des séries avec une complexité $O(N \log N)$. On améliore ainsi la complexité en $O(N \log^2 N)$ obtenue par van der Hoeven [vdH02, §4.5].

Les deux propositions qui suivent donnent les équations de récurrence qui permettront par la suite d'estimer la complexité des deux versions de l'itération de Newton (classique et optimisée). Nous utiliserons la notation suivante pour indiquer la précision des séries manipulées ; étant donné une série $Y(z) = \sum_{n \geq 0} y_n z^n$, on pose :

$$Y(z) \mod z^N := \sum_{n=0}^{N} y_n z^n.$$

Proposition 19. *La complexité $C(N)$ de l'itération de Newton classique :*

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{Y}^{[n]}(z) + \boldsymbol{U}^{[n+1]}(z)\Big(\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)\Big) \mod z^N, \tag{7}$$

avec

$$\boldsymbol{U}^{[n+1]}(z) = \Big(\mathbf{1} - \boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\Big)^{-1},$$

pour calculer les N premiers termes des séries associées au système $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ satisfait la relation suivante, pour $N = 2^n$:

$$C(N) = C(N/2) + K\mathsf{M}(N) + R(N) + O(N), \tag{8}$$

où K est une constante qui dépend de $\boldsymbol{H}$ et $R(N)$ est la complexité du calcul des N premiers termes de la matrice $\boldsymbol{U}^{[n+1]}(z)$.

Démonstration. La convergence quadratique de l'itération de Newton implique qu'à l'étape $n+1$, on calcule $\boldsymbol{Y}(z)$ à précision $N = 2^n$. Pour calculer $\boldsymbol{Y}^{[n+1]}(z)$, on commence par calculer $\boldsymbol{Y}^{[n]}(z)$ à précision $N/2$; le coût de ce calcul correspond au terme $C(N/2)$. Il reste ensuite à calculer le coût de la $(n+1)$-ième étape de l'itération. Le terme $K\mathsf{M}(N)$ provient, d'une part, du calcul du vecteur $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ et de la matrice $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ et, d'autre part, de la multiplication de la matrice $\boldsymbol{U}^{[n+1]}(z)$ et du vecteur $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)$, le tout à précision N. S'ajoute à cela, la complexité du calcul de l'inverse $\boldsymbol{U}^{[n+1]}(z)$, qui correspond au terme $R(N)$ dans l'équation (8). Enfin, l'addition et la soustraction de vecteurs de séries à précision N contribuent pour $O(N)$ opérations. □

[36] Le nombre d'équations m du système n'est pas pris en compte.

Proposition 20. *La complexité $C^\star(N)$ de l'itération de Newton optimisée :*

$$\boldsymbol{U}^{[n+1]}(z) = \boldsymbol{U}^{[n]}(z) + \boldsymbol{U}^{[n]}(z)\big(\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\boldsymbol{U}^{[n]}(z) - (\boldsymbol{U}^{[n]}(z) - \mathbf{Id})\big) \mod z^N, \quad (9)$$

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{Y}^{[n]}(z) + \boldsymbol{U}^{[n+1]}(z)\big(\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)\big) \mod z^N \quad (10)$$

pour calculer les N premiers termes des séries associées au système $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$, satisfait la relation suivante, pour $N = 2^n$:

$$C^\star(N) = C^\star(N/2) + K\mathsf{M}(N) + R^\star(N) + O(N), \quad (11)$$

où K est une constante qui dépend de $\boldsymbol{H}$ et $R^\star(N)$ est la complexité du calcul des N premiers termes de la matrice $\boldsymbol{U}^{[n+1]}(z)$, à partir de $\boldsymbol{Y}^{[n]}(z)$ et $\boldsymbol{U}^{[n]}(z)$ à précision $N/2$.

Démonstration. Le coût provenant du calcul de $\boldsymbol{U}^{[n]}(z)$ et $\boldsymbol{Y}^{[n]}(z)$ à précision $N/2$ correspond au terme $C^\star(N/2)$. La complexité du calcul de $\boldsymbol{U}^{[n+1]}(z)$ est donnée par $R^\star(N)$, indépendamment du coût du calcul de $\boldsymbol{U}^{[n]}(z)$ et de la dérivation $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$. La constante K est la même que celle de la proposition 19 : le terme $K\mathsf{M}(N)$ englobe les complexités correspondant au calcul de $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ et $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$, ainsi que la multiplication de la matrice $\boldsymbol{U}^{[n+1]}(z)$ par le vecteur $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)$; or, nous avons montré au chapitre précédant, que les $N/2$ premiers termes de $\boldsymbol{U}^{[n+1]}$ coïncident avec ceux de la matrice $\big(\mathbf{1} - \boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\big)^{-1}$. □

À partir des récurrences établies par ces deux propositions, nous pouvons estimer la complexité totale des itérations de Newton.

Théorème 7. *Les itérations de Newton classique et optimisée calculent les N premiers termes de la solution du système $\boldsymbol{Y}(z) = \boldsymbol{H}(z, \boldsymbol{Y}(z))$ avec une complexité $O(\mathsf{M}(N))$ pour $N \to \infty$.*

Démonstration. Dans l'itération optimisée, la complexité $R^\star(N)$ du calcul de $\boldsymbol{U}^{[n+1]}(z)$ à précision N, connaissant $\boldsymbol{U}^{[n]}(z)$ à précision $N/2$ et $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ à précision N, est essentiellement celle des multiplications de matrices utilisées. La multiplication de $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ par $\boldsymbol{U}^{[n]}(z)$ a une complexité $\mathsf{MM}(N)$; la seconde multiplication a une complexité $\mathsf{MM}(N/2)$ car les $N/2$ premiers termes de $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\boldsymbol{U}^{[n]}(z) - (\boldsymbol{U}^{[n]}(z) - \mathbf{Id})$ sont nuls (voir lemme 10, page 107). À cela s'ajoute une addition et une soustraction de matrices de séries à précision $N/2$; on a donc :

$$R^\star(N) = \mathsf{MM}(N) + \mathsf{MM}(N/2) + O(N),$$

Dans l'itération de Newton classique, la complexité $R(N)$ est le coût du calcul de l'intégralité de l'inverse :

$$\Big(\mathbf{Id} - \boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\Big)^{-1},$$

à précision N. Ce calcul peut être fait avec un complexité quasi-optimale par itération de Newton (cf. [vzGG99]), ce qui revient à faire l'itération (9) récursivement. La complexité $R(N)$ est donc donnée par l'équation :

$$R(N) = R(N/2) + \mathsf{MM}(N) + \mathsf{MM}(N/2) + O(N).$$

En supposant que $N = 2^n$, on a alors :

$$R(N) = \mathsf{MM}(N) + 2\sum_{k=1}^{n} \mathsf{MM}(N/2^k) + O(N). \quad (12)$$

D'après l'équation (6), $\mathsf{MM}(2N) \geq 2\mathsf{MM}(N)$; on en déduit que

$$R(N) \leq \mathsf{MM}(N) + 2\sum_{k=1}^{n} \frac{1}{2^k}\mathsf{MM}(N) + O(N) \leq 3\mathsf{MM}(N) + O(N).$$

Pour N grand, le coût d'une multiplication de matrices de séries à précision N est dominé par le coût des produits de séries, on peut donc écrire :

$$R(N) \leq k\mathsf{M}(N) + O(N),$$

où k est une constante qui dépend de $\boldsymbol{H}$. Par un raisonnement similaire, on a :

$$R^\star(N) \leq \frac{3}{2}\mathsf{MM}(N) + O(N) \leq k^\star\mathsf{M}(N) + O(N),$$

où $k^\star$ est une constante qui dépend de $\boldsymbol{H}$. Par application du lemme "diviser pour régner" (voir [BCG+07, vzGG99]), on obtient alors le résultat annoncé pour $C(N)$ et $C^\star(N)$. □

Comparaison des itérations de Newton On peut désormais s'intéresser au gain dû à l'optimisation de l'itération de Newton. Pour cela, on commence par comparer les complexités $R(N)$ et $R^\star(N)$. D'après ce qui précède, on a $R(N) = \mu\mathsf{M}(N) + \nu N$ pour certaines constantes μ et ν ; on réécrit alors $R^\star(N)$ sous la forme :

$$R^\star(N) = R(N) - R(N/2) = \mu\mathsf{M}(N)\big(1 - \mathsf{M}(N/2)/\mathsf{M}(N)\big) + \nu N/2.$$

Le facteur $\mu^\star = (1 - \mathsf{M}(N/2)/\mathsf{M}(N))$ gagné par notre optimisation sur le calcul de l'inverse est donc fonction de l'algorithme utilisé pour la multiplication des séries à précision N. En supposant qu'elle a une complexité $\mathsf{M}(N) = cN^\alpha \log^\beta N$, on a :

$$1 - \frac{\mathsf{M}(N/2)}{\mathsf{M}(N)} = 1 - 2^{-\alpha}\frac{\log^\beta N/2}{\log^\beta N}.$$

Avec une multiplication de polynômes à base de FFT, de complexité $M(N) = O(N \log N)$, notre optimisation permet, à chaque étape de l'itération, de gagner sur le calcul de l'inverse un facteur qui tend vers 2 lorsque N est grand. Enfin, si l'on veut comparer les coûts des itérations complètes, on réécrit l'équation (8) sous la forme :

$$C(N) = C(N/2) + (K+\mu)\mathsf{M}(N) + O(N) = (K+\mu)\big(\mathsf{M}(N) + \mathsf{M}(N/2) + \cdots\big) + kN$$

et l'équation (11) sous la forme :

$$C^\star(N) = (K + \mu\mu^\star)\big(\mathsf{M}(N) + \mathsf{M}(N/2) + \cdots\big) + kN,$$

où k et K sont les mêmes constantes dans les deux cas. On en conclut que :

$$\frac{C^\star(N)}{C(N)} = \frac{K + \mu\mu^\star}{K + \mu},$$

avec $3/4 \leq \mu^\star \leq 2$. Notre optimisation est donc d'autant plus significative que la constante K est petite, c'est-à-dire quand les calculs de $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ et $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ sont peu coûteux.

4.2 Complexité binaire

La complexité arithmétique est un indicateur insuffisant du coût des opérations sur les séries génératrices dont les coefficients croissent exponentiellement vite. Afin d'être plus précis, dans cette section, nous étudions la complexité en nombre d'opérations binaires de l'itération de Newton optimisée. Nous traitons uniquement le cas des séries génératrices ordinaires[37]. On rappelle la définition de l'itération de Newton optimisée :

$$\boldsymbol{U}^{[n+1]}(z) = \boldsymbol{U}^{[n]}(z) + \boldsymbol{U}^{[n]}(z)\Big(\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))\boldsymbol{U}^{[n]}(z) - (\boldsymbol{U}^{[n]}(z) - \mathbf{Id})\Big), \tag{13}$$

$$\boldsymbol{Y}^{[n+1]}(z) = \boldsymbol{Y}^{[n]}(z) + \boldsymbol{U}^{[n+1]}(z)\Big(\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) - \boldsymbol{Y}^{[n]}(z)\Big). \tag{14}$$

On vérifie, dans un premier temps, que toutes les séries intervenant dans l'itération de Newton optimisée ont un rayon de convergence supérieur à celui de la solution recherchée. C'est également vrai pour l'itération de point fixe. On peut observer ce phénomène sur les courbes de la figure 1 (en début de chapitre) qui se rapprochent "par en dessous" de la série génératrice des arbres planaires.

Lemme 13. *Les séries* $\boldsymbol{Y}^{[n]}(z)$, $\boldsymbol{U}^{[n]}(z)$, $\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z))$ *et* $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ *définies par les équations* (14) *et* (13) *ont toutes un rayon de convergence supérieur au rayon de convergence* ρ *de* $\boldsymbol{Y}(z)$.

Démonstration. On montre que cette propriété est vraie par induction sur n. Pour $n = 0$, toutes ces séries ont un rayon de convergence infini. Pour tout $n > 0$, nous avons démontré, au chapitre précédent, les inclusions combinatoires suivantes, pour tout $n \geq 0$:

$$\boldsymbol{\mathcal{Y}}^{[n]} \subset \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}}^{[n]}) \subset \boldsymbol{\mathcal{Y}},$$

ce qui se traduit par

$$[z^k]\boldsymbol{Y}^{[n]}(z) \leq [z^k]\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)) \leq [z^k]\boldsymbol{Y}(z), \quad \forall\, k \geq 0.$$

Par conséquent, les séries $\boldsymbol{Y}^{[n]}(z)$ et $\boldsymbol{H}(\boldsymbol{Y}^{[n]}(z))$ ont un rayon de convergence supérieur à ρ. Par analyticité, le rayon de convergence de $\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z))$ est le même que celui de $\boldsymbol{H}(\boldsymbol{Y}^{[n]}(z))$. Enfin, par hypothèse de récurrence, le rayon de convergence de $\boldsymbol{U}^{[n-1]}(z)$ est supérieur à ρ et donc, par construction, $\boldsymbol{U}^{[n]}(z)$ a un rayon de convergence supérieur à ρ. □

Le lemme suivant nous donne une borne supérieure sur la taille des coefficients des séries génératrices ordinaires.

Lemme 14. *Soit la série* $\sum_{n\geq 0} u_n z^n$ *de rayon de convergence* $\rho > 0$. *Si pour tout* $n \geq 0$, *on a* $u_n \in \mathbb{N}$, *alors*

$$\forall\, c > 1, \quad \exists N, \quad \forall\, n \leq N \;\Rightarrow\; I(n) \leq c\, N \log \frac{1}{\rho},$$

où $I(n)$ *est le nombre de bits nécessaires pour représenter* u_n.

Démonstration. Par définition du rayon de convergence,

$$\limsup_{n\to\infty} u_n^{1/n} = \frac{1}{\rho}.$$

[37] La complexité binaire pour le calcul des séries génératrices exponentielles sera traité dans l'article [PSS08].

Comme u_n est entier, cela entraîne

$$\forall\, c > 1, \quad \exists N_0, \quad \forall\, n > N_0 \quad I(n) < c\, n \log \frac{1}{\rho}.$$

Soit $I_0 := \max_{n \le N_0} \log(u_n)$ et $N = I_0/(c \log(1/\rho))$; on a alors

$$\forall\, c > 1, \quad \forall\, n \le N, \quad I(n) < c\, N \log \frac{1}{\rho}. \qquad \square$$

Les deux lemmes précédents nous assurent que la taille des coefficients de toutes les séries impliquées dans l'itération de Newton est bornée par la taille des coefficients de la solution recherchée.

Proposition 21. *Si les séries $\boldsymbol{Y}(z)$, solutions du système $\boldsymbol{Y}(z) = \boldsymbol{H}(z, \boldsymbol{Y}(z))$ ont un rayon de convergence $\rho > 0$, alors pour tout $c > 1$ et tout $n \in \mathbb{N}$, il existe K tel que, pour $k \ge K$:*

$$[z^k]\boldsymbol{Y}^{[n]}(z),\ [z^k]\boldsymbol{U}^{[n]}(z),\ [z^k]\boldsymbol{H}(z, \boldsymbol{Y}^{[n]}(z)),\ [z^k]\boldsymbol{\partial H}/\boldsymbol{\partial Y}(z, \boldsymbol{Y}^{[n]}(z)) \le cn \log \frac{1}{\rho}.$$

On note $\mathsf{M}_{\mathbb{Z}}(N)$ une fonction qui mesure le coût en nombre d'opérations binaires de la multiplication de deux entiers de taille N. Pour obtenir la complexité binaire de l'itération de Newton, nous devons "injecter" ce coût dans nos récurrences, afin de tenir compte de la taille des coefficients des séries.

Proposition 22. *L'itération de Newton optimisée calcule les N premiers termes de la solution du système $\boldsymbol{Y}(z) = \boldsymbol{H}(z, \boldsymbol{Y}(z))$ avec une complexité :*

$$C^{\star}_{\mathbb{Z}}(N) = O\big(\mathsf{M}_{\mathbb{Z}}(c_0\, N)\mathsf{M}(N)\big),$$

où $c_0 := c \log \frac{1}{\rho}$, pour $N \to \infty$.

Démonstration.

$$C^{\star}_{\mathbb{Z}}(N) = C^{\star}_{\mathbb{Z}}(N/2) + K\mathsf{M}_{\mathbb{Z}}(c_0\, N)\mathsf{M}(N/2) + R^{\star}_{\mathbb{Z}}(N) + O(N),$$

avec $\mathsf{M}_{\mathbb{Z}}(N)$ le nombre d'opérations binaires pour multiplier des entiers à N chiffres.

$$R^{\star}_{\mathbb{Z}}(N) = \mathsf{M}_{\mathbb{Z}}(c_0\, N)\big(\mathsf{MM}(N) + \mathsf{MM}(N/2)\big) + O(N).$$

Par un raisonnement similaire à celui utilisé pour le calcul de la complexité arithmétique, on obtient :

$$R^{\star}_{\mathbb{Z}}(N) = O\big(\mathsf{M}_{\mathbb{Z}}(c_0\, N)\mathsf{M}(N)\big) \quad \text{et} \quad C^{\star}_{\mathbb{Z}}(N) = O\big(\mathsf{M}_{\mathbb{Z}}(c_0\, N)\mathsf{M}(N)\big). \qquad \square$$

Il est possible (cf. [vzGG99]) de multiplier deux entiers de N chiffres binaires avec une complexité $\mathsf{M}_{\mathbb{Z}}(N) = O(N \log N \log\log N)$ et par FFT, le coût du produit de séries tronquées à l'ordre N est $\mathsf{M}(N) = O(N \log N)$; on peut donc estimer la complexité binaire de l'itération de Newton optimisée en fonction de N :

$$C^{\star}_{\mathbb{Z}}(N) = O(N^2 \log^2 N \log\log N).$$

La complexité binaire de l'itération de Newton optimisée est donc quasi-optimale (linéaire, à des facteurs poly-logarithmiques près, en la taille de la sortie) pour calculer les N premiers termes des séries génératrices dont le k-ième coefficient est typiquement de taille $O(k)$.

Chapitre 7

Oracle numérique

Sommaire

Dans notre approche, la dernière étape pour calculer l'oracle de Boltzmann est de transformer les itérations combinatoires en itérations numériques permettant d'évaluer les valeurs des séries génératrices en un point donné. Pour cela, nous donnons le second volet du transfert de convergence, qui nous assure que l'itération numérique converge effectivement vers les valeurs souhaitées, pour les spécifications étiquetées. Le cas des spécifications contenant des ensembles et des cycles non étiquetés reste à traiter.

Comme précédemment, nous illustrons ce résultat sur nos exemples récurrents (les arbres généraux planaires et les graphes série-parallèle) et donnons quelques résultats expérimentaux obtenus avec un prototype de programme qui génère automatiquement un oracle à partir d'une spécification donnée. Bien que nous n'ayons pas démontré la convergence de l'itération numérique dans le cas d'une spécification impliquant des opérateurs de Pólya, nous présentons un exemple (les graphes série-parallèle) nous confortant dans l'idée que l'on peut étendre nos résultats à l'ensemble des classes combinatoires spécifiables. Le principal résultat de ce chapitre est un oracle numérique pour les générateurs de Boltzmann :

Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ une spécification combinatoire remplissant les conditions du théorème des espèces implicites. Si α est une valeur choisie à l'intérieur du disque de convergence du vecteur de séries (ordinaires ou exponentielles) $\boldsymbol{Y}(z)$ de $\boldsymbol{\mathcal{Y}}$, et si la spécification $\boldsymbol{\mathcal{H}}$ ne comporte pas d'opérateur de Pólya, alors l'itération numérique suivante converge vers $\boldsymbol{Y}(\alpha)$:

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \left(\mathbf{Id} - \frac{\boldsymbol{\partial H}}{\boldsymbol{\partial Y}}(\alpha, \boldsymbol{y}^{[n]})\right)^{-1} \cdot (\boldsymbol{H}(\alpha, \boldsymbol{y}^{[n]}) - \boldsymbol{y}^{[n]}), \qquad \boldsymbol{y}^{[0]} = \mathbf{0}.$$

1 Transfert de convergence

1.1 Spécifications analytiques

Pour prouver la validité de notre méthode pour calculer l'oracle numérique, nous devons imposer certaines restrictions sur les spécifications combinatoires qui peuvent être traitées de cette façon. La définition suivante caractérise les systèmes auxquels nous pourrons appliquer le transfert de convergence du lemme 16.

Définition 31. *Une spécification est dite* analytique *si ses séries génératrices* $\boldsymbol{H}(z, \boldsymbol{Y})$ *sont analytiques en* $(z, \boldsymbol{Y})$ *au voisinage de* $(0, \mathbf{0})$*, avec des coefficients positifs.*

Le lemme suivant nous garantit que les spécifications considérées dans ce chapitre sont analytiques au sens de cette définition.

Lemme 15. *Toute spécification combinatoire (étiquetée ou non) utilisant les constructions :*

$$\{\mathcal{E}, \mathcal{Z}, +, \times, \textsc{Seq}, \textsc{Set}, \textsc{Cyc}\}$$

et ne comportant pas d'opérateur de Pólya est analytique.

Démonstration. D'une part, l'addition et la multiplication préservent l'analyticité à l'origine et la positivité. Les unions disjointes et les produits cartésiens de spécifications analytiques sont donc analytiques. D'autre part, d'après [Car95], si $f(z)$ est analytique en 0 et $g(z_0, \ldots, z_m)$ est analytique en $\mathbf{0}$, alors $(f \circ g)(z_0, \ldots, z_m)$ l'est aussi. Les spécifications considérées sont donc analytiques en vertu de l'analyticité à l'origine de

$$\exp(z), \quad 1/(1-z), \quad \log(1/(1-z))$$

et de leurs compositions. La positivité est également préservée. □

En revanche, les spécifications non étiquetées dans lesquelles interviennent des ensembles et des cycles n'appartiennent pas à cette catégorie ; elle ne sont donc pas couvertes par le lemme 16. Pour autant, nous verrons avec l'exemple des graphes série-parallèle, à la section 2.2, que l'on peut espérer étendre la convergence numérique à l'ensemble des spécifications combinatoires.

1.2 Convergence de l'itération numérique

La preuve de l'itération de Newton numérique est une conséquence du lemme de transfert suivant et de la convergence de l'itération de Newton combinatoire (cf. proposition 13, page 98).

Lemme 16 (Transfert, seconde partie)**.** *Soit* $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ *une spécification combinatoire telle que* $\boldsymbol{\mathcal{H}}(0, \mathbf{0}) = \mathbf{0}$ *et* $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(0, \mathbf{0})$ *est nilpotente. Si* $\boldsymbol{\mathcal{F}}$ *est une spécification analytique et si l'itération combinatoire*

$$\boldsymbol{\mathcal{Y}}^{[n+1]} = \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Y}}^{[n]}), \qquad \textit{avec } \boldsymbol{\mathcal{Y}}^{[0]} = \mathbf{0},$$

est croissante et converge vers la solution $\boldsymbol{\mathcal{Y}}$ *de* $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$*, alors les séries*[38] $\boldsymbol{Y}(z)$ *ont un rayon de convergence positif* ρ*, et pour tout* α *tel que* $|\alpha| < \rho$*, l'itération :*

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{F}(\alpha, \boldsymbol{y}^{[n]}), \qquad \textit{avec } \boldsymbol{y}^{[0]} = \mathbf{0},$$

converge vers le vecteur $\boldsymbol{Y}(\alpha)$ *des valeurs des séries génératrices de la famille* $\boldsymbol{\mathcal{Y}}$*, au point* α*.*

[38] Pour alléger l'énoncé, on note indifféremment $\boldsymbol{Y}(z)$ pour des séries ordinaires ou exponentielles.

Démonstration. Les notations utilisées pour cette démonstration sont celles des séries génératrices ordinaires. Les mêmes arguments restent valables pour des séries exponentielles.

Comme $\boldsymbol{\mathcal{F}}$ est analytique et que la matrice $(\boldsymbol{I} - \boldsymbol{\partial F}/\boldsymbol{\partial Y})(0, \mathbf{0})$ est inversible, le théorème des fonctions implicites nous assure que la série $\boldsymbol{Y}(z)$ est analytique en 0 (voir [Car95, Ch. IV]).

L'idée est de montrer que, pour tout α tel que $0 \leq |\alpha| < \rho$, d'une part $\boldsymbol{Y}^{[n]}(\alpha)$ converge vers $\boldsymbol{Y}(\alpha)$ et d'autre part $\boldsymbol{Y}^{[n]}(\alpha) = \boldsymbol{y}^{[n]}$, c'est-à-dire que les valeurs obtenues par évaluation de $\boldsymbol{Y}^{[n]}(z)$ au point $z = \alpha$ sont égales aux valeurs calculées par l'itération numérique.

1. La monotonicité de la suite combinatoire implique que pour tout n et tout k positifs, le coefficient de z^k dans $\boldsymbol{Y}^{[n]}(z)$ est borné par celui de $\boldsymbol{Y}(z)$:

$$[z^k]\boldsymbol{Y}^{[n]}(z) \leq [z^k]\boldsymbol{Y}(z). \tag{1}$$

Pour tout $\varepsilon > 0$, la convergence de $\boldsymbol{Y}(z)$ en α entraîne l'existence de K tel que $|\sum_{k>K} a_k r^k| < \varepsilon$, où les $a_k > 0$ sont les coefficients de la série $\boldsymbol{Y}(z)$. D'après (1), on a également, pour tout n :

$$\left| \sum_{k>K} [z^k]\boldsymbol{Y}(z) r^k - \sum_{k>K} [z^k]\boldsymbol{Y}^{[n]}(z) r^k \right| < \varepsilon.$$

Par ailleurs, d'après le lemme 12, la suite $(\boldsymbol{Y}^{[n]}(z))_{n\in\mathbb{N}}$ converge vers $\boldsymbol{Y}(z)$, *i.e.*, il existe N tel que pour tout $n > N$,

$$[z^k]\boldsymbol{Y}^{[n]}(z) = [z^k]\boldsymbol{Y}(z), \quad \text{pour } k = 0, \ldots, K.$$

Donc pour tout $\varepsilon > 0$, il existe N tel que pour tout $n > N$ et tout z avec $|z| \leq \alpha$, on a :

$$|\boldsymbol{Y}^{[n]}(z) - \boldsymbol{Y}(z)| < \varepsilon.$$

Ainsi $\boldsymbol{Y}^{[n]}(\alpha)$ converge vers $\boldsymbol{Y}(\alpha)$.

2. Soit r tel que $|\alpha| \leq r < \rho$. Si l'on admet que $\boldsymbol{F}(z, \boldsymbol{Y})$ est analytique dans le polydisque $|(z, \boldsymbol{Y})| \leq (r, \boldsymbol{Y}(r))$, où $\leq$ est une inégalité composante par composante, alors le vecteur $\boldsymbol{F}(\alpha, \boldsymbol{Y}^{[n]})$ est bien défini, et donc, par induction, on a :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{F}(\alpha, \boldsymbol{y}^{[n]}) = \boldsymbol{F}(\alpha, \boldsymbol{Y}^{[n]}(\alpha)) = \boldsymbol{Y}^{[n+1]}(\alpha).$$

Il nous reste donc à prouver l'analyticité de $\boldsymbol{F}(z, \boldsymbol{Y})$. Soit $\boldsymbol{F}(z, \boldsymbol{Y}) = \sum \boldsymbol{f}_{i,\boldsymbol{j}} z^i Y_1^{j_1} \cdots Y_m^{j_m}$ et $\boldsymbol{Y}(z) = \sum \boldsymbol{c}_k z^k$. Pour chaque composante Y_h du vecteur $\boldsymbol{Y}$, avec $h \in \{1, \ldots, m\}$, l'extraction du coefficient de z^k, pour $k = 0, \ldots, N$, dans l'identité $\boldsymbol{F}(z, \boldsymbol{Y}(z)) = \boldsymbol{Y}(z)$ nous donne une inégalité de la forme :

$$\sum_{i+j_1\ell_1+\cdots+j_m\ell_m\leq N} f_{h,i,\boldsymbol{j}}\, r^i \Big(\sum_{k_1=0}^{\ell_1} c_{1,k} r^{k_1}\Big)^{j_1} \cdots \Big(\sum_{k_m=0}^{\ell_m} c_{m,k} r^{k_m}\Big)^{j_m} = \sum_{k\leq N} c_{h,k} r^k \leq Y_h(r), \qquad N \in \mathbb{N},$$

où les premiers indices des coefficients représentent les coordonnées des vecteurs. Les coefficients étant positifs, la première somme converge vers $Y_h(r)$ lorsque $N \to \infty$. Cela prouve la convergence de $F_h(z, \boldsymbol{Y})$ pour $|(z, \boldsymbol{Y})| \leq (r, \boldsymbol{Y}(r))$ et par conséquent celle de $\boldsymbol{F}(z, \boldsymbol{Y})$, ce qui conclut cette preuve. □

Théorème 8 (Oracle numérique)**.** *Soit $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}})$ une spécification combinatoire telle que $\boldsymbol{\mathcal{H}}(0, \mathbf{0}) = \mathbf{0}$ et la matrice $\boldsymbol{\partial\mathcal{H}}/\boldsymbol{\partial\mathcal{Y}}(0, \mathbf{0})$ est nilpotente. Soit $\boldsymbol{Y}(z) = \boldsymbol{H}(z, \boldsymbol{Y}(z))$ le système fonctionnel associé sur les séries génératrices (ordinaires ou exponentielles). Si $\boldsymbol{H}(z, \boldsymbol{Y}(z))$ ne comporte pas d'opérateur de Pólya et α est une valeur choisie à l'intérieur du disque de convergence de $\boldsymbol{Y}(z)$, alors l'itération numérique suivante converge vers $\boldsymbol{Y}(\alpha)$:*

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \left(\mathbf{Id} - \frac{\boldsymbol{\partial H}}{\boldsymbol{\partial Y}}(\alpha, \boldsymbol{y}^{[n]})\right)^{-1} \cdot (\boldsymbol{H}(\alpha, \boldsymbol{y}^{[n]}) - \boldsymbol{y}^{[n]}), \qquad \boldsymbol{y}^{[0]} = \mathbf{0}.$$

Démonstration. Ce résultat est une conséquence de la convergence de l'itération de Newton combinatoire et du lemme de transfert 16. □

L'itération de Newton optimisée est également un oracle numérique, a priori légèrement plus efficace (au vu des résultats de complexité obtenus pour le calcul des développements des séries). Néanmoins, nos implantations n'ont pas permis d'observer ce gain jusqu'à maintenant. Ceci est, semble-t-il, dû au fait que, contrairement à nos attentes, en Maple, la multiplication de matrices de flottants[39] est aussi coûteuse que l'inversion. Ces pourquoi nous présentons nos résultats uniquement sur l'itération de Newton classique.

2 Implantation et applications

Dans cette section, nous commençons par reprendre, une dernière fois, les exemples des arbres planaires et des graphes série-parallèle, afin d'observer la convergence numérique que nous venons de démontrer, que ce soit par itération simple ou par itération de Newton. Nous donnons ensuite quelques résultats expérimentaux obtenus avec un prototype d'oracle générique en Maple, pour les spécifications étiquetées.

2.1 Exemple des arbres généraux planaires

Rappelons que les arbres planaires sont définis par :

$$\mathcal{Y} = \mathcal{Z} \times L(\mathcal{Y}) \quad \Rightarrow \quad Y(z) = \frac{z}{1 - Y(z)}.$$

L'itération combinatoire simple se traduit en itération numérique :

$$y^{[n+1]} = \alpha/(1 - y^{[n]}), \qquad \text{avec } \; y^{[0]} = 0.$$

Le rayon de convergence de la série $Y(z)$ est $1/4$; une évaluation au point $\alpha = 0.1$ est donc possible. Cette itération numérique donne alors les valeurs suivantes des $y^{[i]}$, qui correspondent aux valeurs des séries convergentes $Y^{[i]}(z)$ obtenues au chapitre précédent (cf. page 116), évaluées en $z = \alpha$, et qui ont pour limite $Y(\alpha) \sim 0.11270166537925831148207346002176$. Comme aupa-

[39] Ce phénomène n'est observé que pour les flottants Maple ; dans le cas des flottants machine, comme on s'y attend, l'inversion est plus coûteuse d'un facteur constant.

ravant, nous avons indiqué en gras les décimales correctes à chaque itération.

$$
\begin{aligned}
y^{[0]} &= \mathbf{0}\\
y^{[1]} &= \mathbf{0.1}\\
y^{[2]} &= \mathbf{0.11}1111111111111111111111111111111\ldots\\
y^{[3]} &= \mathbf{0.112}5000000000000000000000000000\ldots\\
y^{[4]} &= \mathbf{0.112}6760563380281690140845070422 5\ldots\\
y^{[5]} &= \mathbf{0.112}6984126984126984126984126984 1\ldots\\
y^{[6]} &= \mathbf{0.11270125}22361359570661896243291 6\ldots
\end{aligned}
$$

L'itération de Newton combinatoire, quant à elle, devient l'itération numérique :

$$y^{[n+1]} = y^{[n]} + \frac{\alpha/(1-y^{[n]}) - y^{[n]}}{1-\alpha/(1-y^{[n]})^2}, \qquad y^{[0]} = 0. \tag{2}$$

Pour $\alpha = 0.1$, les valeurs obtenues :

$$
\begin{aligned}
y^{[0]} &= \mathbf{0}\\
y^{[1]} &= \mathbf{0.11}1111111111111111111111111111111\ldots\\
y^{[2]} &= \mathbf{0.11270125}22361359570661896243291 6\ldots\\
y^{[3]} &= \mathbf{0.11270166537923}0322595 4763928873 92\ldots\\
y^{[4]} &= \mathbf{0.11270166537925831148207345}989331\ldots
\end{aligned}
$$

permettent d'observer une convergence quadratique de cette itération, au sens où le nombre de décimales correctes est doublé à chaque étape. Remarquons que l'équation $y = 0.1/(1-y)$ a une autre solution réelle positive, $y \sim 0.887298334620741688 52$, mais qu'en commençant avec la valeur $y^{[0]} = 0$, l'itération de Newton converge effectivement vers la solution numérique correspondant à la valeur de la série génératrice des arbres évaluée en 0.1.

Dans cet exemple, tous les calculs ont été réalisés avec 32 décimales de précision, mais il est possible (et préférable) d'améliorer l'efficacité de l'itération en n'utilisant que la précision nécessaire au calcul et en la doublant à chaque étape lorsque l'itération devient quadratique. Avec une itération de la forme

$$y^{[n+1]} = y^{[n]} + F(\alpha, y^{[n]}),$$

la valeur que l'on cherche est corrigée, à l'étape $n+1$, par le terme $F(\alpha, y^{[n]})$; si la valeur de celui-ci est égale à $c.10^{-p}$ avec $1 \leq c < 10$, cela signifie que les p premières décimales de $y^{[n]}$ étaient correctes. La valeur de $y^{[n+1]}$ est donc obtenue avec une précision de $2p+1$ décimales. Si l'itération a une convergence quadratique, à l'étape suivante, la valeur de $y^{[n+2]}$ devra être calculée avec une précision $2(2p+1)$. En résumé, à chaque étape, on peut ajuster la précision d du calcul de la façon suivante :

$$d \leftarrow 2\,\min(d, 2p+1), \qquad \text{avec } \; p = y^{[n+1]} - y^{[n]}.$$

2.2 Exemple des graphes série-parallèle

Graphes étiquetés On rappelle que les graphes série-parallèle sont définis par :

$$\begin{cases} \mathcal{Y}_2 &= L_{\geq 2}(\mathcal{Z} + \mathcal{Y}_3) \\ \mathcal{Y}_3 &= E_{\geq 2}(\mathcal{Z} + \mathcal{Y}_2) \end{cases},$$

Les séries génératrices exponentielles sont données par le système :

$$\begin{cases} \widehat{Y}_2(z) = (z + \widehat{Y}_3(z))^2/(1 - z - \widehat{Y}_3(z)) \\ \widehat{Y}_3(z) = \exp(z + \widehat{Y}_2(z)) - 1 - z - \widehat{Y}_2(z). \end{cases}$$

La singularité dominante de $\widehat{\boldsymbol{Y}}(z)$ est $\rho = 2 - \sqrt{5} + \ln((1 + \sqrt{5})/2) \approx 0.245$. Pour calculer les valeurs $\widehat{\boldsymbol{Y}}(z)$, on utilise l'itération de Newton numérique :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \boldsymbol{U}(\alpha, \boldsymbol{y}^{[n]})\big(\boldsymbol{H}(\alpha, \boldsymbol{y}^{[n]}) - \boldsymbol{y}^{[n]}\big),$$

avec

$$\boldsymbol{U}(\alpha, \boldsymbol{y}) = \begin{pmatrix} 1 & 1 - \frac{1}{(1-\alpha-y_3)^2} \\ 1 - e^{\alpha+y_2} & 1 \end{pmatrix}^{-1} \quad \text{et} \quad \boldsymbol{H}(\alpha, \boldsymbol{y}) = \begin{pmatrix} \frac{(\alpha+y_3)^2}{1-\alpha-y_3} \\ e^{\alpha+y_2} - 1 - \alpha - y_2 \end{pmatrix}.$$

Avec $\alpha = 0.24 < \rho$, les valeurs obtenues pour les premières itérations sont :

$$\begin{aligned}
\boldsymbol{y}^{[1]} &= (\mathbf{0.1}230510663209943063722\ldots, && \mathbf{0.0}646266475071172143953 5\ldots) \\
\boldsymbol{y}^{[2]} &= (\mathbf{0.1}627000389319615796926\ldots, && \mathbf{0.09}201293266034877734970\ldots) \\
\boldsymbol{y}^{[3]} &= (\mathbf{0.17}243333070032457106 86\ldots, && \mathbf{0.09}798441803578338336038\ldots) \\
\boldsymbol{y}^{[4]} &= (\mathbf{0.1730}460965507535353574\ldots, && \mathbf{0.09836}831514307466499845\ldots) \\
\boldsymbol{y}^{[5]} &= (\mathbf{0.173048639}2973095133433\ldots, && \mathbf{0.0983698991}7963665326450\ldots) \\
\boldsymbol{y}^{[6]} &= (\mathbf{0.1730486393408452105}149\ldots, && \mathbf{0.0983698992067876912 6}015\ldots)
\end{aligned}$$

La figure 1 donne les temps de calculs obtenus avec une implantation directe de l'itération de Newton numérique en Maple[40] sur notre exemple des graphes série-parallèle. Les quatre courbes correspondent à différentes valeurs de α, choisies de telle façon qu'un générateur de Boltzmann utilisant ces valeurs engendrera des graphes dont la taille moyenne est 10 (resp. 100, 10^3 et 10^4) pour la courbe rouge (resp. verte, bleue et noire). Pour chaque courbe, les différents points correspondent aux temps de calcul de l'oracle avec la précision donnée en abscisse. Les "sauts" que l'on peut observer sur chaque courbe correspondent aux moments où le nombre d'itérations pour atteindre la précision souhaitée est augmenté de un.

Graphes non étiquetés Bien que nous n'ayons pas démontré la validité de notre approche dans le cas d'une spécification comportant des opérateurs de Pólya, nous avons testé le comportement de l'itération numérique pour les graphes série-parallèle non étiquetés.

[40] Ce programme est écrit en Maple 11 et a été lancé sur un processeur Intel à 3.2 GHz avec 2 Go de mémoire.

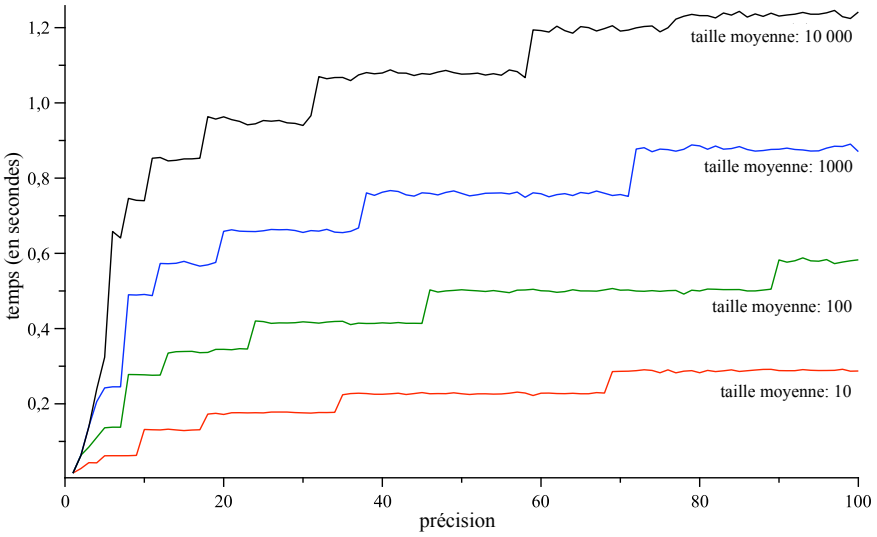


FIG. 1 – Évaluation de l'oracle numérique (par itération de Newton) pour les graphes série-parallèle (temps par rapport à la précision). Les différentes valeurs de α sont 0.2290606680, 0.2450366213, 0.2451428138 et 0.2451438373).

Les séries génératrices ordinaires des graphes série-parallèle sont données par le même système que pour les graphes étiquetés, en remplaçant la série exponentielle $\exp\big(\widehat{F}(z)\big)$ des ensembles de $\mathcal{F}$-structures par son analogue ordinaire :

$$\overline{\mathrm{Exp}}\,(F(z)) = \exp\left(\sum_{k\geq 1}\frac{F(z^k)}{k}\right)$$

L'itération de Newton est alors la même que la précédente. Néanmoins, elle présente une difficulté supplémentaire qui est l'évaluation du terme $\overline{\mathrm{Exp}}(\alpha + y_2^{[n]})$, étant donné que celui-ci dépend des valeurs de $\boldsymbol{Y}(\alpha^i)$, pour $i \geq 2$. L'idée est de commencer par calculer récursivement ces valeurs, pour pouvoir calculer l'itération pour les valeurs de $\boldsymbol{Y}(\alpha)$. On réécrit la série des ensembles sous la forme :

$$\overline{\mathrm{Exp}}(z + Y_2(z)) = R(z)\cdot e^{z+Y_2(z)} \quad \text{avec} \quad R(z) = \exp\left(\sum_{k\geq 2}\frac{z^k + Y_2(z^k)}{k}\right)$$

et l'itération de Newton numérique devient :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \boldsymbol{U}(\alpha, r, \boldsymbol{y}^{[n]})\big(\boldsymbol{H}(\alpha, r, \boldsymbol{y}^{[n]}) - \boldsymbol{y}^{[n]}\big),$$

avec $r = R(\alpha)$

$$\boldsymbol{U}(\alpha, r, \boldsymbol{y}) = \begin{pmatrix} 1 & 1 - \frac{1}{(1-\alpha-y_3)^2} \\ 1 - r\cdot e^{\alpha+y_2} & 1 \end{pmatrix}^{-1}$$

et

$$\boldsymbol{H}(\alpha, r, \boldsymbol{y}) = \begin{pmatrix} (\alpha+y_3)^2/(1-\alpha-y_3) \\ r\cdot e^{\alpha+y_2} - 1 - \alpha - y_2 \end{pmatrix}.$$

Pour que cette itération devienne un algorithme effectif, il nous reste à déterminer combien de termes de $R(\alpha)$ doivent être calculés en fonction de la valeur de α et de la précision à laquelle on souhaite connaître les valeurs $\boldsymbol{y}$. On ne détermine pas ce nombre de termes a priori, mais on calcule $R(\alpha)$ de façon incrémentale jusqu'à ce que le terme correcteur ajouté à l'étape courante ait une valeur inférieure à la précision souhaitée. De même, pour stopper le calcul récursif, lorsque α_k devient suffisamment petit par rapport à α, on peut évaluer directement les valeurs de $\boldsymbol{Y}(\alpha^k)$ à partir des premiers termes du développement de

$$\boldsymbol{Y}(z) = \sum_{k\geq 0} \boldsymbol{Y}_k z^k,$$

qui peuvent être obtenus par itération de Newton sur les séries (voir chapitre précédent). Pour calculer $\boldsymbol{y}$ au point α_0, avec une précision p, on utilisera alors l'algorithme suivant, avec $\alpha = \alpha_0$.

```
OracleSP(α0, α, p)
  si α < α0/1000 alors
    S ← 0;
    k ← 0
    faire
        S ← S + Y_k α^k
        k ← k + 1
    tant que Y_k α^k > 10^-p
    renvoyer S
  sinon
    ### calcul récursif de R ###
    R^[1] ← 1;
    r ← 2
    faire
      R^[r+1] ← R^[r] · exp((α^r + OracleSP(α0, α^r, p))/r)
      erreur ← R^[r+1] − R^[r]
      r ← r + 1
    tant que erreur > 10^-p
    R ← R^[r+1]

    ### calcul de y par itération de Newton ###
    y^[0] ← 0;
    n ← 0
    faire
      y^[n+1] = y^[n] + U(α, R, y^[n])(H(α, R, y^[n]) − y^[n])
      erreur ← y^[n+1] − y^[n]
      n ← n + 1
    tant que erreur > 10^-p
    renvoyer y^[n+1]
```

La singularité dominante[41] est $\rho \approx 0.21638$; on peut donc calculer $\boldsymbol{y}$ au point $\alpha = 0.21$. Les

[41] La valeur de la singularité est déterminée de façon heuristique.

premières itérations sont les suivantes, en partant de $\boldsymbol{y}^{[0]} = \mathbf{0}$.

$$\begin{aligned}
\boldsymbol{y}^{[1]} &= (\mathbf{0.1}076544062279541610248230\ldots, && \mathbf{0}.086055107546864038030306 01\ldots\ ) \\
\boldsymbol{y}^{[2]} &= (\mathbf{0.1}499887685016433643039636\ldots, && \mathbf{0.11}10825690902974438243804\ldots\ ) \\
\boldsymbol{y}^{[3]} &= (\mathbf{0.157}500525711405309719385 7\ldots, && \mathbf{0.11}59135012074061657990486\ldots\ ) \\
\boldsymbol{y}^{[4]} &= (\mathbf{0.15779}877986896977220370 69\ldots, && \mathbf{0.11609}337969969893351876 86\ldots\ ) \\
\boldsymbol{y}^{[5]} &= (\mathbf{0.1577992390}122141022215320\ldots, && \mathbf{0.1160996261}359131512704389\ldots\ ) \\
\boldsymbol{y}^{[6]} &= (\mathbf{0.157799239013311994370}4663\ldots, && \mathbf{0.1160996261366008322216}1995\ldots)
\end{aligned}$$

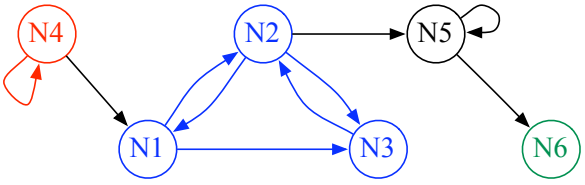
Ces valeurs correspondent aux valeurs que l'on obtient en calculant les premiers termes du développement du vecteur de séries $\boldsymbol{Y}(z)$ et en évaluant la somme $\sum_{k\geq 0} \boldsymbol{Y}_k \alpha^k$ jusqu'au premier indice k pour lequel $\boldsymbol{Y}_k \alpha^k$ est inférieur à une précision fixée (10^{-20}, par exemple).

2.3 Prototype et tests sur des spécifications aléatoires

Calculer les valeurs des séries par itération n'a d'intérêt que si la spécification considérée est récursive. On peut utiliser cette observation afin d'améliorer encore un peu notre oracle ; considérons, par exemple, la spécification suivante :

$$\begin{aligned}
G := \{&N_1 = Union(Z, Prod(Z, Prod(Set(N_3), Union(Set(Prod(N_2, Z)), N_2)))),\\
&N_2 = Union(Z, Prod(Z, Prod(Union(N_1, Cycle(N_3)), Sequence(Prod(N_3, N_5))))),\\
&N_3 = Union(Z, Prod(Z, Prod(Cycle(Z), Prod(N_2, Prod(Z, N_3))))),\\
&N_4 = Union(Z, Prod(Z, Prod(N_1, Union(N_4, N_4)))),\\
&N_5 = Union(Z, Prod(Z, Prod(Union(Prod(N_5, Cycle(Z)), N_5), N_6))),\\
&N_6 = Union(Z, Prod(Z, Prod(Cycle(Prod(Z, Z)), Cycle(Z))))\}.
\end{aligned}$$

Remarquons que l'équation qui définit N_6 ne dépend d'aucune autre. On peut donc calculer la valeur de $N_6(\alpha)$ indépendamment du reste du système. Plus globalement, on peut tracer le graphe de dépendances des N_i et décomposer G en quatre composantes fortement connexes, $\{N_1, N_2, N_3\}$, $\{N_4\}$, $\{N_5\}$ et $\{N_6\}$:



On peut calculer séparément les valeurs de leurs séries associées, dans l'ordre du graphe acyclique dirigé induit par cette décomposition. Une itération de Newton différente est utilisée pour chacune des composantes présentant au moins un cycle (dans notre exemple : $\{N_1, N_2, N_3\}$, $\{N_4\}$ et $\{N_5\}$). Ainsi, en décomposant les spécifications en composantes fortement connexes, on peut diminuer le nombre d'équations des systèmes sur lesquels on opère et donc travailler sur des matrices de plus petite dimension.

Nous avons implanté un prototype d'oracle de Boltzmann pouvant traiter les spécifications étiquetées (et non étiquetées, restreintes aux constructions d'union, de produit et de séquence) et utilisant l'itération de Newton. À partir d'une spécification donnée sous la forme d'une grammaire Combstruct, ce programme engendre automatiquement une fonction Maple qui prend comme paramètre la valeur du point auquel on souhaite évaluer les séries génératrices associées

à la grammaire et la précision en nombre de décimales. La table 1 nous donne des indications concernant les performances[40] de ce prototype. Pour ces tests, nous avons engendré des grammaires aléatoires en faisant varier le nombre d'équations et le nombre de constructions imbriquées par équation. Ces paramètres sont donnés par les lignes 1 et 2 de la table 1. Les données présentées dans chaque colonne sont alors des moyennes effectuées sur une centaine de grammaires. Étant donné que l'on ne lance l'itération de Newton que sur des composantes fortement connexes isolées, pour chaque grammaire, la taille de la plus grande composante a un impact significatif sur la performance de l'oracle, c'est pourquoi nous avons indiqué la moyenne de la taille de cette composante pour les spécifications considérées.

Les temps de calcul sont donnés en secondes. Pour chaque spécification, l'oracle est appelé avec un paramètre proche de la singularité du système[41]. La dernière ligne du tableau 1 donne une moyenne, sur les grammaires testées, de l'espérance de la taille des structures[42] qu'engendrerait un générateur de Boltzmann utilisant comme paramètre 0.999999ρ.

# équations	4	10	50		100		500
# constructions/eqn	10	10	10	50	10	50	50
# plus grande c.f.c.	2.47	3.42	7.95	18.62	10.93	67.18	339.1
temps (0.99ρ)	0.05	0.11	0.17	0.47	0.23	7.29	61.73
temps (0.999999ρ)	0.08	0.16	0.19	0.56	0.25	8.11	61.86
taille moyenne engendrée	$4.1\,10^{14}$	$1.4\,10^{7}$	$2.2\,10^{5}$	$1.0\,10^{5}$	$1.2\,10^{6}$	$5.0\,10^{4}$	$3.3\,10^{4}$

Tab. 1 – Résultats expérimentaux (temps en secondes).

2.4 Applications

Parallèlement à ces exemples "artificiels", notre prototype a aussi été utilisé pour calculer l'oracle afin d'engendrer des structures aléatoires pour des systèmes apparaissant dans des applications concrètes. Par exemple, A. Darrasse utilise la méthode de Boltzmann pour engendrer des documents XML [Dar08] : il a développé un programme qui, étant donné une grammaire RELAX NG quelconque, produit des arbres aléatoires uniformes de très grande taille et qui sont des documents XML valides. Ces documents aléatoires peuvent, par exemple, être utilisés pour tester la vulnérabilité des services web. Avec des grammaires context-free telles que MathML et DocBook, qui correspondent à des systèmes combinatoires de plusieurs centaines d'équations, notre oracle permet, en quelques secondes, de calculer les valeurs des séries qui permettent d'engendrer des documents aléatoires de plus de 10 000 nœuds.

Dans [Oud07], J. Oudinet s'intéresse à la génération de chemins dans de grands graphes modélisant des systèmes concurrents. Il implante la méthode présentée dans [DGG$^+$06], basée sur une décomposition des graphes en modules indépendants et l'utilisation de la méthode récursive que nous avons présentée au chapitre 1, pour engendrer des mots uniformes dans des langages réguliers. En utilisant notre oracle, pour une grammaire de 1 183 équations, il a pu, par exemple, engendrer en moins de 2 minutes, environ 4 000 chemins de plus 2.10^5 nœuds.

[42] L'espérance varie énormément d'un système à l'autre, ce chiffre est simplement donné pour souligner le fait qu'avec cette précision, il est possible d'engendrer des objets de très grande taille.

Conclusion et perspectives

Conclusion et perspectives

Les résultats présentés dans cette thèse font de la méthode de Boltzmann une méthode effective et efficace pour engendrer des structures combinatoires aléatoires.

C'est une méthode *effective* au sens où nous disposons désormais de tous les éléments permettant d'implanter des générateurs automatiques à partir des spécifications combinatoires. Les algorithmes génériques développés dans la première partie de ce mémoire, associés à ceux de l'article initial de Duchon *et al.* [DFLS04], forment un ensemble cohérent permettant d'assembler des générateurs de Boltzmann pour un grand nombre de classes combinatoires.

La mise en œuvre systématique de la méthode de Boltzmann passe par l'automatisation du calcul de l'oracle. La seconde partie de ce mémoire présente une chaîne de traitement complète produisant, à partir d'une spécification combinatoire, un oracle de Boltzmann sous la forme d'une itération de Newton numérique. Un transfert de convergence est opéré des structures combinatoires vers les valeurs numériques en passant par une itération de Newton sur les séries génératrices. Cette itération sur les séries est par ailleurs un algorithme particulièrement efficace pour calculer les suites de dénombrement des structures combinatoires.

Les nombreux exemples présentés dans ce mémoire visent à illustrer l'*efficacité* de la méthode de Boltzmann. Le principe sur lequel elle repose est vecteur d'efficacité : relâcher la contrainte sur la taille des structures à engendrer entraîne une certaine souplesse au niveau des algorithmes qui se traduit par un gain de complexité par rapport à des méthodes génériques à taille fixée. Mais l'efficacité de la méthode de Boltzmann réside également dans les algorithmes qui l'implantent. En suivant la décomposition récursive des structures, les algorithmes de génération ont une complexité linéaire qui permet d'atteindre sans difficulté des tailles jusqu'alors inaccessibles. L'itération de Newton numérique conduit également à un oracle rapide pour calculer les valeurs numériques qui dictent les choix probabilistes faits par ces algorithmes. Notre prototype d'oracle a déjà permis, pour de gros systèmes (de l'ordre de 10^3 équations), d'évaluer les séries génératrices pour des valeurs du paramètre qui permettent d'engendrer des structures de très grande taille (10^6 et plus). Il devient, dès lors, tout à fait envisageable d'utiliser la méthode de Boltzmann pour des applications à grande échelle.

Concernant le calcul de l'oracle, quelques points restent en suspens pour pouvoir traiter l'ensemble des spécifications combinatoires couvertes par la méthode de Boltzmann ; ces questions font l'objet d'un travail en cours [PSS08] :

- Nous prouvons que notre approche reste valide à l'étape numérique, pour les opérateurs de Pólya. Pour cela, nous étendons la notion de spécification analytique de façon à prendre en compte ces opérateurs.
  L'exemple des graphes série-parallèle (voir page 134) montre que l'algorithme permettant d'obtenir l'oracle conduit alors à des approximations sur les valeurs numériques à différents

niveaux (la troncature des sommes infinies et l'arrêt de la récursion sur les puissances du paramètre de Boltzmann). Nous devons donc également vérifier que ces approximations n'entraînent pas de perte de précision sur les valeurs renvoyées.

- Les hypothèses du théorème des espèces implicites, et plus précisément la condition $\boldsymbol{\mathcal{H}}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$, n'autorisent pas l'utilisation des atomes de taille 0 (que l'on note $\mathcal{E}$) dans les spécifications combinatoires. Or, ces atomes permettent de définir un certain nombre de classes (comme les arbres binaires énumérés par leurs noeud internes : $\mathcal{B} = \mathcal{E} + \mathcal{Z} \times \mathcal{B}^2$) pour lesquelles on veut pouvoir fournir un oracle. À cette fin, nous donnons les conditions pour qu'une spécification $\boldsymbol{\mathcal{H}}$ telle que $\boldsymbol{\mathcal{H}}(\mathbf{0}, \mathbf{0}) \neq \mathbf{0}$, définisse effectivement une unique espèce de structures.
- Enfin, à un autre niveau, nous achevons l'étude de la complexité binaire de l'itération de Newton sur les séries génératrices, dans le cas des structures étiquetées (pour lesquelles les coefficients ne sont plus des entiers, mais des fractions rationnelles).

Nous avons vu, dans le premier chapitre de ce mémoire (page 39), que pour engendrer de grandes structures, on peut avoir recours à des générateurs singuliers. Pour ces générateurs, le paramètre de Boltzmann est égal à la singularité dominante des séries génératrices associées à la classe combinatoire considérée. Or, il apparaît de façon expérimentale, qu'en se rapprochant de la singularité, l'itération de Newton nécessite de plus en plus d'étapes avant d'atteindre le régime quadratique. Il serait donc intéressant de savoir dans quelle mesure on peut utiliser des méthodes classiques d'accélération de convergence (Schröder) pour améliorer les performances de l'oracle à proximité de la singularité.

Une autre question se pose alors : comment estimer la valeur numérique de cette singularité ? Jusqu'à maintenant, on a toujours supposé que l'on savait régler la valeur du paramètre de Boltzmann correctement, c'est-à-dire déterminer l'intervalle dans lequel il peut être choisi. Or c'est un problème que l'on ne sait pas résoudre dans le cas général. On peut envisager deux façons différentes d'utiliser les résultats de cette thèse pour répondre à cette question. On peut essayer de calculer cette valeur par itération de Newton (ce qui revient à chercher une valeur qui annule le déterminant de la matrice jacobienne du système) ; mais on sort alors du cadre purement combinatoire, en perdant le bénéfice des propriétés liées aux structures (telles que la croissance de l'itération). Une autre possibilité est d'utiliser l'oracle de Boltzmann comme une boîte noire pour calculer les singularités par dichotomie, à condition de savoir déterminer, à partir du comportement de l'oracle, si la valeur de la singularité est dépassée.

Enfin, la calcul des singularités est relié au problème du choix du paramètre de Boltzmann : pour automatiser le réglage des générateurs, il faudrait fournir une méthode systématique qui calcule la valeur du paramètre en fonction de l'espérance de la taille visée pour les structures à engendrer.

Table des figures

Liste des tableaux

Bibliographie

[BCG+07] A. Bostan, F. Chyzak, M. Guisti, B. Salvy, and É. Schost. Algorithmes en calcul formel et en automatique. Polycopié du cours de MPRI : Algorithmes efficaces en calcul formel, décembre 2007.

[BCS97] P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig.

[BDLP99] E. Barcucci, A. Del Lungo, and E. Pergola. Random generation of trees and other combinatorial objects. *Theoretical Computer Science*, 218 :219–232, 1999.

[BDLPP99] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. Eco : a methodology for the enumeration of combinatorial objects. *Journal of Difference Equations and Applications*, pages 1–56, 1999.

[BDN07] F. Bassino, J. David, and C. Nicaud. REGAL : a library to randomly and exhaustively generate automata. In Jan Holub and Jan Žďárek, editors, *12th (CIAA'07)*, volume 4783, pages 303–305, Prague, Czech Republic, July 2007.

[BDN08] F. Bassino, J. David, and C. Nicaud. Random generation of possibly incomplete deterministic automata. In *Génération Aléatoire de Structures COMbinatoires (Gascom'08)*, pages 31–40, June 2008.

[BF08] N. Broutin and P. Flajolet. The height of random binary unlabelled trees. In *Proceedings of the Fifth Colloquium on Mathematics and Computer Science : Algorithms, Trees, Combinatorics and Probabilities*, Blaubeuren, Germany, September 2008.

[BFKV07] M. Bodirsky, É. 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.

[BFP06] O. Bodini, E. Fusy, and C. Pivoteau. Random sampling of plane partitions. In Renzo Pinzani and Vincent Vajnovszki, editors, *Gascom 2006*, pages 124–135, Dijon, France, 2006. LE2I.

[BFP07] O. Bodini, E. Fusy, and C. Pivoteau. Random sampling of plane partitions. submitted, 2007.

[BJ08] O. Bodini and A. Jacquot. Boltzmann samplers for colored combinatorial objects. In *Génération Aléatoire de Structures COMbinatoires (Gascom'08)*, June 2008.

[BK72] E. A. Bender and D. E. Knuth. Enumeration of plane partitions. *Journal of Combinatorial Theory Series A*, 13(1) :40–54, 1972.

[BK78] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4) :581–595, 1978.

[BLL98] F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial species and tree-like structures*, volume 67 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1998. Translated from the 1994 French original by Margaret Readdy, With a foreword by Gian-Carlo Rota.

[BN07] F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Theoretical Computer Science*, 381(1-3) :86–104, 2007.

[BPS94] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of directed animals. *Theoretical Computer Science*, 127(2) :333–350, 1994.

[BPS08a] N. Bernasconi, K. Panagiotou, and A. Steger. On properties of random dissections and triangulations. In *SODA '08 : Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–141, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[BPS08b] N. Bernasconi, K. Panagiotou, and A. Steger. On the degree sequences of random outerplanar and series-parallel graphs. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *APPROX-RANDOM*, volume 5171 of *Lecture Notes in Computer Science*, pages 303–316. Springer, 2008.

[Bre99] D. M. Bressoud. *Proofs and confirmations : the story of the alternating sign matrix conjecture*. Cambridge University Press, New York, NY, USA, 1999.

[BS05] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4) :420–446, 2005.

[Car95] H. Cartan. *Elementary theory of analytic functions of one or several complex variables*. Dover Publications, 1995. Reprint of the 1973 ed, translated from the 1961 French original.

[CK01] R. Cerf and R. Kenyon. The low-temperature expansion of the Wulff crystal in the 3D Ising model. *Communications in Mathematical Physics*, 222(1) :147–179, 2001.

[Dar08] A. Darrasse. Random XML sampling the Boltzmann way. Technical Report 0807.0992, arXiv, 2008.

[DDZ98] A. Denise, I. Dutour, and P. Zimmermann. CS : a MuPAD package for counting and randomly generating combinatorial structures. In *Proceedings of 10th Conference on Formal Power Series and Algebraic Combinatorics (FPSAC'98)*, pages 195–204, 1998. Also published in MathPAD 8 (1) 1998.

[Den01] A. Denise. Structures aléatoires, modèles et analyse des génomes. Habilitation à Diriger des Recherches, décembre 2001.

[Dev86] L. Devroye. *Non-Uniform Random Variate Generation*. Springer Verlag, 1986.

[DF98] I. Dutour and J. M. Fédou. Object grammars and random generation. *Discrete Mathematics and Theoretical Computer Science*, 2 :47–61, 1998.

[DFLS02] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Random sampling from Boltzmann principles. In *Automata, languages and programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 501–513. Springer, Berlin, 2002.

[DFLS04] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13(4–5) :577–625, 2004. Special issue on Analysis of Algorithms.

[DGG+06] A. Denise, M.-C. Gaudel, S.-D. Gouraud, R. Lassaigne, and S. Peyronnet. Uniform random sampling of traces in very large models. In *1st International ACM Workshop on Random Testing*, pages 10 –19, July 2006.

[DLL82] H. Décoste, G. Labelle, and P. Leroux. Une approche combinatoire pour l'itération de Newton-Raphson. *Advances in Applied Mathematics*, 3 :407–416, 1982.

[DRT00] A. Denise, Olivier R., and M. Termier. Random generation of words of context-free languages according to the frequencies of letters. In *Mathematics and computer science (Versailles, 2000)*, Trends in Mathematics, pages 113–125. Birkhäuser, Basel, 2000.

[DS07] A. Darrasse and M. Soria. Degree distribution of random apollonian network structures and boltzmann sampling. In Philippe Jacquet, editor, *Analysis of Algorithms 2007 (AofA07)*, Discrete Mathematics and Theoretical Computer Science Proceedings, 2007. In press.

[DZ99] A. Denise and P. Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science*, 218 :233–248, 1999. Preliminary version available in INRIA Research Report RR-3242, September 1997.

[FFP07] P. Flajolet, É. Fusy, and C. Pivoteau. Boltzmann sampling of unlabelled structures. In David Applegate, Gerth Stølting Brodal, Daniel Panario, and Robert Sedgewick, editors, *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithmics and Combinatorics*, volume 126 of *SIAM Proceedings in Applied Mathematics*, pages 201–211. SIAM, 2007. Workshops held in New Orleans, LA, January 6, 2007.

[FGD95] P. Flajolet, X. Gourdon, and P. Dumas. Mellin transforms and asymptotics : Harmonic sums. *Theoretical Computer Science*, 144(1–2) :3–58, June 1995.

[FS08] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008. Free electronic version of a book to be published in 2008 ; 810p.+xii, available electronically from the authors' home pages.

[FSZ91] P. Flajolet, B. Salvy, and P. Zimmermann. Automatic average-case analysis of algorithms. *Theoretical Computer Science*, 79(1, (Part A)) :37–109, 1991. Algebraic and computing treatment of noncommutative power series (Lille, 1988).

[Fus05] É. Fusy. Quadratic exact-size and linear approximate-size random generation of planar graphs. In *2005 International Conference on Analysis of Algorithms*, Discrete Mathematics Theoretical Computer Science Proceedings, AD., pages 125–138 (electronic). 2005.

[Fus07] É. Fusy. *Combinatoire des cartes planaires et applications algorithmiques*. PhD thesis, École Polytechnique, June 2007.

[FZVC94] P. Flajolet, P. Zimmermann, P., and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1-2) :1–35, 1994.

[Gan81] E. R. Gansner. Matrix correspondences of plane partitions. *Pacific Journal of Mathematics*, 92(2) :295–315, 1981.

[GB93] D. Gouyou-Beauchamps. Quelques exemples d'algorithmes de génération aléatoire (summary by M. Soria). In Bruno Salvy, editor, *Algorithms Seminar, 1992–1993*, number 2130, December 1993.

[GB03] D. Gouyou-Beauchamps. Combinatorics and random generation (summary by Nicolas Bonichon). In Frédéric Chyzak, editor, *Algorithms Seminar, 2001–2002*, volume 5003 of *Research Report*. Institut National de Recherche en Informatique et en Automatique, November 2003.

[Gre83] D. H. Greene. *Labelled formal languages and their uses*. PhD thesis, Stanford, CA, USA, 1983.

[HC83] T. Hickey and J. Cohen. Uniform random generation of strings in a context-free language. *SIAM Journal on Computing*, 12(4) :645–655, 1983.

[HT03] F. Hivert and N. Thiéry. Mupad-combinat, an open-source package for research in algebraic combinatories. *Séminaire Lotharingien de Combinatoire*, 51 :70, 2003.

[Jer94] M. Jerrum. Uniform sampling modulo a group of symmetries using markov chain simulation. Technical Report ECS-LFCS-94-288, University of Edinburgh, 1994.

[Joy81] A. Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42(1) :1–82, 1981.

[Knu70] D. E. Knuth. Permutations, matrices, and generalized Young tableaux. *Pacific Journal of Mathematics*, 34 :709–727, 1970.

[Knu06] D. E. Knuth. *The Art of Computer Programming : Volume 4, Combinatorial Algorithms*. Addison Wesley, 2005–2006. Fascicles 2-4.

[Kra99] C. Krattenthaler. Another involution principle-free bijective proof of Stanley's hook-content formula. *Journal of Combinatorial Theory Series A*, 88(1) :66–92, 1999.

[Lab85] G. Labelle. Éclosions combinatoires appliquées à l'inversion multidimensionnelle des séries formelles. *Journal of Combinatorial Theory Series A*, 39(1) :52–82, 1985.

[Lab90] G. Labelle. Dérivées directionnelles et développements de Taylor combinatoires. *Discrete Mathematics*, 79(3) :279–297, 1990.

[Lot83] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, 1983.

[Mac96] P. A. MacMahon. Memoir on the theory of the partition of numbers. part i. *Royal Society of London Philosophical Transactions Series A*, 187 :619–673, 1896.

[MK06] L. Mutafchiev and E. Kamenov. Asymptotic formula for the number of plane partitions of positive integers. *Dokladi na Blgarskata Akademiya na Naukite. Comptes Rendus de l'Académie Bulgare des Sciences*, 59(4) :361–366, 2006.

[MN07] T. Maeda and T. Nakatsu. Amoebas and instantons. *International Journal of Modern Physics A. Particles and Fields. Gravitation. Cosmology. Nuclear Physics*, 22(5) :937–983, 2007.

[NW78] A. Nijenhuis and H. S. Wilf. *Combinatorial algorithms*. Academic Press Inc., New York, second edition, 1978.

[OR03] A. Okounkov and N. Reshetikhin. Correlation function of schur process with application to local geometry of a random 3-dimensional young diagram. *Journal of the American Mathematical Society*, 16 :581–603, 2003.

[OR07] A. Okounkov and N. Reshetikhin. Random skew plane partitions and the pearcey process. In *Communications in Mathematical Physics*, volume 269, pages 571–609, February 2007.

[Ott48] R. Otter. The number of trees. *Annals of Mathematics*, 49(3) :583–599, 1948.

[Oud07] J. Oudinet. Uniform random walks in very large models. In *RT '07 : Proceedings of the 2nd international workshop on Random testing*, pages 26–29, Atlanta, GA, USA, November 2007. ACM Press.

[Pak02] I. Pak. Hook length formula and geometric combinatorics. *Séminaire Lotharingien de Combinatoire*, 46 :Art. B46f, 13 pp. (electronic), 2001/02.

[Pak06] I. Pak. Partition bijections, a survey. *The Ramanujan Journal*, 12(1) :5–75, 2006.

[PR87] G. Pólya and R. C. Read. *Combinatorial Enumeration of Groups, Graphs and Chemical Compounds*. Springer Verlag, New York, 1987.

[Pro97] J. Propp. Generating random elements of finite distributive lattices. *Electronic Journal of Combinatorics*, 4(2) :Research Paper 15, approx. 12 pp. (electronic), 1997. The Wilf Festschrift (Philadelphia, PA, 1996).

[PS09] K. Panagiotou and A. Steger. Maximal biconnected subgraphs of random planar graphs. In *SODA*. SIAM, 2009. To appear.

[PSS08] C. Pivoteau, B. Salvy, and M. Soria. Newton iteration for combinatorial systems with applications to enumeration and random generation. In preparation, 2008.

[PSSar] C. Pivoteau, B. Salvy, and M. Soria. Boltzmann oracle for combinatorial systems. In *Algorithms, Trees, Combinatorics and Probabilities*. Discrete Mathematics and Theoretical Computer Science, To appear. Proceedings of the Fifth Colloquium on Mathematics and Computer Science. Blaubeuren, Germany. September 22-26, 2008.

[PW07] K. Panagiotou and A. Weißl. Properties of random graphs via boltzmann samplers. In Philippe Jacquet, editor, *Analysis of Algorithms 2007 (AofA07)*, Discrete Mathematics and Theoretical Computer Science Proceedings, 2007. In press.

[Rem85] J.-L. Remy. Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *RAIRO Informatique Théorique.*, 19(2) :179–195, 1985.

[Rou08] O. Roussel. Génération aléatoire par le modéle de boltzmann pour l'opérateur «boîte». Master's thesis, Master Parisien de Recherche en Informatique, 2008.

[Sch33] G. Schulz. Iterative Berechnung der reziproken Matrix. *Zeitschrift für angewandte Mathematik und Physik*, 13 :57–59, 1933.

[SJ89] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1) :93–133, 1989.

[vdH02] J. van der Hoeven. Relax, but don't be too lazy. *Journal of Symbolic Computation*, 34(6) :479–542, 2002.

[Ver96] A. M. Vershik. Statistical mechanics of combinatorial partitions, and their limit configurations. *Functional Analysis and its Applications*, 30(2) :90–105, 1996.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, New York, NY, USA, 1999.

[Wil77] H. S. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24 :281–291, 1977.

[Wil97] D. B. Wilson. Determinant algorithms for random planar structures. In *SODA '97 : Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 258–267, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[Wri31] E. M. Wright. Asymptotic partition formulae, i : Plane partitions. *The Quarterly Journal of Mathematics. Oxford. Second Series*, Ser. 2 :177–189, 1931.

[You01] A. Young. On quantitative substitutional analysis. *Proceedings of the London Mathematical Society*, 33 :97–146, 1901.

[Zei96] D. Zeilberger. Proof of the alternating sign matrix conjecture. *Electronic Journal of Combinatorics*, 3(2) :Research Paper 13, approx. 84 pp. (electronic), 1996. The Foata Festschrift.

[Zim94a] P. Zimmermann. Gaïa : a package for the random generation of combinatorial structures. *MapleTech*, 1(1) :38–46, 1994.

[Zim94b] P. Zimmermann. Random generation of unlabelled combinatorial structures (summary by E. Muray). In Bruno Salvy, editor, *Algorithms Seminar, 1993–1994*, volume 2381 of *Research Report*. Institut National de Recherche en Informatique et en Automatique, October 1994.