N° d'ordre: 1430

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par Isabelle DUTOUR

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Grammaires d'objets : énumération, bijections et génération aléatoire

---

Soutenue le : 19 janvier 1996

Après avis de :	Philippe Flajolet	Rapporteurs
	Christophe Reutenauer	

Devant la Commission d'examen formée de :

Maylis Delest	Professeur	Président et Rapporteur
Elena Barcucci	Professeur	Examinateurs
Bruno Courcelle	Professeur	
Jean-Marc Fédou	Maître de Conférences	
Philippe Flajolet	Directeur de Recherche à l'INRIA	
Christophe Reutenauer	Professeur	

_ 1996_

Maylis Delest me fait un grand honneur en acceptant de présider ce jury. Elle a toujours suivi mes travaux avec intérêt et m'a souvent apporté conseils, suggestions et encouragements. Qu'elle trouve ici toute l'expression de ma reconnaissance.

L'enthousiasme avec lequel Jean-Marc Fédou a dirigé mon travail a contribué de façon essentielle à l'élaboration de cette thèse. Sa disponibilité et ses nombreux conseils m'ont été très précieux tout au long de mes recherches. Il m'a été très agréable de travailler avec lui, et je suis heureuse de pouvoir lui exprimer ici toute ma gratitude.

Deux rapporteurs m'ont fait l'honneur de lire ce mémoire avec attention : Philippe Flajolet, dont les travaux sur la génération aléatoire ont inspiré le dernier chapitre de ce document, et Christophe Reutenauer que j'ai eu le plaisir de rencontrer lors de son récent séjour à Bordeaux. Je les remercie du temps qu'il ont consacré à la lecture de mon travail, et de leur participation à ce jury.

Je suis très sensible à la présence d'Elena Barcucci dans ce jury. Je la remercie tout particulièrement pour son chaleureux accueil lors de mon séjour à Florence.

Je tiens à exprimer ma gratitude à Bruno Courcelle pour l'intérêt qu'il a porté à l'égard de mon travail. Sa présence dans ce jury m'honore et je l'en remercie.

Je suis heureuse de pouvoir remercier ici Xavier Viennot pour l'attention qu'il a accordé à mon travail, ainsi que pour ses conseils et ses idées. Mes remerciements s'adressent également aux autres membres de l'équipe Combinatoire et Algorithmique du LaBRI pour leur soutien et leurs conseils avisés. Les cours de Robert Cori sont à l'origine de mon intérêt pour l'informatique théorique et la combinatoire, intérêt que Jean-Guy Penaud a su prolonger en dirigeant mon mémoire de DEA. Je remercie tout particulièrement Mireille Bousquet-Mélou dont les qualités scientifiques et humaines ont toujours été pour moi d'un grand secours.

Je tiens à manifester toute ma reconnaissance à Renzo Pinzani pour m'avoir invitée à l'Université de Florence ainsi que pour le charmant accueil qu'il m'y a reservé en compagnie d'Elena Barcucci et d'Alberto Del Lungo.

Merci à Nadine Rouillon et Alain Denise, mes anciens compagnons de bureau, pour leur soutien et leurs encouragements qui n'ont jamais cessés, même après leur départ. Quelle belle technique que la messagerie électronique!

A ces remerciements, j'associe les amis et les doctorants qui m'ont accompagnée durant la préparation de cette thèse.

Un très grand merci enfin à Michel qui m'a épaulée et supportée dans des conditions parfois difficiles. Je suis heureuse de lui dédier cette thèse.

Table des Matières

Introduction

Le travail présenté dans ce document se situe dans le cadre de la combinatoire énumérative et bijective. Nous formalisons la notion de *grammaires d'objets*, descriptions récursives d'objets combinatoires qui sont exprimées sous forme de dessins. Nous montrons que l'utilisation des grammaires d'objets permet souvent une résolution systématique de problèmes variés liés à l'énumération, à la construction de bijections, ou encore à la génération aléatoire uniforme d'objets.

On trouve dans la littérature combinatoire divers exemples d'utilisations de décompositions visuelles, qui permettent en particulier de résoudre des problèmes d'énumération. Afin de "développer nos intuitions sur les fonctions génératrices", Graham, Knuth et Patashnik [63] exploitent des schémas tels que celui Figure 1 pour calculer la fonction génératrice selon leur longueur des rectangles de hauteur 2 construits à l'aide de dominos 2×1 et 1×2 ; cette fonction est la série génératrice des nombres de Fibonacci. Un autre exemple simple est la décomposition classique des arbres binaires complets (Figure 2) à partir de laquelle on obtient directement la fonction génératrice selon le nombre de sommets, la série génératrice des nombres de Catalan. Dans [38], Delest et Dulucq énumèrent les polyominos verticalement convexes dirigés selon le périmètre et l'aire en utilisant la méthodologie dite *de Schützenberger* codant les polyominos par des mots ; des représentations graphiques justifient la décomposition des polyominos directement liée à celle des mots.

La *méthodologie de Schützenberger*, ou *DSV*, exposée dans [86] et [87], est une méthode d'énumération utilisant les grammaires algébriques. Elle consiste tout d'abord à établir une bijection entre les objets que l'on cherche à énumérer et les mots d'un langage algébrique L, de telle sorte que le paramètre d'énumération pour un objet donné coïncide avec le nombre de

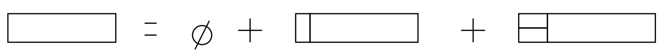


Figure 1: Les rectangles de hauteurs 2.

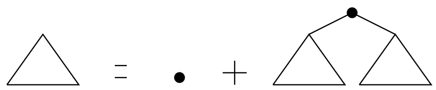


Figure 2: Les arbres binaires complets.

lettres du mot. Lorsque ce langage L est engendré par une grammaire algébrique non ambiguë, la série génératrice formelle des mots de L, définie par

$$\mathcal{L} = \sum_{w \in L} w \ ,$$

satisfait le système d'équations en variables non commutatives obtenu à partir de cette grammaire. Le passage à l'image commutative permet d'obtenir un système d'équations algébriques en variables commutatives, dont l'une des composantes de la solution est la série génératrice des mots de L et donc des objets en bijection avec les mots de L. Deux ouvrages de référence sur les séries formelles en variables non commutatives sont le livre de Salomaa et Soittola [84] (*Automata - theoretic aspects of formal power series*) et celui de Berstel et Reutenauer [15] (*Les séries rationnelles et leurs langages*). Comme le titre du second l'indique, les auteurs s'intéressent plus particulièrement aux séries formelles rationelles, pour lesquelles Reutenauer [81] a écrit un article de synthèse.
La méthode DSV permet de retrouver des résultats connus, en expliquant mieux l'algébricité des objets étudiés. Citons par exemple les travaux relatifs aux cartes planaires de Cori [26], Cori et Richard [27], Cori et Vauquelin [28], qui retrouvent et expliquent des formules d'énumération données par Tutte [88]. Le premier nouveau résultat obtenu par cette méthode est la série génératrice des polyominos convexes selon le périmètre par Delest et Viennot [42]. Bien d'autres ont suivi : voir par exemple [34, 38]. Un aperçu de l'étroite relation entre les problèmes d'énumération en combinatoire et les langages est donné dans [36, 33, 91]. De par sa définition, la méthode DSV ne peut s'appliquer directement qu'à des paramètres pour lesquels la série génératrice des objets est algébrique. Une extension est la *théorie des q-grammaires* développée tout d'abord par Delest et Fédou [39]. Cette méthode, inspirée des mécanismes de compilation et basée sur les grammaires d'attributs, permet essentiellement d'obtenir des équations non algébriques (q-équations) à partir d'un codage algébrique des objets. Delest et Fédou ont appliqué les q-grammaires en particulier à l'énumération des polyominos [40]. Ces travaux ont été prolongés par Dubernard [48].

En analysant le principe de la méthode DSV, ou de ses extensions, on s'aperçoit que le plus important ne réside pas dans la bijection entre les objets et les mots, mais plutôt dans la décomposition des objets qu'elle induit, voir par exemple [38]. D'où l'idée d'une méthode DSV "sans les mots", basée uniquement sur une description récursive des objets étudiés. Une *grammaire d'objets* est décrite à l'aide d'*ensembles d'objets*, d'*objets terminaux* et d'*opérations d'objets* définissant la construction des objets des différents ensembles à partir des terminaux. Une notion fondamentale pour l'application des grammaires d'objets à l'énumération d'objets combinatoires est la notion de grammaire d'objets *non ambiguë* (connue aussi pour les grammaires algébriques), qui signifie que chaque objet engendré par la grammaire l'est de façon unique. Nous représentons les grammaires d'objets non ambiguës le plus souvent par des équations graphiques, comme les Figures 1 et 2. Lorsque les paramètres d'énumération considérés s'interprètent sur les opérations à l'aide de *valuations d'objets linéaires* (resp. *q-linéaires*), nous pouvons déduire directement de la grammaire d'objets le système d'équations algébriques (resp. de q-équations) vérifié par la série génératrice des objets engendrés. Les valuations d'objets *linéaires* sont exactement le cadre d'application de la méthodologie DSV classique. Pour les systèmes de q-équations, il n'existe pas de méthode générale de résolution. Cependant, quelques cas particuliers sont traités dans la littérature, par exemple dans [19, 22, 23, 37, 49, 53, 79]. Nous donnons dans cette thèse une légère extension d'une méthode de résolution de Prellberg et Brak [79] grâce à laquelle nous traitons quelques exemples d'énumération (des familles de chemins de Motzkin et d'arbres planaires).

La définition des grammaires d'objets, qui étend le concept classique de grammaire de mots à un concept plus général de description récursive d'objets, s'intègre dans le cadre plus général de l'*Algèbre Universelle* [50, 95, 30] et celui des *magmas* ou *algèbres multi-sortes* ("many-sorted"). Les méthodes de l'Algèbre Universelle s'avèrent très utiles dans divers domaines de l'informatique théorique : l'étude de diverses propriétés de programmes (Guessarian [65]), d'arbres (Berstel et Reutenauer [14]), de grammaires de graphes (Courcelle [31, 29]), ou encore de toutes autres grammaires "context-free", comme cela est suggéré dans l'article fondamental de Mezei et Wright [75]. Dans le cadre des grammaires d'objets, notre souci est de rester le plus proche possible des objets manipulés; ainsi, nous ne retenons pas toute la rigueur des définitions de l'Algèbre Universelle. Par exemple, nous identifions les opérations d'objets et les ensembles d'objets avec les symboles qui les représentent. Avec les grammaires d'objets, notre approche est voisine d'autres travaux consistant à déduire des propriétés combinatoires à partir d'une description récursive des objets; citons en particulier la théorie des espèces [70], les travaux de Flajolet [55], ainsi que ceux de Flajolet, Salvy, Zimmermann sur le logiciel $\Lambda_\Upsilon\Omega$ [58, 60]. $\Lambda_\Upsilon\Omega$ est l'implémentation d'une méthode pour analyser automatiquement des algorithmes : la *méthode des séries génératrices*, décrite dans les travaux de thèse de Salvy [85] et Zimmermann [98] et dans l'article [59]. Les programmes analysés sont décrits à l'aide de *spécifications*, concept basé sur les notions de constructeur et de taille. Nous verrons plus loin dans cette introduction que Flajolet, Zimmermann et Van Cutsem [61] ont utilisé les spécifications dans le cadre de la génération aléatoire de structures combinatoires. Notons que dans ces approches, les constructions se font à partir d'*atomes*, et en utilisant les opérations d'*union*, de *produit*, d'*ensemble de*, de *cycle de*, ou encore de *suite de*.

Quelques objets combinatoires étudiés

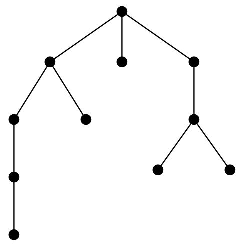
Nos exemples sont basés principalement sur trois familles fondamentales d'objets combinatoires : les *arbres*, les *chemins* et les *polyominos*.

Les arbres

L'*arbre* est l'une des structures de données les plus utilisées. En effet, la théorie des arbres a de nombreuses applications en informatique. Citons, par exemple, *le nombre de Strahler* calculé sur un arbre, appelé *nombre de registres* si l'arbre est binaire, et qui sert alors en compilation pour une évaluation efficace des expressions arithmétiques [56, 71]. Un raffinement de cette méthode est utilisé pour l'analyse et la synthèse d'images d'arbres [4]. Elle fait intervenir *la matrice de ramification* d'un arbre que l'on sait déterminer pour les arbres binaires aléatoires [78]. C'est également la structure d'arbre qui intervient dans les algorithmes de recherche ou de tri, l'analyse des programmes, l'organisation des fichiers dans les systèmes d'exploitation, ou encore la représentation des programmes traités par un compilateur.

Un *arbre*, ou *arbre planaire*, se définit naturellement de manière récursive : il est constitué d'un *sommet* distingué r, la *racine*, et d'un ensemble ordonné d'arbres $A_1, \ldots, A_n$ non vides. Les racines des arbres $A_1, \ldots, A_n$ sont les *fils* de r. De nombreux paramètres sont étudiés : le nombre de sommets (*taille*), le nombre d'arêtes, la *profondeur* et la *longueur du chemin interne*. Si on définit la *hauteur* d'un sommet d'un arbre comme le nombre d'arêtes le reliant à la racine, la profondeur de l'arbre est alors la hauteur maximale des sommets, et la longueur du chemin interne la somme des hauteurs des sommets (voir Figure 3).

Parmi toutes les classes d'arbres, une est particulièrement remarquable : celle des *arbres*



nombre de sommets : 11
nombre d'arêtes : 10
profondeur : 4
longueur du chemin interne : 22

Figure 3: Un arbre.

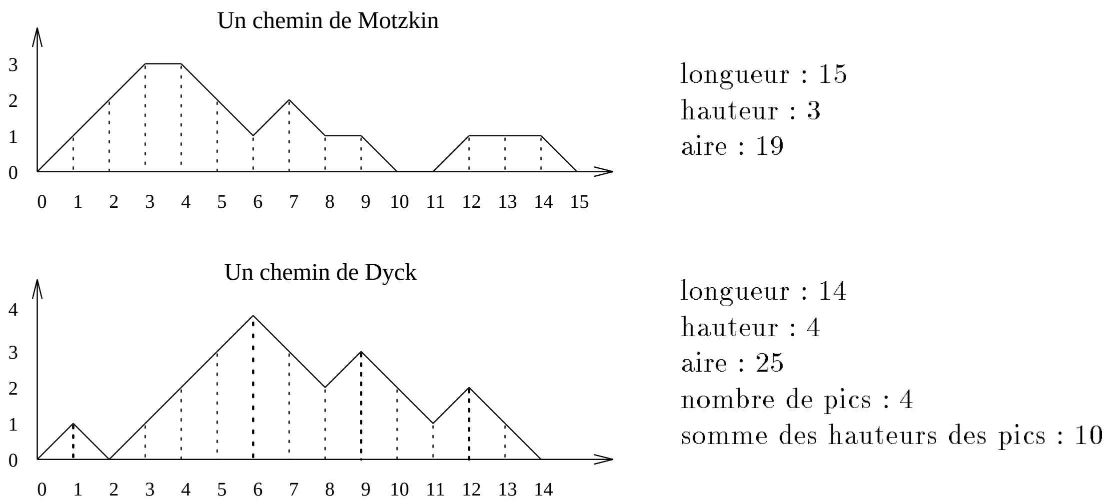


Figure 4: Un chemin de Motzkin et un chemin de Dyck.

binaires complets, arbres dont les sommets ont zéro ou deux fils.

Les chemins

Les *chemins*, de *Dyck* et de *Motzkin* surtout, jouent un rôle important dans plusieurs domaines de la combinatoire. Le lien entre fractions continues et chemins de Motzkin (ou Dyck) résulte de l'article fondateur de Flajolet [54], qui fournit une interprétation combinatoire de certaines fractions continues. Ces travaux sont prolongés par Roblet [82] qui met en place une théorie combinatoire pour l'étude des approximants de Padé (conditions d'existence, algorithmes de calculs ou liens avec les fractions continues). Ces chemins interviennent également dans des problèmes purement informatique, telle que l'évaluation d'algorithmes sur les fichiers par Flajolet, Françon, Vuillemin [56] ; les auteurs introduisent la notion d'*histoire de fichier*. Le concept de chemin est aussi essentiel dans la théorie combinatoire des polynômes orthogonaux développée par Viennot [89, 90]. Enfin, ces chemins, ou plutôt les mots qu'ils représentent, codent de nombreux objets combinatoires, et ils apparaissent donc dans beaucoup de travaux concernant l'énumération ou la génération aléatoire [91, 45, 7, 9].

Un *chemin* est défini dans le plan discret $\pi = \mathbb{Z} \times \mathbb{Z}$ comme une suite $\omega = (s_0, \ldots, s_n)$ de points de π, appelés *sommets*. La *longueur* de ω est n. Un *pas* est un couple (s_i, s_{i+1}), s_i et s_{i+1} étant les *extrémités* du pas. On définit des pas particuliers dans le quadrillage. Par exemple, si $s_i = (x, y)$, on dit que (s_i, s_{i+1}) est un pas

Nord,	si $s_{i+1} = (x, y+1)$,	Nord-Est,	si $s_{i+1} = (x+1, y+1)$,
Sud,	si $s_{i+1} = (x, y-1)$,	Sud-Est,	si $s_{i+1} = (x+1, y-1)$.
Est,	si $s_{i+1} = (x+1, y)$,		

Deux familles de chemins sont particulièrement remarquables, les *chemins de Motzkin* et les *chemins de Dyck*.

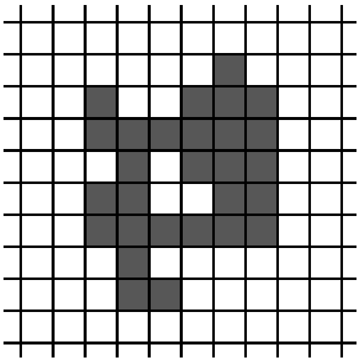
Les *chemins de Motzkin* sont des chemins de $\mathbb{N} \times \mathbb{N}$ formés de trois types de pas, Sud-Est, Est et Nord-Est, commençant sur l'axe des abscisses en $(0,0)$ et finissant également sur cet axe en $(k,0)$, où k est un entier positif quelconque. On peut définir les *chemins de Motzkin bicolorés* et plus généralement *k-colorés*, ce sont des chemins de Motzkin pour lesquels les pas Est peuvent prendre respectivement deux couleurs et k couleurs.

Les *chemins de Dyck* sont des chemins de $\mathbb{N} \times \mathbb{N}$ formés de deux types de pas, Sud-Est et Nord-Est, commençant sur l'axe des abscisses en $(0,0)$ et finissant également sur cet axe en $(2k,0)$, où k est un entier positif quelconque. Ce sont en fait les chemins de Motzkin qui n'ont pas de pas Est.

Les principaux paramètres étudiés sur ces familles de chemins sont la *longueur* qui est le nombre de pas, la *hauteur* qui est la hauteur maximale des extrémités des pas (distance à l'axe des abscisses) et l'*aire* qui est souvent définie comme la somme des hauteurs des extrémités des pas (voir Figure 4). Pour les chemins de Dyck, deux autres paramètres importants sont le nombre de pics et la somme des hauteurs des pics.

Les polyominos

Les *polyominos* sont des objets connus et étudiés depuis longtemps en combinatoire. Ils interviennent également dans l'étude de certains modèles de physique statistique sous une forme



largeur : 6
hauteur : 8
aire : 28
périmètre : 42

Figure 5: Un polyomino.

équivalente, les *animaux*; citons les travaux de Dhar [46, 47], Hakim et Nadal [67], Nadal, Derrida et Vannimenus [76]. Un tour d'horizon des problèmes combinatoires posés par la physique statistique est effectué par Viennot dans [92]. Le lien entre la percolation dirigée et l'énumération des polyominos est en particulier souligné dans les récents travaux de Bousquet-Mélou [21, 20].

Un *polyomino* est défini dans le plan $\mathbb{R} \times \mathbb{R}$, à translation près, comme une union finie de cellules élémentaires (carrés unitaires) d'intérieur connexe. Les paramètres les plus étudiés sur un polyomino sont l'*aire* qui est le nombre de cellules élémentaires qui le composent, la *largeur* (resp. *hauteur*) qui est le nombre de colonnes (resp. lignes) du polyomino, et le *périmètre* qui est le nombre de segments unitaires de son pourtour (voir Figure 5). Etablir une formule exacte pour l'énumération selon l'aire ou selon le périmètre des polyominos généraux reste à l'heure actuelle un problème ouvert. En revanche, de nombreux résultats ont été démontrés concernant l'énumération de certaines classes particulières de polyominos. On trouvera des articles de synthèse dans [35, 66, 93]. La plupart de ces classes sont définies par des contraintes relevant de deux notions : la *convexité* et l'existence d'une *direction privilégiée*. Un polyomino est *verticalement* (resp. *horizontalement*) *convexe* si son intersection avec chaque ligne verticale (resp. horizontale) est convexe. On dit qu'un polyomino est *convexe* s'il est à la fois verticalement et horizontalement convexe. Un polyomino est *dirigé* si chacune de ses cellules peut être atteinte à partir d'une cellule distinguée, la *racine*, par un chemin formé de pas Nord et Est. Parmi les polyominos qui sont à la fois convexes et dirigés, une classe remarquable et très étudiée est celle des *polyominos parallélogrammes* (voir par exemple [37, 51, 52]). Un polyomino parallélogramme est défini par un couple de chemins uniquement composés de pas Nord et Est, et ne s'intersectant qu'à leurs extrémités. Des exemples de diverses classes de polyominos sont donnés Figure 6.

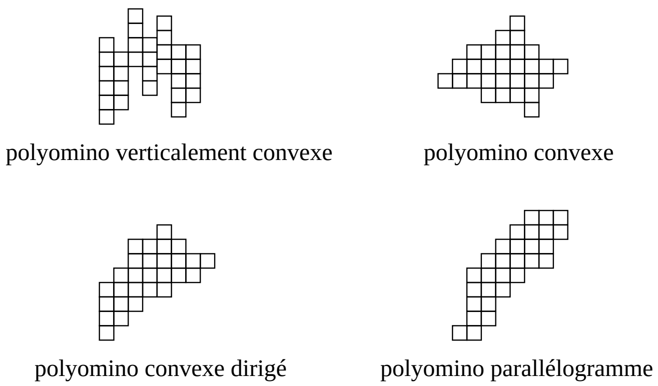


Figure 6: Polyominos de différentes classes.

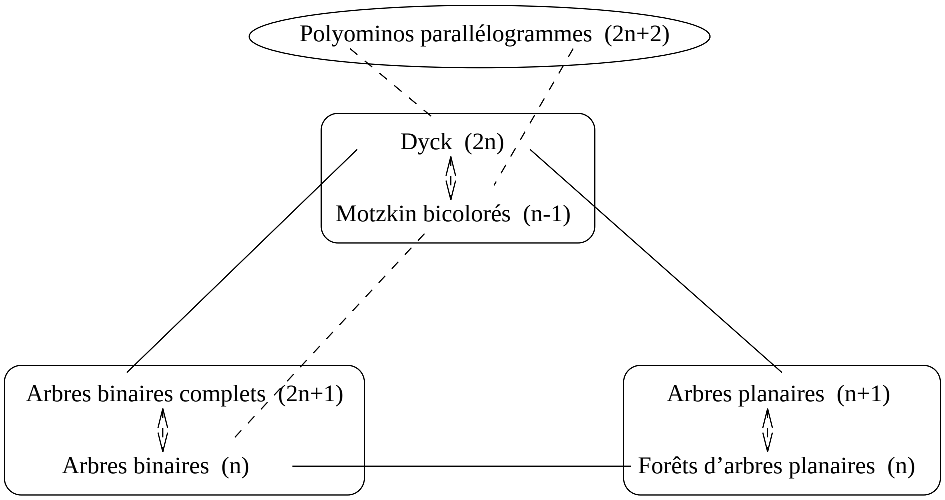


Figure 7: Le “Triangle d'or” des bijections.

Construction de bijections

Construire des bijections entre ensembles d'objets est souvent très utile pour l'énumération des objets selon certains paramètres. Par exemple, lorsque les paramètres d'énumération se lisent difficilement sur un ensemble d'objets, il est agréable de pouvoir se ramener par une bijection à un ensemble d'objets pour lequel les paramètres s'interprètent bien, et qui ainsi, est plus facile à énumérer. C'est le cas de la méthode DSV qui ramène l'énumération des objets considérés à celle des mots d'un langage algébrique. Citons quelques bijections devenues classiques en combinatoire [42, 91, 93], et qui font partie de ce que Viennot appelle le “Triangle d'or” des bijections (voir Figure 7) :

les mots de Dyck (longueur $2n$) et les arbres binaires complets ($2n+1$ sommets),

les mots de Dyck (longueur $2n$) et les arbres planaires ($n+1$ sommets),

les mots de Dyck (longueur $2n$) et les polyominos parallélogrammes (périmètre $2n+2$),

les arbres binaires (n sommets) et les forêts d'arbres planaires (n sommets).

Tous ces objets sont comptés par les nombres de Catalan,

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

Nous présentons dans cette thèse des méthodes de construction de bijections entre ensembles d'objets en utilisant des grammaires d'objets non ambiguës. Nous définissons pour cela la notion de grammaires d'objets *isomorphes*; ce sont des grammaires ayant la même “structure”, c'est-à-dire le même nombre d'ensembles d'objets, d'objets terminaux et d'opérations d'objets (de domaines de définition compatibles). Nous pouvons ainsi établir une correspondance entre les arbres de dérivation des grammaires d'objets et construire automatiquement une ou plusieurs bijections entre les objets engendrés. Il est en effet possible de décrire par ce procédé de nombreuses bijections connues, comme celles du "Triangle d'or". Par exemple, la bijection classique entre les polyominos parallélogrammes et les chemins de Dyck (la hauteur des colonnes correspond à la hauteur des pics) s'exprime simplement à l'aide des deux grammaires d'objets isomorphes présentées Figure 8. Nous exhibons également de cette manière une nouvelle bijection entre les polyominos convexes dirigés et les chemins du Grand Dyck (ou Dyck bilatère). Cette correspondance, comme celle de Bousquet-Mélou [18], permet de retrouver bijectivement le résultat étonnamment simple de Lin et Chang [72] : le nombre de polyominos convexes dirigés de périmètre $2n+4$ est égal au coefficient binomial $\binom{2n}{n}$.

Une question alors naturelle est la suivante : étant données deux grammaires d'objets quelconques, peut-on les transformer pour obtenir des grammaires isomorphes, tout en conservant les ensembles d'objets engendrés ? Et si oui, comment les construire ? L'étude d'une transformation sur les grammaires d'objets, que nous nommons S-itération, permet de répondre à cette question dans le cas des grammaires de dimension 1 (avec un seul ensemble d'objets). La S-itération est l'analogue d'une transformation connue dans d'autres domaines comme les systèmes de réécritures, les techniques d'évaluation partielle et les transformations de programmes et de systèmes réguliers : le *dépliage* ("unfolding"). On peut par exemple se référer aux articles de Burstall et Darlington [24] et de Courcelle [30]. Une question importante dans ces travaux est la correction du système obtenu par dépliage, c'est à dire l'équivalence au système original. Ce genre de problèmes ne se pose pas pour la S-itération de grammaires d'objets car elle n'effectue qu'une simple réécriture de la grammaire d'objets initiale. Ainsi, les grammaires d'objets construites engendrent les mêmes ensembles d'objets, et sont donc équivalentes à la grammaire initiale.

La S-itération nous permet de déduire le théorème fondamental suivant : à partir de deux

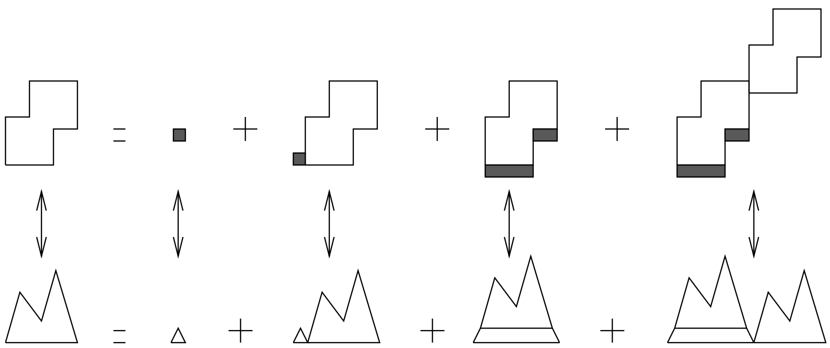


Figure 8: Polyominos parallélogrammes et chemins de Dyck.

grammaires d'objets non ambiguës de dimension 1 ayant au moins une opération d'objets d'arité ≥ 2, on peut construire deux grammaires d'objets isomorphes engendrant les mêmes ensembles d'objets que les grammaires initiales. En particulier, l'ensemble des mots de Dyck, engendré par la grammaire classique $D = \epsilon + xD\overline{x}D$, entre dans ce cadre. Ainsi, cela implique que, pour tous les objets qui peuvent se décrire par une grammaire d'objets non ambiguë de dimension 1 et de degré ≥ 2, nous savons construire une bijection avec les mots de Dyck en exhibant des grammaires d'objets isomorphes. Ce résultat donne un élément de réponse à cette remarque interrogative faite notamment par Viennot dans [91] : "Pourquoi les mots de Dyck jouent-ils un rôle si important dans le codage des objets combinatoires?". Il vient effectivement confirmer le fait que beaucoup d'objets sont étroitement liés aux mots de Dyck.

De nombreuses questions restent encore ouvertes. Le théorème que nous avons établi concerne uniquement les grammaires d'objets de dimension 1. Qu'en est-il pour les grammaires de dimension supérieure? Nous pouvons également définir d'autres transformations sur les grammaires d'objets. Est-ce qu'une combinaison de la S-itération avec d'autres formes d'itérations permettrait de construire des grammaires d'objets isomorphes plus simples, et donc ainsi exploitables pour la détermination des paramètres conservés par les bijections?

Génération aléatoire uniforme

Une autre application intéressante des grammaires d'objets est la *génération aléatoire uniforme* d'objets combinatoires, *uniforme* signifiant que tous les objets de même "taille" ont la même probabilité d'être tirés.

Notre travail se situe dans le cadre de la *méthode récursive*, procédé de génération formalisé par Nijenhuis et Wilf [77]. Cette technique permet de générer récursivement des objets aléatoires à partir de relations de récurrence les énumérant et d'interprétations constructives de ces relations. De nombreux objets combinatoires sont ainsi tirés aléatoirement : partages d'entiers, partitions d'ensembles [77], arbres [3, 80], graphes [97], polyominos [69], etc. Hickey et Cohen [68] ont les premiers appliqué et formalisé cette méthode dans le cas particulier des langages algébriques. Des améliorations ont été apportées par Flajolet, Goldwurm, Steyaert [57], Mairson [73], et Goldwurm [62]. L'efficacité de cette technique est également améliorée par Denise [43] pour la génération aléatoire des mots de langages rationnels. On retrouve une adaptation de la *méthode récursive* dans les travaux de Greene [64] sur les langages formels étiquetés. Enfin, récemment, Flajolet, Zimmermann et Van Cutsem [61] ont donné une approche systématique de cette méthode pour des objets (étiquetés ou non) *décomposables*, c'est à dire que l'on peut décrire récursivement à partir d'*atomes* et à l'aide des opérations d'*union*, de *produit*, de construction d'*ensembles*, de *suites* et de *cycles*. Ils montrent qu'à partir d'une telle description, qu'ils nomment *spécification*, on peut construire automatiquement un programme de génération aléatoire uniforme des objets considérés selon le nombre d'atomes. L'implémentation des calculs avancés développés dans [61] a été réalisée dans *Gaïa*, un "package" *Maple* conçu par Zimmermann [99], puis adapté par Murray à la dernière version de *Maple* (*Maple V Release 3*) sous le nom de *CombStruct*.

Une alternative de la *méthode récursive* souvent utilisée est la *méthode de rejet*. Celle-ci consiste à générer aléatoirement des objets dans un surensemble de la classe considérée, jusqu'à obtenir un objet de cette classe. Cette méthode est couramment utilisée pour tirer aléatoirement divers objets combinatoires : graphes (McKay, Wormald [74, 97]), arbres (Alonso [2]), animaux et polyominos (Barcucci, Pinzani, Sprugnoli [9], Hochstättler, Loebl, Moll [69]), mots (Barcucci, Pinzani, Sprugnoli [9, 11], Berstel, Pocchiola [13], Denise [44], Wacrenier [94]), etc.

Notre travail développe une étude inspirée de celle de Flajolet, Zimmermann et Van Cutsem [61] sur les spécifications ; nous l'adaptons aux grammaires d'objets auxquelles sont associées des valuations d'objets linéaires et q-linéaires. A partir d'une grammaire d'objets non ambiguë G et d'un ensemble de valuations d'objets G-linéaire ou G-q-linéaire, nous savons construire automatiquement, pour tous les ensembles d'objets de G, les procédures d'énumération et les procédures de génération uniforme selon les valuations d'objets. Nous avons réalisé un "package" *Maple*, *qAlGO*, implémentant ces résultats. *qAlGO* vient en complément de *CombStruct*. En effet, *CombStruct* permet le tirage aléatoire d'objets construits à l'aide d'*unions*, de *produits*, de constructions de *cycles*, d'*ensembles*, ou de *suites*, selon le paramètre *nombre d'atomes*. De ce point de vue, *qAlGO* traite uniquement les unions et les produits (opérations d'objets), mais il permet la génération aléatoire d'objets combinatoires selon divers paramètres. En particulier, il rend accessible la génération aléatoire d'objets selon des paramètres *non algébriques* : par exemple l'*aire* pour les différentes classes de polyominos et de chemins, ou la *longueur du chemin interne* pour les arbres.

A titre d'exemple, nous avons réalisé quelques études expérimentales sur certaines classes de polyominos générés selon l'aire. L'assistance du logiciel *CalICo* (Calcul et Image en Combinatoire) développé par notre équipe au LaBRI a été très confortable pour cette étude. En effet, *CalICo* [41, 83] offre un environnement logiciel de manipulations et visualisations d'objets combinatoires ; il permet notamment de faire communiquer des interfaces graphiques spécifiques pour chaque type d'objet et des logiciels de calcul formel tels que *Maple* ou *Mathematica*. Nous pouvons ainsi très facilement visualiser les polyominos que nous générons à l'aide de *Maple* et *qAlGO* grâce à l'interface graphique *XAnimal* de *CalICo*. Cela nous a permis de remarquer, par exemple Figure 9, combien sont effilés les polyominos convexes dirigés tirés uniformément selon l'aire ! Un calcul expérimental (sur 1000 tirages) fournit comme valeurs :

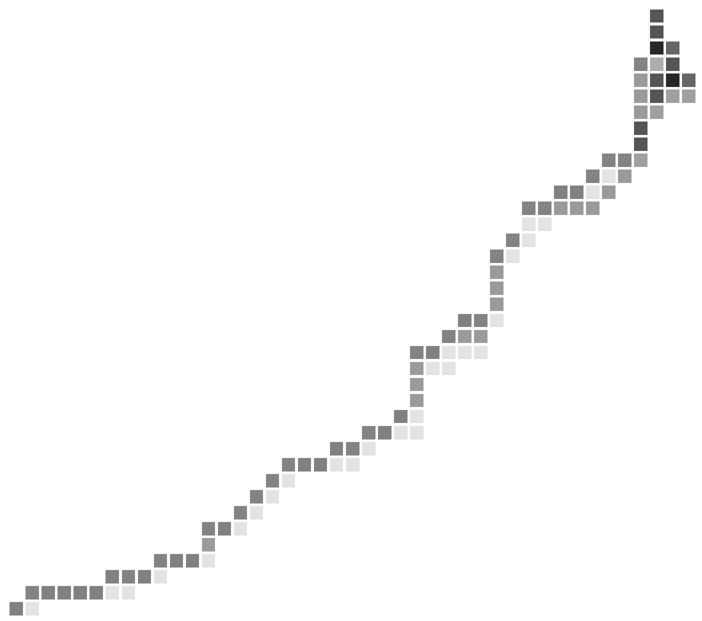


Figure 9: Un polyomino convexe dirigé aléatoire d'aire 100.

- $2,37$ pour la hauteur moyenne d'une colonne,
- $1,37$ pour le nombre moyen de contacts entre deux colonnes,

ce qui vient effectivement confirmer l'allure effilée des polyominos. Nous obtenons les mêmes valeurs pour les polyominos parallélogrammes d'aire 200. Nous retrouvons effectivement le résultat de Bender [12] pour la hauteur moyenne d'une colonne $(2,37)$ qu'il a calculée par des méthodes d'analyse asymptotique.

Nous pouvons imaginer l'étude de bien d'autres paramètres et objets combinatoires. Le caractère automatique de *qAlGO* facilite ce type d'expérimentation.

Plan de la thèse

Le Chapitre 1 - Grammaire d'objets - présente les définitions et propriétés concernant les grammaires d'objets. En particulier, il établit différentes formes normales pour ces grammaires, et introduit la notion de non ambiguïté d'une grammaire d'objets. Il donne également les deux notations utilisées le plus souvent pour décrire une grammaire d'objets : équationnelle et schématique.

Le Chapitre 2 - Enumérations - s'intéresse à l'application des grammaires d'objets à l'énumération d'objets combinatoires. Il définit le concept de valuations d'objets et établit le théorème principal d'énumération, qui consiste à déduire directement d'une grammaire d'objets non ambiguë un système d'équations vérifiées par les fonctions génératrices des différents ensembles d'objets, selon des valuations d'objets données. Les équations obtenues sont algébriques pour des valuations linéaires, et ce sont des q-équations pour des valuations q-linéaires. Enfin, ce chapitre étend une méthode de résolution de q-équations et l'applique à quelques exemples d'énumérations.

Le Chapitre 3 - Bijections - donne des méthodes de construction de bijections entre ensembles d'objets en utilisant des grammaires d'objets non ambiguës. Pour cela, il introduit tout d'abord la définition du système caractéristique de polynômes associé à une grammaire d'objets qui caractérise la structure de la grammaire. Puis, il définit la notion fondamentale du chapitre qui est celle de grammaires d'objets isomorphes, grammaires de même structure et qui fournissent ainsi des bijections. L'étude d'une transformation des grammaires d'objets, la S-itération, permet d'énoncer le théorème suivant : à partir de deux grammaires d'objets non ambiguës de dimension 1 ayant au moins une opération d'objets d'arité ≥ 2, on peut construire deux grammaires d'objets isomorphes engendrant les mêmes ensembles d'objets que les grammaires initiales.

Le Chapitre 4 - Génération aléatoire - est consacré à l'application des grammaires d'objets à la génération aléatoire uniforme d'objets combinatoires. Une méthode complète est décrite pour construire automatiquement les procédures d'énumération et de génération d'objets à partir d'une grammaire d'objets non ambiguë et de valuations d'objets linéaires ou q-linéaires. Cette technique est implémentée en *Maple* sous le nom de *qAlGO*. Ainsi, la génération aléatoire d'objets selon des paramètres non algébriques est accessible à l'aide de *qAlGO*. A titre d'exemple, nous effectuons quelques études expérimentales sur des polyominos tirés uniformément selon l'aire.

L'Annexe A - Le Programme Maple qAlGO - décrit les fonctionnalités de *qAlGO* ainsi que la syntaxe *Maple* à utiliser pour les grammaires d'objets et les valuations d'objets.

Chapitre 1

Grammaires d'objets

C'est de la *théorie des langages*, et plus particulièrement des *langages* et *grammaires algébriques* que sont inspirés les concepts définissant les grammaires d'objets. Une grammaire algébrique G est un quadruplet $< \mathcal{N}, \mathcal{T}, \mathcal{P}, S >$ comprenant

$$\begin{cases} \text{un ensemble } \mathcal{N} \text{ de non-terminaux,} \\ \text{un ensemble } \mathcal{T} \text{ de terminaux,} \\ \text{un ensemble } \mathcal{P} \text{ de productions dans } \mathcal{N} \times (\mathcal{N} \cup \mathcal{T})^*, \\ \text{un symbole } S \text{ dans } \mathcal{N}\text{, l'axiome.} \end{cases}$$

Un *mot* engendré par G est uniquement composé de terminaux, il est dans $\mathcal{T}^*$; il est construit à partir de l'axiome S, par une suite finie de *dérivations* (ou *réécritures*) des non-terminaux grâce aux productions. Cette suite de dérivations est souvent représentée par un *arbre de dérivation.* L'ensemble des mots engendrés à partir de S est appelé *langage algébrique* engendré par G, et noté $L_G(S)$. Un exemple très classique de langage algébrique utilisé en combinatoire est le langage de *Dyck* qui code de nombreuses structures telles que des arbres, des chemins ou des polyominos. Ces mots sont engendrés par la grammaire algébrique suivante :

$$G_D = < \{D\}, \{\epsilon, x, \overline{x}\}, \{D \to \epsilon, D \to xD\overline{x}D\}, D >,$$

notée plus simplement

$$D \to \epsilon \mid xD\overline{x}D \ .$$

Cette grammaire est *non ambiguë*, c'est à dire que chaque mot est engendré par un unique arbre de dérivation.

Le formalisme que nous établissons pour les *grammaires d'objets* étend le concept classique de grammaire à un concept plus général de *description récursive d'objets*, à l'aide d'*ensembles d'objets*, d'*objets terminaux* et d'*opérations d'objets.* Il correspond dans le cadre plus général de l'*Algèbre Universelle* [50, 95, 30] à celui des *magmas* ou *algèbres multi-sortes* ("many-sorted"). Si S est un ensemble de symboles appelés *sortes*, on note $seq(S)$ l'ensemble des suites $(s_1, s_2, ..., s_n)$ d'éléments de S, et on appelle S-*signature*, ou *signature*, un ensemble de symboles de fonctions F muni de deux applications

- $\alpha : F \rightarrow seq(S)$; l'*arité* d'un symbole de fonction f est la suite $\alpha(f)$, le *rang* de f est la longueur de $\alpha(f)$,
- $\sigma : F \rightarrow S$; le *genre* (sorte) d'un symbole de fonction f est le symbole $\sigma(f)$.

Un F-magma consiste à associer à chaque élément de S un ensemble, et à chaque élément de F une opération. Plus précisément, un F-magma est un ensemble $M =< (M_s)_{s\in S}, (f_M)_{f\in F} >$ où pour chaque f de F, f_M est une application $f_M : M_{\alpha(f)} \rightarrow M_{\sigma(f)}$, où $M_{\alpha(f)}$ désigne le produit cartésien $M_{f_1} \cdot M_{f_2} \cdot \ldots \cdot M_{f_n}$ lorsque $\alpha(f) = (f_1, .., f_n)$. Les applications f_M sont appelées les opérations de M. Cette différenciation entre les symboles qui représentent les opérations et les opérations elles-mêmes est primordiale pour les problèmes issus de problèmes de spécifications algébriques, puisqu'elle permet de distinguer très précisément la syntaxe de la sémantique. En ce qui concerne les grammaires d'objets, nous ne faisons pas cette distinction car notre souci est de rester le plus proche possible des objets manipulés, donc de la sémantique.

Après quelques définitions préliminaires, nous définissons précisément les grammaires d'objets en Section 1.2. La troisième section est une justification de l'analogue de nos notations avec celles de la syntaxe abstraite. Nous présentons ensuite différentes formes normales pour les grammaires d'objets, ainsi que la notion de grammaire d'objets *non ambiguë*; ce sont des propriétés très importantes qui seront utiles pour nos applications. La sixième section propose des notations agréables lorsque les grammaires d'objets sont non ambiguës. Enfin, nous terminons ce chapitre par quelques exemples de grammaires d'objets.

1.1 Définitions et notations préliminaires

On se place dans une famille d'ensembles d'objets que l'on note $\mathcal{E}$.

Définition 1.1 *On appelle <u>opération d'objets</u> dans $\mathcal{E}$ toute application $\phi : E_1 \times \ldots \times E_k \longrightarrow E$ où $E \in \mathcal{E}$ et $E_i \in \mathcal{E}$ pour i dans $[1, k]$.*

Une opération d'objets décrit la façon dont on obtient un objet de E à partir de k objets appartenant respectivement à $E_1, \ldots, E_k$.

Notations : On note $dom(\phi) = E_1 \times \ldots \times E_k$ le domaine de définition de ϕ, et $dest(\phi) = E$ la destination de ϕ qui est un ensemble contenant $Im(\phi)$, l'image de $dom(\phi)$ par ϕ. On dit que chaque E_i est une composante de $dom(\phi)$.

Définition 1.2 *On dit qu'un arbre étiqueté A est un arbre d'opérations d'objets sur l'ensemble E s'il peut se décrire récursivement de la manière suivante :*

- *soit A est réduit à une feuille étiquetée par un élément de E,*
- *soit la racine de A a k fils ordonnés et est étiquetée par une opération d'objets ϕ d'arité k, avec $dest(\phi) = E$ et $dom(\phi) = E_1 \times \ldots \times E_k$, où le sous-arbre issu du $i^{ème}$ fils est un arbre d'opérations d'objets sur l'ensemble E_i, pour tout i dans $[1, k]$.*

Les sommets internes d'un arbre d'opérations d'objets sont étiquetés par des opérations d'objets et les feuilles par des objets.

Définition 1.3 *Soit un arbre A d'opérations d'objets sur un ensemble E. Son expression d'objets associée $exp(A)$ est obtenue en concaténant les étiquettes de ses nœuds dans l'ordre préfixe, et son évaluation $ev(A)$ est l'objet de E défini récursivement de la façon suivante :*

1. *Si A est réduit à une feuille étiquetée o, alors $exp(A) = o$ et $ev(A) = o$,*
2. *Sinon, la racine de A est étiquetée par une opération d'objets ϕ et a k sous-arbres $A_1, \ldots, A_k$; alors $exp(A) = \phi\, exp(A_1) \ldots exp(A_k)$ et $ev(A) = \phi(ev(A_1), \ldots, ev(A_k))$.*

Remarque : $exp(A)$ est simplement un mot, une concaténation de symboles, alors que $ev(A)$ est un objet. C'est la même nuance que dans le domaine de l'Algèbre Universelle entre un terme et son évaluation.

Un exemple : les polyominos parallélogrammes

Notons E_{pp} l'ensemble des polyominos parallélogrammes. Rappelons que l'on peut définir un polyomino parallélogramme par un couple de chemins uniquement composés de pas Nord et Est, et ne s'intersectant qu'à leurs extrémités (voir Introduction et Figure 10).

Voici l'exemple de trois opérations d'objets dans $\mathcal{E} = \{E_{pp}\}$ décrites schématiquement Figures 11, 12 et 13 :

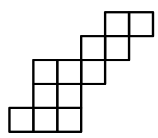


Figure 10: Un polyomino parallélogramme.

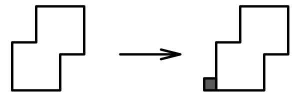


Figure 11: L'opération d'objets ϕ_1.

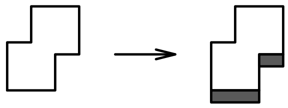


Figure 12: L'opération d'objets ϕ_2.

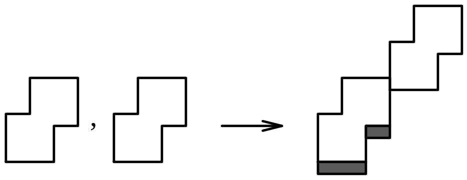
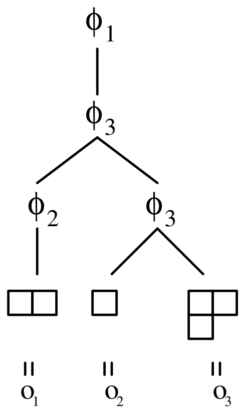


Figure 13: L'opération d'objets ϕ_3.



L'expression d'objets associée :

$\phi_1\,\phi_3\,\phi_2\,o_1\,\phi_3\,o_2\,o_3$

Figure 14: Un arbre d'opérations d'objets sur E_{pp}.

- $\phi_1 : E_{pp} \longrightarrow E_{pp}$ (*extension vers l'ouest*). Opération qui prend en argument un polyomino et qui colle une cellule unitaire à l'ouest de la cellule du bas de la première colonne.

- $\phi_2 : E_{pp} \longrightarrow E_{pp}$ (*extension vers le sud*). Opération qui prend en argument un polyomino et qui lui ajoute une cellule unitaire en bas de chaque colonne.

- $\phi_3 : E_{pp} \times E_{pp} \longrightarrow E_{pp}$ (*collage asymétrique*). Opération qui prend en argument deux polyominos, applique ϕ_2 au premier, et les colle de façon à ce que la cellule du haut de la dernière colonne du premier polyomino coïncide avec la cellule du bas de la première colonne du deuxième.

Avec ces trois opérations d'objets, on peut construire des arbres d'opérations d'objets sur E_{pp}, ainsi que les expressions d'objets associées. Un exemple est donné Figure 14. L'évaluation de l'arbre de la Figure 14 est le polyomino parallélogramme de la Figure 10.

1.2 Grammaires d'objets : définition

Définition 1.4 *Une grammaire d'objets est un quadruplet $< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ où :*

$\mathcal{E}$	$= \{E_i\}_{i\in I}$	*est une famille finie d'ensembles d'objets. (I partie finie de* $\mathbb{N}$*)*
$\mathcal{T}$	$= \{T_{E_i}\}_{i\in I}$	*est une famille finie de sous-ensembles finis d'éléments de $\mathcal{E}$, $T_{E_i} \subset E_i$, dont les éléments sont appelés objets* terminaux.
$\mathcal{P}$		*est un ensemble d'opérations d'objets ϕ dans $\mathcal{E}$.*
S		*est un élément désigné de $\mathcal{E}$, appelé* axiome *de la grammaire.*

Remarque : On appellera parfois grammaire d'objets un triplet $< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ en se permettant de ne préciser qu'après coup quel élément de $\mathcal{E}$ est choisi comme axiome.

Définition 1.5 *La dimension d'une grammaire d'objets est le nombre d'ensembles d'objets contenus dans $\mathcal{E}$.*

Exemple 1.1 : On note $\square$ la cellule unitaire d'un polyomino. Voici deux exemples de grammaires d'objets de dimension 1 :

$$G_1 =< \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2\}, E_{pp} > \quad \text{et} \quad G_2 =< \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2, \phi_3\}, E_{pp} > \quad .$$

Remarque : Dans la théorie des magmas en Algèbre Universelle, une différence fondamentale est faite entre les opérations et ensembles d'une part, et les symboles qui les représentent d'autre part. Dans le cadre des grammaires d'objets, nous ne prenons pas en compte une telle distinction. Les notations en seraient inutilement compliquées pour nos applications.

Définition 1.6 *Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets et soit $E \in \mathcal{E}$ un ensemble d'objets. Un arbre de dérivation de G sur E est un arbre d'opérations d'objets sur E tel que :*

1. *ses nœuds internes sont étiquetés par des opérations d'objets de $\mathcal{P}$,*

2. *ses feuilles par des objets terminaux.*

On note A_G l'ensemble des arbres de dérivation de G, et E_G l'ensemble des expressions d'objets associées.

Définition 1.7 *Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets. Un objet o est engendré dans G par un ensemble E de $\mathcal{E}$ lorsqu'il existe un arbre de dérivation de G sur E dont l'évaluation est o.*

L'*ensemble des objets engendrés* dans G par E, que l'on note $\mathcal{O}_G(E)$, est un sous-ensemble de E. Si S de $\mathcal{E}$ est désigné comme l'axiome de G, on dit que $\mathcal{O}_G(S)$ est l'ensemble d'objets engendrés par G.

Exemple 1.2 : Le polyomino parallélogramme de la Figure 10 appartient à $\mathcal{O}_{G_2}(E_{pp})$. Un arbre de dérivation dans G_2 pour le parallélogramme de la Figure 10 est donné Figure 15. On remarque qu'à la différence avec l'arbre de la Figure 14, on ne trouve ici aux feuilles que l'objet terminal.
$\mathcal{O}_{G_1}(E_{pp})$ est l'ensemble des diagrammes de Ferrers, sous-ensemble de polyominos parallélogrammes, et $\mathcal{O}_{G_2}(E_{pp})$ est l'ensemble des polyominos parallélogrammes tout entier.

Définition 1.8 *Deux grammaires d'objets sont dites équivalentes lorsqu'elles engendrent le même ensemble d'objets.*

1.3 Grammaires d'objets et syntaxe abstraite

Les arbres de dérivation d'une grammaire d'objets sont équivalents à ce qu'on appelle dans le domaine de la compilation les *arbres abstraits*. Un arbre abstrait est une représentation compacte de l'*arbre syntaxique*, qui est l'arbre de dérivation de l'expression dans la grammaire algébrique représentant le langage. (voir par exemple [1])

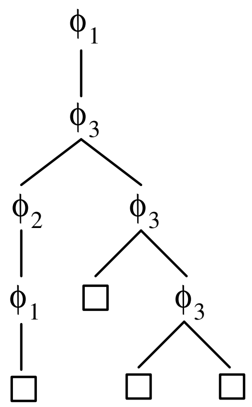


Figure 15: Un arbre de dérivation dans G_2.

Le choix d'utiliser des arbres de type abstrait comme arbres de dérivation dans les grammaires d'objets permet de mieux refléter la structure hiérarchique naturelle d'application des opérations d'objets pour la construction des objets à partir des terminaux. On peut cependant retrouver sous-jacente une grammaire algébrique associée à une grammaire d'objets en transformant les arbres abstraits en arbres syntaxiques.

Proposition 1.9 *Pour toute grammaire d'objets G, l'ensemble des expressions E_G d'objets associées aux arbres de dérivation dans G est un langage algébrique.*

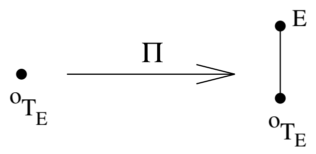
PREUVE :

Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ une grammaire d'objets. Pour montrer que E_G est un langage algébrique, on montre qu'il est engendré par une grammaire algébrique $\Gamma(G) =< N, X, \mathcal{R}, S >$ définie par $N = \{E/E \in \mathcal{E}\}$, $X = \bigcup_{T \in \mathcal{T}} \{o_T/o_T \in T\} \cup \{\phi/\phi \in \mathcal{P}\}$, S l'axiome de G reste l'axiome de $\Gamma(G)$, et $\mathcal{R}$ contient les règles

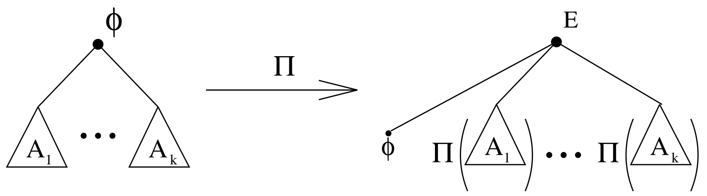
- $E \longrightarrow \phi E_1 \dots E_k$,
  pour tout $\phi \in \mathcal{P}$, $\phi : E_1 \times \dots \times E_k \longrightarrow E$,
- $E \longrightarrow o_{T_E}$,
  pour tout $o_{T_E} \in T_E$ et pour tout $T_E \in \mathcal{T}$.

En effet, ceci est la conséquence de la correspondance classique [1] entre arbres abstraits (arbres de dérivation de G) et arbres syntaxiques (arbres de dérivation de $\Gamma(G)$). On note cette correspondance $\Pi : A_G \longrightarrow \mathcal{D}_{\Gamma(G)}$.

- si $A \in A_G$ est réduit à une feuille $o_{T_E} \in T_E$, alors $\Pi(A)$ est l'arbre à deux sommets de racine E et d'unique feuille o_{T_E}.



- si la racine de A est l'opération ϕ d'arité k, elle a k sous-arbres $A_1, \ldots, A_k$. Alors $\Pi(A)$ est l'arbre de racine E ayant $k+1$ fils : une feuille étiquetée ϕ et les k sous-arbres $\Pi(A_1), \ldots, \Pi(A_k)$.



La lecture de gauche à droite des feuilles des arbres de dérivation de $\Gamma(G)$ donne, de par leur définition, les expressions d'objets associées aux arbres de dérivation de G. Ainsi, $\Gamma(G)$ engendre bien E_G.

□

1.4 Formes normales

Il est souvent utile de pouvoir transformer une grammaire d'objets pour la décrire sous une forme adaptée à la résolution d'un problème donné. Nous présentons dans cette section trois formes normales pour les grammaires d'objets : *réduite*, *forme 1-2* et *complète*. Les formes *réduite* et *forme 1-2* sont des extensions de formes normales usuelles pour les grammaires algébriques : réduite et forme normale de Chomsky. La forme *complète* est spécifique aux grammaires d'objets.

Nous donnons pour chacune des trois formes normales la construction permettant d'obtenir, à partir d'une grammaire d'objets de forme quelconque, une grammaire d'objets équivalente en forme normale.

1.4.1 Grammaires d'objets réduites

Nous étendons ici aux grammaires d'objets la notion de grammaires algébriques réduites (voir par exemple [5]).

Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets. Nous définissons les notions d'ensemble d'objets *productif* dans G et d'ensemble d'objets *accessible* dans G.

Définition 1.10 *Soit E un élément de $\mathcal{E}$. On dit que E est <u>productif</u> dans G lorsque $\mathcal{O}_G(E) \neq \emptyset$.*

En d'autres termes, un ensemble d'objets E est productif lorsqu'il engendre au moins un objet dans G.

Définition 1.11 *Soient E et F deux éléments de $\mathcal{E}$. On dit que E est <u>accessible à partir de F</u> dans G lorsqu'il existe un arbre de dérivation sur F qui a un nœud étiqueté par une opération d'objets dont le domaine de définition a pour composante E.*
On dit que E est <u>accessible</u> dans G s'il est accessible à partir de l'axiome.

Définition 1.12 *G est dite <u>réduite</u> lorsque tout ensemble E de $\mathcal{E}$ est productif et accessible dans G.*

Proposition 1.13 *Pour toute grammaire d'objets G, il existe une grammaire d'objets réduite équivalente, que l'on peut construire à partir de G.*

<u>PREUVE :</u>
Pour prouver la proposition, nous prouvons deux lemmes qui correspondent aux deux conditions de la définition d'une grammaire d'objets réduite.

Lemme 1.14 *Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ une grammaire d'objets. On peut décider, pour tout $E \in \mathcal{E}$, si E est productif ou non.*

<u>PREUVE :</u>
On fait la construction suivante :

- $\mathcal{E}_0 = \{E \in \mathcal{E} \ / \ T_E \neq \emptyset, T_E \in \mathcal{T}\}$,
- $\mathcal{E}_i = \mathcal{E}_{i-1} \bigcup \{E \in \mathcal{E} \ / \ \exists \phi \in \mathcal{P}, dest(\phi) = E$ et chaque composante de $dom(\phi) \in \mathcal{E}_{i-1}\}$, pour $i > 0$,
- $\mathcal{E}_{total} = \bigcup_{i \geq 0} \mathcal{E}_i$.

Pour prouver le lemme, il suffit de prouver les deux assertions suivantes :

1. $\mathcal{E}_{total}$ est la famille d'ensembles d'objets qui sont productifs,
2. on peut calculer $\mathcal{E}_{total}$: on l'obtient en un nombre borné d'opérations.

La technique de démonstration de ces deux assertions est similaire à celle développée dans le cas des grammaires algébriques [5].

1. Il est clair que les éléments de $\mathcal{E}_{total}$ sont des ensembles d'objets productifs. Montrons que $\mathcal{E}_{total}$ les contient tous. Soit E dans $\mathcal{E}$ un ensemble d'objets productif. Alors il existe un objet o dans $\mathcal{O}_G(E)$ et un arbre de dérivation A_o dans A_G tel que $ev(A_o) = o$. Si L est la profondeur de A_o, alors E est dans $\mathcal{E}_{L-1}$, qui est inclus dans $\mathcal{E}_{total}$, et donc $E \in \mathcal{E}_{total}$.

2. Les $\mathcal{E}_i$ sont croissants et inclus dans l'ensemble fini $\mathcal{E}$. La suite devient donc stationnaire. Pour calculer $\mathcal{E}_{total}$, on calcule donc les $\mathcal{E}_i$ successifs tant qu'ils sont strictement croissants, et l'on s'arrête dès que deux d'entre eux sont égaux.

Ainsi, à partir d'une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$, si $\mathcal{E}'$ désigne la famille d'ensembles d'objets qui sont productifs, on peut construire la grammaire $G' =< \mathcal{E}', \mathcal{T}', \mathcal{P}' >$ où $\mathcal{P}' = \{\phi \in \mathcal{P} \;/\; dest(\phi) \in \mathcal{E}'$ et chaque composante de $dom(\phi) \in \mathcal{E}'\}$. On qualifie de pré-réduite la grammaire d'objets G'.

□

Il suffit donc maintenant de prouver qu'à partir d'une grammaire d'objets pré-réduite, on peut construire une grammaire d'objets réduite équivalente.

Lemme 1.15 *Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ une grammaire d'objets. On peut décider, pour tout $E \in \mathcal{E}$, si E est accessible à partir de l'axiome S ou non.*

<u>PREUVE :</u>
On fait la construction suivante :

- $\mathcal{F}_0 = \{S\}$,
- $\mathcal{F}_i = \mathcal{F}_{i-1} \bigcup \{E \in \mathcal{E} \;/\; \exists E' \in \mathcal{F}_{i-1}$ et $\exists \phi \in \mathcal{P}$ tels que $dest(\phi) = E'$ et E est une composante de $dom(\phi)\}$, pour $i > 0$,
- $\mathcal{F}_{total} = \bigcup_{i \geq 0} \mathcal{F}_i$.

Pour prouver le lemme, il suffit de prouver les deux assertions suivantes :

1. $\mathcal{F}_{total}$ est la famille d'ensembles d'objets accessibles à partir de S,
2. on peut calculer $\mathcal{F}_{total}$: on l'obtient en un nombre borné d'opérations.

La démonstration est similaire à celle développée pour le Lemme 1.14.

Ainsi, à partir d'une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$, si $\mathcal{E}'$ désigne la famille d'ensembles d'objets accessibles à partir de S, on peut construire la grammaire $G' =< \mathcal{E}', \mathcal{T}', \mathcal{P}', S >$ où $\mathcal{P}' = \{\phi \in \mathcal{P} \;/\; dest(\phi) \in \mathcal{E}'\}$.

□

En enchainant les deux constructions, nous obtenons donc, à partir d'une grammaire d'objets quelconque, une grammaire d'objets réduite équivalente.

$\square$

Exemple 1.3 : Soit une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ telle que

$$\begin{array}{rcl} \mathcal{E} & = & \{E_1, E_2, E_3, E_4\} \\ \mathcal{T} & = & \{\emptyset, \{o_2\}, \emptyset, \{o_4\}\} \\ S & = & E_1 \\ \mathcal{P} & = & \left\{ \begin{array}{l} \phi_1 : E_2 \rightarrow E_1 \ , \ \psi_1 : E_2 \times E_3 \rightarrow E_1 \ , \ \theta_1 : E_3 \times E_4 \rightarrow E_1 \\ \phi_2 : E_3 \rightarrow E_2 \ , \ \psi_2 : E_2 \rightarrow E_2 \\ \phi_3 : E_3 \rightarrow E_3 \ , \ \psi_3 : E_3 \times E_3 \rightarrow E_3 \\ \phi_4 : E_2 \rightarrow E_4 \ , \ \psi_4 : E_1 \times E_2 \rightarrow E_4 \end{array} \right. \end{array}$$

Appliquons la construction du Lemme 1.14 :

$$\mathcal{E}_0 = \{E_2, E_4\}, \text{ puis } \mathcal{E}_1 = \{E_1, E_2, E_4\} = \mathcal{E}_{total}$$

et on obtient $G_1 =< \mathcal{E}_1, \mathcal{T}_1, \mathcal{P}_1, S >$ où

$$\begin{array}{rcl} \mathcal{E}_1 & = & \{E_1, E_2, E_4\} \\ \mathcal{T}_1 & = & \{\emptyset, \{o_2\}, \{o_4\}\} \\ S & = & E_1 \\ \mathcal{P}_1 & = & \left\{ \begin{array}{l} \phi_1 : E_2 \rightarrow E_1 \\ \psi_2 : E_2 \rightarrow E_2 \\ \phi_4 : E_2 \rightarrow E_4 \ , \ \psi_4 : E_1 \times E_2 \rightarrow E_4 \end{array} \right. \end{array}$$

Appliquons alors la construction du Lemme 1.15 :

$$\mathcal{F}_0 = \{E_1\}, \text{ puis } \mathcal{F}_1 = \{E_1, E_2\} = \mathcal{F}_{total}$$

et on obtient $G_2 =< \mathcal{E}_2, \mathcal{T}_2, \mathcal{P}_2, S >$ où

$$\begin{array}{rcl} \mathcal{E}_2 & = & \{E_1, E_2\} \\ \mathcal{T}_2 & = & \{\emptyset, \{o_2\}\} \\ S & = & E_1 \\ \mathcal{P}_2 & = & \left\{ \begin{array}{l} \phi_1 : E_2 \rightarrow E_1 \\ \psi_2 : E_2 \rightarrow E_2 \end{array} \right. \end{array}$$

La grammaire d'objets G_2 est la grammaire d'objets réduite équivalente de G.

Remarque : Comme dans le cas des grammaires algébriques, l'ordre d'application de ces deux constructions est important, il est nécessaire de les effectuer dans l'ordre décrit ci-dessus. En effet, sur l'exemple précédent, si nous effectuons les constructions dans l'ordre inverse, nous obtenons $\mathcal{F}_{total} = \{E_1, E_2, E_3, E_4\}$, puis $\mathcal{E}_{total} = \{E_1, E_2, E_4\}$. Ainsi, E_4 reste dans la grammaire finale alors qu'il n'est pas accessible depuis l'axiome E_1, et la grammaire d'objets obtenue n'est pas réduite.

1.4.2 Grammaires d'objets en forme 1-2

Cette notion étend aux grammaires d'objets l'usuelle forme normale de Chomsky pour les grammaires algébriques. Cette forme 1-2 est très utile pour les problèmes de génération aléatoire abordés au Chapitre 4.

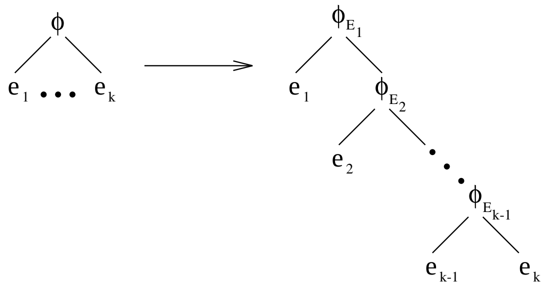
Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ une grammaire d'objets.

Définition 1.16 *On dit que G est en forme 1-2 si toutes les opérations d'objets de $\mathcal{P}$ sont uniquement d'arité* 1 *ou* 2*.*

Proposition 1.17 *Pour toute grammaire d'objets G, il existe une grammaire d'objets en forme 1-2 équivalente, que l'on peut construire à partir de G.*

PREUVE :
Il suffit de découper les domaines de définition de toutes les opérations d'objets d'arité > 2. Pour chaque opération d'objets d'arité $k > 2$, $\phi : E_1 \times \ldots \times E_k \longrightarrow E$,

- on rajoute :
  - ⋆ $k-2$ ensembles d'objets, F_{E_i} pour $i = 2 \ldots k-1$:
$$\begin{cases} F_{E_{k-1}} & = & E_{k-1} \times E_k \\ F_{E_i} & = & E_i \times F_{E_{i+1}} \quad , i = 2 \ldots k-2 \end{cases}$$
  - ⋆ $k-2$ opérations d'objets identité, ϕ_{E_i} pour $i = 2 \ldots k-1$:
$$\begin{cases} \phi_{E_{k-1}} & : & E_{k-1} \times E_k & \longrightarrow & F_{E_{k-1}} \\ \phi_{E_i} & : & E_i \times F_{E_{i+1}} & \longrightarrow & F_{E_i} \quad , i = 2 \ldots k-2 \end{cases}$$
- on remplace ϕ par $\phi_{E_1} : E_1 \times F_2 \longrightarrow E$ définie comme suit :
  soit o dans $E_1 \times F_{E_2}$, alors il existe $(e_1, \ldots, e_k)$ dans $E_1 \times \ldots \times E_k$ de telle sorte que $o = (e1, (e_2, \ldots, (e_{k-1}, e_k) \ldots))$, et alors $\phi_{E_1}(o) = \phi(e_1, \ldots, e_k)$:



□

Exemple 1.4 : Soit une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ telle que

$$\begin{array}{rcl} \mathcal{E} & = & \{E_1, E_2, E_3\} \\ \mathcal{T} & = & \{\{o_1\}, \emptyset, \{o_3\}\} \\ S & = & E_1 \\ \mathcal{P} & = & \left\{ \begin{array}{l} \phi_1 : E_2 \times E_3 \to E_1 \ , \ \psi_1 : E_1 \times E_2 \times E_3 \to E_1 \\ \phi_2 : E_1 \to E_2 \\ \phi_3 : E_2 \times E_1 \times E_3 \to E_3 \end{array} \right. \end{array}$$

La grammaire d'objets en forme 1-2 équivalente est $G_1 =< \mathcal{E}_1, \mathcal{T}_1, \mathcal{P}_1, S >$ où

$$\begin{array}{rcl} \mathcal{E}_1 & = & \{E_1, E_2, E_3, F_{E_2}, F_{E_1}\} \\ \mathcal{T}_1 & = & \{\{o_1\}, \emptyset, \{o_3\}, \emptyset, \emptyset\} \\ S & = & E_1 \\ \mathcal{P}_1 & = & \left\{ \begin{array}{l} \phi_1 : E_2 \times E_3 \to E_1 \ , \ \psi_{E_1} : E_1 \times F_{E_2} \to E_1 \\ \phi_2 : E_1 \to E_2 \\ \phi_{E_2} : E_2 \times F_{E_1} \to E_3 \\ \psi_{E_2} : E_2 \times E_3 \to F_{E_2} \\ \phi_{E_1} : E_1 \times E_3 \to F_{E_1} \end{array} \right. \end{array}$$

1.4.3 Grammaires d'objets complètes

C'est une notion spécifique aux grammaires d'objets. En général, pour une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$, on a $\mathcal{O}_G(E) \subseteq E$ pour tout E dans $\mathcal{E}$. Il se peut en effet que certains objets d'un ensemble ne soit pas engendrés par cet ensemble dans G ($\mathcal{O}_G(E) \subset E$). L'idée est alors de se ramener à une grammaire d'objets pour laquelle les ensembles d'objets intervenant dans la grammaire sont exactement ceux engendrés par la grammaire.

Définition 1.18 *G est dite complète lorsque pour tout E dans $\mathcal{E}$, $\mathcal{O}_G(E) = E$.*

Pour écrire une grammaire d'objets G sous une forme complète, il suffit de remplacer chaque ensemble d'objets E par $\mathcal{O}_G(E)$ dans $\mathcal{E}$, ainsi qu'au sein des définitions des opérations d'objets.

Exemple 1.5 : La grammaire $G_1 =< \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2\} >$ n'est pas complète, car elle n'engendre pas tous les polyominos parallélogrammes, mais les diagrammes de Ferrers F. Une grammaire complète est $< \{F\}, \{\{\square\}\}, \{\phi_1, \phi_2\} >$ où les opérations d'objets ϕ_1 et ϕ_2 sont restreintes à F.

Remarque : Les grammaires d'objets que nous étudierons par la suite seront écrites sous une forme complète et réduite.

1.5 Notion de non-ambiguïté

Comme pour les grammaires algébriques, la non-ambiguïté est une propriété indispensable dès que l'on s'intéresse aux problèmes d'énumération.
Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ une grammaire d'objets.

Définition 1.19 *On dit que G est <u>non ambiguë</u> si, pour tout ensemble $E \in \mathcal{E}$, tout objet de $\mathcal{O}_G(E)$ est engendré par un unique arbre de dérivation sur E.*

Remarquons que lorsque G est réduite, il suffit de vérifier que tout objet de $\mathcal{O}_G(S)$ est engendré par un unique arbre de dérivation sur S pour vérifier que G est non ambiguë.

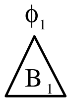
Un critère utile pour montrer la non-ambiguïté d'une grammaire d'objets est donné par la proposition suivante :

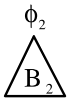
Proposition 1.20 *Une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ est non ambiguë lorsqu'elle vérifie les deux propriétés suivantes :*

1. *Pour tout $\phi \in \mathcal{P}$, ϕ est une opération d'objets injective.*

2. *Pour tout $E \in \mathcal{E}$, tous les $Im(\phi)$ tels que $dest(\phi) = E$ sont deux à deux disjoints ; chacun des $Im(\phi)$ est également disjoint de $T_E \in \mathcal{T}$, l'ensemble des objets terminaux de E.*

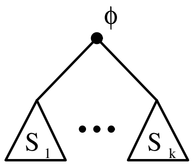
<u>PREUVE :</u>
Soit $E \in \mathcal{E}$ et soit $o \in \mathcal{O}_G(E)$. Supposons que $o = ev(A_1) = ev(A_2)$, alors on montre par récurrence sur la profondeur L de A_1 que les arbres de dérivation A_1 et A_2 sont égaux.
Le cas $L = 0$ est trivial (A_1 et A_2 sont réduits à un sommet étiqueté o), et on suppose que l'hypothèse est vraie pour $L \leq l$. On considère A_1 et A_2 de profondeur $L = l + 1$

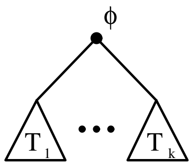
$A_1 =$ 

et $A_2 =$ 

.

On a $o = \underbrace{ev(A_1)}_{\in Im(\phi_1)} = \underbrace{ev(A_2)}_{\in Im(\phi_2)}$, la propriété (2) implique alors que $\phi_1 = \phi_2 = \phi$, et donc

$A_1 =$ 

et $A_2 =$ 

.

La propriété (*1*) implique que $\forall i \in [1,k]$, $ev(S_i) = ev(T_i)$. Comme les sous-arbres S_i sont de profondeur $\leq l$, par hypothèse de récurrence $S_i = T_i$ pour tout i dans $[1,k]$, et donc $A_1 = A_2$.

□

La vérification des propriétés *1* et *2* de la proposition 1.20 est suffisante pour montrer la non-ambiguïté d'une grammaire d'objets, mais elle n'est pas nécessaire. En effet, pour tout ensemble E de $\mathcal{E}$, les $Im(\phi)$ tels que $dest(\phi) = E$ ne sont pas obligatoirement inclus dans $\mathcal{O}_G(E)$, c'est à dire que certains objets des $Im(\phi)$ peuvent ne pas être engendrés dans G par E. La non vérification des propriétés *1* et *2* de la Proposition 1.20 à l'extérieur de l'ensemble $\mathcal{O}_G(E)$ n'interfère donc pas sur la non-ambiguïté de la grammaire d'objets. De cette remarque, nous déduisons la proposition suivante :

Proposition 1.21 *Si G est une grammaire d'objets réduite et complète, alors la vérification des propriétés 1 et 2 de la proposition 1.20 est une condition nécessaire et suffisante à la non-ambiguïté de G.*

Exemple 1.6 : La grammaire $G_2 =< \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2, \phi_3\} >$ est une grammaire non ambiguë pour l'ensemble des polyominos parallélogrammes E_{pp}. En effet, si P est un polyomino parallélogramme, nous avons les cas disjoints suivants :

- soit P est la cellule de base $\square$,
- soit la première colonne de P ne contient qu'une seule cellule : $P \in Im(\phi_1)$,
- soit on peut enlever à chaque colonne de P une cellule en bas, tout en gardant un ensemble de cellules connexe : $P \in Im(\phi_2)$,
- soit, en allant de gauche à droite, on peut enlever une cellule en bas des colonnes de P jusqu'à ce que deux colonnes ne soient adjacentes que par une seule cellule, P se décompose alors en deux polyominos parallélogrammes : $P \in Im(\phi_3)$.

Ainsi, E_{pp} est la réunion disjointe $\{\square\} \bigcup Im(\phi_1) \bigcup Im(\phi_2) \bigcup Im(\phi_3)$. Or les opérations d'objets ϕ_1, ϕ_2, ϕ_3 sont injectives. Donc, la grammaire G_2 est non ambiguë.

1.6 Notations équationnelle et schématique

Lorsqu'une grammaire d'objets est complète et non ambiguë, on peut la décrire sous une forme plus explicite qu'un simple quadruplet : un système d'équations (ensembles, objets, opérations d'objets), ou encore un système d'équations graphiques. Chaque équation donne la décomposition d'un ensemble d'objets en une union disjointe des objets terminaux et des images des opérations d'objets.

Notation équationnelle

Une grammaire d'objets complète et non ambiguë $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ s'écrit comme un système d'équations Σ de la forme :

$$\Sigma = \{E_1 = p_1(E_1, \ldots, E_n), \ldots, E_n = p_n(E_1, \ldots, E_n)\}$$

où pour tout $i \in [1, n]$

- E_i est un élément de $\mathcal{E}$,
- p_i est une somme de monômes qui peuvent être
  - soit un objet terminal appartenant à T_{E_i},
  - soit une expression de la forme $\phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}})$ où $\phi \in \mathcal{P}$, avec $dest(\phi) = E_i$ et $dom(\phi) = E_{i_{1_\phi}} \times \ldots \times E_{i_{k_\phi}}$,

$$p_i(E_1, \ldots, E_n) \quad = \quad \sum_{e_i \in T_{E_i}} e_i \quad + \sum_{dest(\phi)=E_i} \phi(E_{i_{1_\phi}}, \ldots, E_{ik_\phi}) \ .$$

D'un point de vue ensembliste, le signe + représente l'union disjointe et l'expression $\phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}})$ l'image de ϕ, $Im(\phi)$.

Par exemple, la grammaire des polyominos parallélogrammes que nous avons vue précédemment peut s'écrire simplement :

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

Remarque : La notation équationnelle des grammaires d'objets peut être interprétée en terme de système polynomial dans le cadre de l'Algèbre Universelle. Les problèmes d'unicité de la solution d'un tel système souvent étudiés dans ce domaine ne se posent pas ici, car nous nous intéressons seulement à des grammaires d'objets complètes et non ambiguës.

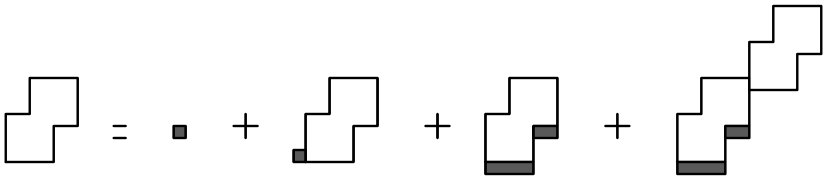


Figure 16: Grammaire schématique des polyominos parallélogrammes.

Représentation schématique

Représenter schématiquement une grammaire d'objets consiste à remplacer, dans sa notation équationnelle, les objets, ensembles d'objets et opérations d'objets par un “croquis” lié à leur représentation graphique. Cette notation est très souvent adoptée ([17, 19, 38, 63, 79]) car elle fournit en général le moyen le plus simple pour décrire l'action des opérations d'objets. Un exemple est donné Figure 16 pour les polyominos parallélogrammes.

1.7 Quelques exemples de grammaires d'objets

1.7.1 Les grammaires de mots

Les grammaires algébriques sont un cas particulier des grammaires d'objets, les seules opérations d'objets utilisées sont des concaténations de mots.

Proposition 1.22 *Tout langage algébrique peut être engendré par une grammaire d'objets.*

<u>Preuve :</u>
Soit L un langage algébrique, et soit $A_L =< A, V, \mathcal{R}, S >$ une grammaire algébrique engendrant L. Alors, la grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ définie comme suit engendre le langage L :

- $\mathcal{E} = A = \{E_i\}_{i \in I}$,
- on garde le même axiome $S \in \mathcal{E}$,
- $\mathcal{T} = \{T_i\}_{i \in I}$ où $T_i = \{\alpha \;/\; E_i \rightarrow \alpha \in \mathcal{R} \; et \; \alpha \in V^*\}$, c'est à dire l'ensemble des membres droits des règles de $\mathcal{R}$ dont toutes les lettres sont dans V (ou égaux au mot vide ϵ) et dont le membre gauche est E_i.
- $\mathcal{P}$: pour toute règle de $\mathcal{R}$ de la forme

$$E \longrightarrow \mu_1 \, E_1 \, \mu_2 \, E_2 \, \ldots \, \mu_n \, E_n \, \mu_{n+1} \quad \text{où} \quad \begin{array}{l} E \in A \;,\; E_i \in A \quad i \in [1,n] \\ \mu_j \in V^* \quad j \in [1, n+1] \end{array}$$

on définit :

$$\begin{array}{llllll}
\phi : & E_1 \times \ldots \times E_n & \longrightarrow & E & & \\
& (e_1\,, \ldots\,, e_n) & \longmapsto & \mu_1\, e_1\, \mu_2\, e_2\, \ldots\, \mu_n\, e_n\, \mu_{n+1} & où & E \in \mathcal{E} \\
& & & & & E_i \in \mathcal{E}\ ,\ e_i \in E_i \quad i \in [1, n] \\
& & & & & \mu_j \in V^* \quad j \in [1, n+1]
\end{array}$$

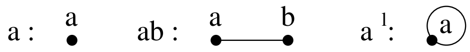
Comme pour la proposition 1.9, la correspondance s'établit entre les arbres de dérivation dans G (arbres abstraits) et ceux dans A_L (arbres syntaxiques).

□

1.7.2 Les grammaires de graphes

Les motivations de l'étude des grammaires de graphes sont nombreuses. On trouve des applications dans divers domaines tels que la génération et reconnaissance de familles de graphes, la reconnaissance de forme, la logique et le calcul distribué.

Prenons l'exemple des grammaires de graphes HR (Hyperedge Remplacement) étudiées par Courcelle dans [32] pour leurs liens avec la logique monadique du second ordre. Les graphes considérés ici sont des *s-graphes*. Ils ont des sommets distingués que l'on appelle les *sources*. Chaque source est étiquetée différemment (s désigne la correspondance entre les étiquettes et les sommets distingués). Différentes opérations sont définies sur ces s-graphes : la *composition parallèle*, la *restriction des sources*, et le *renommage des sources*. Les s-graphes de base sont :



Avec les s-graphes de base et les opérations citées ci-dessus (ainsi que des compositions), on peut écrire des grammaires de graphes, que l'on nomme des grammaires de graphes HR, qui sont des grammaires d'objets.

Exemple 1.7 : Les graphes *série-parallèles* à deux sources, étiquetées a et b.

Deux opérations sont utilisées (voir Figure 17) :

- la composition parallèle : //
  union de deux s-graphes en fusionnant les sources de même étiquette,
- la composition série : •
  concaténation de deux s-graphes en fusionnant la source étiquetée b du premier avec la source étiquetée a du second ; ce sommet "fusion" devient alors un sommet interne du graphe résultant.

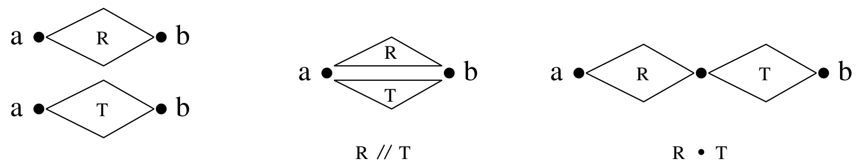


Figure 17: Les opérations // et $\bullet$ sur les graphes.

L'ensemble S des graphes série-parallèles peut être défini par la grammaire d'objets suivante

$$< \{S\}, \{ab\}, \{//, \bullet\} > .$$

Cette grammaire d'objets est clairement ambiguë et peut donc difficilement être exploitée pour l'énumération.

1.7.3 Les polyominos convexes dirigés

La classe des polyominos convexes dirigés permet de donner l'exemple d'une grammaire d'objets comprenant plusieurs ensembles d'objets, c'est à dire de dimension > 1. La Figure 18 exhibe une grammaire d'objets pour l'ensemble des polyominos convexes dirigés E_{pcd} et fait intervenir deux autres ensembles d'objets : les polyominos convexes dirigés marqués E_{pcdm} (une cellule de la dernière colonne est marquée), et les polyominos parallélogrammes E_{pp}. Elle est la représentation schématique de la grammaire :

$$< \{E_{pcd}, E_{pcdm}, E_{pp}\}, \{\{\square\}, \emptyset, \{\square\}\}, \{\theta_1, \theta_2, \theta_3, \psi_1, \psi_2, \psi_3, \phi_1, \phi_2, \phi_3\}, E_{pcd} > .$$

Les opérations d'objets sont décrites graphiquement Figure 18 :

pour les polyominos convexes dirigés :

$$\theta_1 : E_{pcd} \longrightarrow E_{pcd}, \qquad \theta_2 : E_{pp} \times E_{pcd} \longrightarrow E_{pcd}, \qquad \theta_3 : E_{pcd} \longrightarrow E_{pcd}.$$

pour les polyominos convexes dirigés marqués :

$$\psi_1 : E_{pcd} \longrightarrow E_{pcdm}, \qquad \psi_2 : E_{pcdm} \longrightarrow E_{pcdm}, \qquad \psi_3 : E_{pp} \times E_{pcdm} \longrightarrow E_{pcdm}.$$

pour les polyominos parallélogrammes :

$$\phi_1 : E_{pp} \longrightarrow E_{pp}, \qquad \phi_2 : E_{pp} \longrightarrow E_{pp}, \qquad \phi_3 : E_{pp} \times E_{pp} \longrightarrow E_{pp}.$$

Les opérations d'objets $\theta_1, \theta_2, \psi_2$ et ψ_3 se définissent de la même manière que les opérations d'objets ϕ_2 et ϕ_3 que nous avons décrites en Section 1.1. Par contre, les opérations d'objets θ_3 et ψ_1 sont nouvelles. θ_3 prend en argument un polyomino marqué et colle une cellule unitaire à la cellule marquée de la dernière colonne. Quant à ψ_1, elle prend en argument un polyomino et marque la cellule du bas de la dernière colonne.

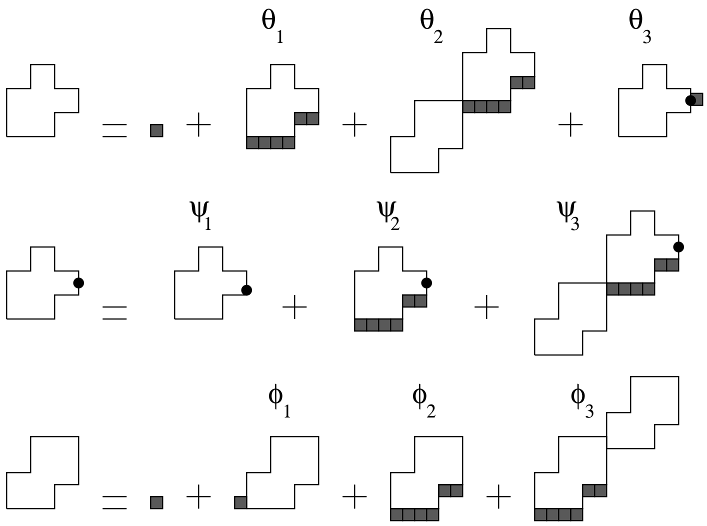


Figure 18: Une grammaire non ambiguë pour les polyominos convexes dirigés.

Proposition 1.23 *La grammaire d'objets décrite Figure 18 est non ambiguë.*

PREUVE :
Si P est un polyomino convexe dirigé, nous avons les cas disjoints suivants :

- soit P est la cellule de base,
- soit on peut enlever à chaque colonne de P une cellule en bas, tout en gardant un ensemble de cellules connexe : $P \in Im(\theta_1)$,
- sinon,
  - soit il y a une seule cellule dans la dernière colonne de P, on l'enlève en marquant la cellule à laquelle elle est adjacente, on obtient ainsi un polyomino convexe dirigé marqué : $P \in Im(\theta_3)$,
  - soit, en allant de droite à gauche, on peut enlever une cellule en bas des colonnes de P jusqu'à ce que deux colonnes ne soient adjacentes que par une seule cellule, P se décompose alors en un polyomino parallélogramme et un polyomino convexe dirigé : $P \in Im(\theta_2)$.

Ainsi, $\mathcal{E}_{pcd}$ est la réunion disjointe $\{\square\} \bigcup Im(\theta_1) \bigcup Im(\theta_2) \bigcup Im(\theta_3)$. Or les opérations d'objets θ_1, θ_2, θ_3 sont injectives : la première équation de la Figure 18 décrit cette décomposition non ambiguë.
Si M est un polyomino convexe dirigé marqué :

- soit la marque sur la dernière colonne de M se trouve sur la cellule du bas : $M \in Im(\psi_1)$,
- sinon,
  - soit on peut enlever à chaque colonne de M une cellule en bas : $M \in Im(\psi_2)$,
  - soit, en allant de droite à gauche, on peut enlever une cellule en bas des colonnes de M jusqu'à ce que deux colonnes ne soient adjacentes que par une seule cellule, M se décompose alors en un polyomino parallélogramme et un polyomino convexe dirigé marqué : $M \in Im(\psi_3)$.

Ainsi, $\mathcal{E}_{pcdm}$ est la réunion disjointe $Im(\psi_1) \bigcup Im(\psi_2) \bigcup Im(\psi_3)$. Or les opérations d'objets ψ_1, ψ_2, ψ_3 sont injectives : la deuxième équation de la Figure 18 décrit cette décomposition non ambiguë.

Nous avons déjà montré dans l'exemple de la Section 1.5 que la troisième équation est une grammaire d'objets non ambiguë pour les polyominos parallélogrammes. Ainsi, la grammaire d'objets de la Figure 18 est non ambiguë.

□

Chapitre 2

Enumération

Ce chapitre est consacré à l'application des grammaires d'objets à des problèmes d'énumération d'objets combinatoires. Le but est de répondre à la question : “quel est le nombre d'objets d'un type donné ayant un certain paramètre égal à la valeur k ?”. L'outil le plus adapté pour réaliser ceci est sans nul doute la notion de *fonction génératrice*. (Voir par exemple l'ouvrage de Graham, Knuth et Patashnik [63], et celui de Wilf [96])

Rappelons tout d'abord la notion de *série formelle*, qui généralise celle de polynôme. Si K est un corps, et $X = \{x_1, \ldots, x_n\}$ un ensemble de variables, alors une *série formelle* f sur K à variables dans X est une expression de la forme

$$f = f(x) = f(x_1, \ldots, x_n) = \sum_{k_1, \ldots, k_n \geq 0} a_{k_1, \ldots, k_n} {x_1}^{k_1} \ldots {x_n}^{k_n}$$

où les $a_{k_1, \ldots, k_n}$, appelés *coefficients* de f, constituent une suite multiple d'ordre n à valeurs dans K. On note souvent $a_{k_1, \ldots, k_n} = < {x_1}^{k_1} \ldots {x_n}^{k_n}, f >$. L'ensemble de ces séries formelles est noté $K[[x_1, \ldots, x_n]]$ ou $K[[X]]$. Si les $a_{k_1, \ldots, k_n}$ sont “presque tous nuls” (nuls sauf un nombre fini), alors f est un polynôme. On note $K[x_1, \ldots, x_n]$, ou $K[X]$, l'ensemble des polynômes à variables dans X et à coefficients dans K.

Nous pouvons maintenant rappeler la notion de *fonction* (ou *série*) *génératrice*. Etant donnée une suite de nombres a_k (réels ou complexes, mais en fait ici entiers ≥ 0), on appelle *fonction génératrice ordinaire* la série formelle :

$$\sum_{k \geq 0} a_k x^k \ .$$

La notion de fonction génératrice ordinaire se généralise immédiatement aux suites multiples $a_{k_1, \ldots, k_n}$:

$$\sum_{k_1, \ldots, k_n \geq 0} a_{k_1, \ldots, k_n} {x_1}^{k_1} \ldots {x_n}^{k_n} \ .$$

La première section définit la notion de *valuation d'objets*, et établit le théorème principal du chapitre; il consiste à déduire directement d'une grammaire d'objets *complète et non ambiguë*, un système d'équations vérifiées par les fonctions génératrices des différents ensembles d'objets, selon des valuations d'objets données. Dans le cas où les valuations d'objets sont *linéaires* (Section 2.2), nous obtenons des systèmes d'équations algébriques, qui peuvent être habituellement résolus. Lorsque les valuations d'objets sont *q-linéaires* (Section 2.3), les systèmes d'équations obtenus sont des systèmes de q-équations, pour lesquels il n'existe pas de méthode générale de résolution. Cependant, quelques cas particuliers sont traités dans la littérature, par exemple dans [19, 22, 37, 49, 53, 79]. La quatrième section donne une légère extension d'une méthode de résolution de Prellberg et Brak [79], qui est utile dans la dernière section pour traiter quelques exemples d'énumération.

2.1 Théorème d'énumération

Soit $X = \{x_1, \ldots, x_n\}$ un ensemble de variables. Nous considérons l'ensemble des polynômes $K[X]$ et l'ensemble des séries formelles $K[[X]]$.

Le souci de manipuler des séries formelles nous amène à établir la définition suivante :

Définition 2.1 *Soit E un ensemble d'objets. Une application $V_E : E \longrightarrow K[X]$ est une valuation d'objets sur E si elle vérifie la propriété suivante :*

$$\forall\,(k_1, \ldots, k_n) \in \mathbb{N}^n,\ \{e \in E / < {x_1}^{k_1} \ldots {x_n}^{k_n}\,,\, V_E(e) > \neq 0\} \text{ est un ensemble fini.}$$

Ainsi, la fonction génératrice associée à E, $\sum_{e\in E} V_E(e)$ est une série formelle de $K[[X]]$. On notera $V_E(E) = \sum_{e\in E} V_E(e)$.

Exemple 2.1 : Sur l'ensemble des polyominos parallélogrammes :

L'application $val-périmètre$ est une valuation d'objets sur E_{pp}.

$$\begin{array}{rccl} val-périmètre : & E_{pp} & \longrightarrow & \mathbb{N}[t] \\ & e & \longmapsto & t^{périmètre(e)} \end{array}$$

En effet, il y a un nombre fini de polyominos parallélogrammes d'un certain périmètre donné.

Par contre, l'application $val-colonnes$ n'est pas une valuation d'objets sur E_{pp}.

$$\begin{array}{rccl} val-colonnes : & E_{pp} & \longrightarrow & \mathbb{N}[x] \\ & e & \longmapsto & x^{nbre-colonnes(e)} \end{array}$$

En effet, le nombre de polyominos parallélogrammes ayant un nombre de colonnes fixé est infini (le nombre de cellules dans chaque colonne n'est pas borné).

Définition 2.2 *Soit $G = < \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets. Pour tout E dans $\mathcal{E}$, soit V_E une valuation d'objets sur E. On dit que $\nu = \{V_E : E \longrightarrow K[X], E \in \mathcal{E}\}$ est un ensemble de valuations d'objets associé à G.*

Théorème 2.3 *Soit $G = < \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets non ambiguë et complète. Soit ν un ensemble de valuations d'objets associé à G : $\nu = \{V_E : E \longrightarrow K[X], E \in \mathcal{E}\}$.*
Pour tout E dans $\mathcal{E}$, à partir de son équation dans G qui est de la forme

$$E \;=\; \sum_{e \in T_E} e \;+\; \sum_{dest(\phi)=E} \phi(E_{1_\phi}, \ldots, E_{k_\phi}) \;, \tag{1}$$

on peut déduire directement l'équation :

$$V_E(E) \;=\; V_E(T_E) \;+\; \sum_{dest(\phi)=E} V_E(\phi(E_{1_\phi}, \ldots, E_{k_\phi})) \;. \tag{2}$$

PREUVE :
La grammaire d'objets G est non ambiguë et complète, donc l'équation (1) se traduit en une égalité ensembliste,

$$E = T_E \bigcup \left(\bigcup_{dest(\phi)=E} \underbrace{\phi(E_{1_\phi}, \ldots, E_{k_\phi})}_{Im(\phi)} \right) .$$

Le fait que ces unions sont disjointes permet de déduire directement l'équation (2).

$\square$

Exemple 2.2 : La grammaire G_2 engendre les polyominos parallélogrammes de façon non ambiguë,

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

En appliquant à E_{pp} la valuation *val-périmètre*, que nous notons de façon plus condensée V_p, nous obtenons :

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

Les sections suivantes 2.2 et 2.3 présentent deux types de valuations qui permettent d'obtenir un système d'équations portant sur les fonctions génératrices des différents ensembles.

2.2 Valuations d'objets linéaires

Les valuations d'objets linéaires sont exactement le cadre d'application de la méthodologie DSV classique, elles mènent à des séries génératrices algébriques.

Définition 2.4 *Soient $E, E_1, \ldots, E_k$ des ensembles d'objets, et $V_E, V_{E_1}, \ldots, V_{E_k}$ des valuations d'objets respectivement sur $E, E_1, \ldots, E_k$. Soit ϕ une opération d'objets avec $dest(\phi) = E$ et $dom(\phi) = E_1 \times \ldots \times E_k$. On dit que V_E est linéaire par rapport à ϕ s'il existe un polynôme p_ϕ de $K[X]$ tel que*

$$V_E(\phi(e_1, \ldots, e_k)) = p_\phi V_{E_1}(e_1) \ldots V_{E_k}(e_k) \text{ , pour tout } (e_1, \ldots, e_k) \in dom(\phi).$$

Si ϕ est une opération d'objets *injective*, on a :

$$V_E(\phi(E_1, \ldots, E_k)) = p_\phi V_{E_1}(E_1) \ldots V_{E_k}(E_k) \ .$$

Exemple 2.3 : La valuation d'objets *val-périmètre* (ou V_p) définie sur les polyominos parallélogrammes est linéaire par rapport à l'opération d'objets ϕ_1. En effet, ϕ_1 consiste à coller une cellule en bas à gauche d'un polyomino, ce qui augmente de deux le périmètre du polyomino initial. Ainsi, pour tout e de E_{pp}, $périmètre(\phi_1(e)) = périmètre(e) + 2$, et $V_p(\phi_1(e)) = t^2 V_p(e)$, et comme ϕ_1 est injective, $V_p(\phi_1(E_{pp})) = t^2 V_p(E_{pp})$.

Définition 2.5 *Soit $\nu = \{V_E : E \longrightarrow K[X], E \in \mathcal{E}\}$ un ensemble de valuations d'objets associé à une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$. On dit que ν est G-linéaire si pour tout E de $\mathcal{E}$, V_E est linéaire par rapport à toutes les opérations d'objets ϕ de $\mathcal{P}$ telles que $dest(\phi) = E$.*

Nous pouvons alors énoncer un corollaire du Théorème 2.3.

Corollaire 2.6 *Si l'ensemble des valuations d'objets ν est G-linéaire, alors l'équation (2) du Théorème 2.3 devient :*

$$V_E(E) \quad = \quad V_E(T_E) \quad + \sum_{dest(\phi)=E} p_\phi \, V_{E_{1_\phi}}(E_{1_\phi}) \ldots V_{E_{k_\phi}}(E_{k_\phi}) \ . \tag{3}$$

Nous obtenons ainsi un système d'équations algébriques ayant pour inconnues les fonctions génératrices des ensembles E de $\mathcal{E}$, $V_E(E)$. Une des solutions du système est la fonction génératrice des objets engendrés par G : $V_S(S)$ si S est l'axiome.

Exemple 2.4 : Nous avons montré précédemment que la valuation d'objets *val-périmètre* (V_p) sur les polyominos parallélogrammes est linéaire par rapport à ϕ_1 : $V_p(\phi_1(E_{pp})) = t^2 V_p(E_{pp})$. De même, nous pouvons montrer que V_p est linéaire par rapport à ϕ_2 et ϕ_3 : $V_p(\phi_2(E_{pp})) = t^2 V_p(E_{pp})$ et $V_p(\phi_3(E_{pp}, E_{pp})) = V_p(E_{pp})^2$. Ainsi, de l'équation

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

nous déduisons l'équation sur la série génératrice

$$V_p(E_{pp}) = t^4 + t^2 V_p(E_{pp}) + t^2 V_p(E_{pp}) + V_p(E_{pp})^2 \ .$$

La résolution de cette équation fournit la série génératrice pour les polyominos parallélogrammes selon le périmètre :

$$V_p(E_{pp}) = \frac{1 - 2t^2 - \sqrt{1-4t^2}}{2} = \sum_{n\geq 1} \frac{1}{n+1}\binom{2n}{n} t^{2(n+1)} ,$$

qui est la série génératrice des nombres de Catalan.

2.3 Valuations d'objets q-linéaires

Nous considérons ici deux ensembles disjoints de variables $X = \{x_1, \ldots, x_n\}$ et $Q = \{q_1, \ldots, q_r\}$, ainsi que l'ensemble des polynômes $K[X, Q]$ et l'ensemble des séries formelles $K[[X, Q]]$.

Notations : Nous nous donnons les notations suivantes :

- $x = (x_1, \ldots, x_n)$ et $q = (q_1, \ldots, q_r)$,
- pour une matrice $n \times r$ à coefficients dans $\mathbb{N}$, $A = (a_{ij})$, $1 \leq i \leq n$, $1 \leq j \leq r$, nous notons
  - $xq^A = (x_1 {q_1}^{a_{11}} \ldots {q_r}^{a_{1r}}, \ldots, x_n {q_1}^{a_{n1}} \ldots {q_r}^{a_{nr}})$,
  - pour $f(x, q) \in K[[X, Q]]$, $f(x, q)|_{x \leftarrow xq^A} = f(xq^A, q)$.

Définition 2.7 *Soient $E, E_1, \ldots, E_k$ des ensembles d'objets, et $V_E, V_{E_1}, \ldots, V_{E_k}$ des valuations d'objets respectivement sur $E, E_1, \ldots, E_k$. Soit ϕ une opération d'objets avec $dest(\phi) = E$ et $dom(\phi) = E_1 \times \ldots \times E_k$. On dit que V_E est q-linéaire par rapport à ϕ s'il existe un polynôme λ_ϕ de $K[X, Q]$ et des matrices ${A_i}^\phi$ pour tout $i \in [1, k]$ tels que*

$$V_E(\phi(e_1, \ldots, e_k)) = \lambda_\phi \prod_{i=1}^{k} V_{E_i}(e_i)|_{x \leftarrow xq^{{A_i}^\phi}} \text{ , pour tout } (e_1, \ldots, e_k) \in dom(\phi).$$

Si ϕ est une opération d'objets *injective*, nous avons alors :

$$V_E(\phi(E_1, \ldots, E_k)) = \lambda_\phi \prod_{i=1}^{k} V_{E_i}(E_i)|_{x \leftarrow xq^{{A_i}^\phi}}.$$

Remarque : Les valuations linéaires coïncident avec les valuations q-linéaires telles que pour tout ϕ et tout i, ${A_i}^\phi = (0)$.

Exemple 2.5 : Sur l'ensemble des polyominos parallélogrammes :

$$\begin{array}{rccl} V_{ca} : & E_{pp} & \longrightarrow & \mathbb{N}[x, q] \\ & e & \longmapsto & x^{nbre-colonnes(e)} q^{aire(e)} \end{array}$$

où $nbre - colonnes(e)$ retourne le nombre de colonnes de e et $aire(e)$ son nombre de cellules. Cette valuation d'objets est q-linéaire par rapport à l'opération d'objets ϕ_2. En effet, ϕ_2 consiste à coller une cellule en bas de chaque colonne d'un polyomino, ce qui ajoute autant de cellules que le polyomino a de colonnes. Ainsi, pour tout e de E_{pp}, $V_{ca}(\phi_2(e)) = V_{ca}(e)|_{x \leftarrow xq}$ et $V_{ca}(\phi_2(E_{pp})) = V_{ca}(E_{pp})|_{x \leftarrow xq}$.

Définition 2.8 *Soit $\nu = \{V_E : E \longrightarrow K[X,Q], E \in \mathcal{E}\}$ un ensemble de valuations d'objets associé à une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$. On dit que ν est G-q-linéaire lorsque toutes les valuations d'objets de ν sont q-linéaires.*

Nous pouvons alors énoncer un corollaire du Théorème 2.3.

Corollaire 2.9 *Si l'ensemble des valuations d'objets ν est G-q-linéaire, alors l'équation (2) du Théorème 2.3 devient :*

$$V_E(E) \quad = \quad V_E(T_E) \; + \sum_{dest(\phi)=E} \lambda_\phi \prod_{i=1_\phi}^{k_\phi} V_{E_i}(E_i)|_{x \leftarrow xq^{A_i \phi}} \; . \tag{4}$$

Nous obtenons ainsi un système de q-équations ayant pour inconnues les fonctions génératrices des ensembles E de $\mathcal{E}$, $V_E(E)$.

Exemple 2.6 : Nous avons montré précédemment que la valuation d'objets V_{ca} sur les polyominos parallélogrammes est q-linéaire par rapport à ϕ_2 : $V_{ca}(\phi_2(E_{pp})) = V_{ca}(E_{pp})|_{x \leftarrow xq}$. De même, nous pouvons montrer que V_{ca} est linéaire (donc q-linéaire) par rapport à ϕ_1 et q-linéaire par rapport à ϕ_3 : $V_{ca}(\phi_1(E_{pp})) = xqV_{ca}(E_{pp})$ et $V_{ca}(\phi_3(E_{pp}, E_{pp})) = V_{ca}(E_{pp})|_{x \leftarrow xq} V_{ca}(E_{pp})$. Ainsi, de l'équation

$$V_{ca}(E_{pp}) = V_{ca}(\square) + V_{ca}(\phi_1(E_{pp})) + V_{ca}(\phi_2(E_{pp})) + V_{ca}(\phi_3(E_{pp}, E_{pp})) \; ,$$

nous déduisons la q-équation sur la série génératrice (en notant $f_{ca}(x,q) = V_{ca}(E_{pp})$)

$$f_{ca}(x,q) = xq + xqf_{ca}(x,q) + f_{ca}(xq,q) + f_{ca}(xq,q)f_{ca}(x,q) \; .$$

Dans [40], Delest et Fédou ont obtenu cette même q-équation, et ils ont montré que la fonction génératrice des polyominos parallélogrammes selon la largeur (x) et l'aire (q) est rattachée au rapport de deux q-analogues de fonctions de Bessel J_0 et J_1.

$$f_{ca}(x,q) = \frac{J_1(x,q)}{J_0(x,q)} \; ,$$

avec

$$J_0(x,q) = \sum_{n \geq 0} x^n \frac{(-1)^n q^{\binom{n+1}{2}}}{{(q)_n}^2} \; , \qquad \text{et} \qquad J_1(x,q) = \sum_{n \geq 1} x^n \frac{(-1)^{n-1} q^{\binom{n+1}{2}}}{(q)_n (q)_{n-1}} \; .$$

Remarque : Lorsqu'une grammaire d'objets complète non ambiguë engendre un ensemble d'objets, celle-ci peut servir à étudier différentes séries génératrices des objets selon divers paramètres, en appliquant successivement les valuations d'objets correspondantes. Par exemple, avec la même grammaire d'objets pour les polyominos parallélogrammes, nous avons calculé leur fonction génératrice selon le périmètre, puis celle selon la largeur et l'aire.

2.4 La résolution de q-équations

Il n'existe aucune méthode générale de résolution pour les systèmes de q-équations quelconques, mais quelques cas particuliers peuvent être traités.

Une forme d'heuristique de résolution est développée par Bousquet-Mélou et Fédou dans [22] ; son principe consiste à ramener la résolution d'un système de q-équations à celle d'un système de q-équations différentielles. C'est une sorte de q-analogue de la méthode classique utilisée pour résoudre des systèmes linéaires d'équations différentielles ordinaires.

Dans [19], Bousquet-Mélou donne une formule générale pour la résolution d'un certain type de q-équation qui intervient en utilisant une décomposition en colonnes des polyominos (méthode de Temperley). Une méthode similaire est développée par Feretić et Svrtan dans [53].

Dans [79], Prellberg et Brak étudient différents modèles de polygones (polyominos) dont les fonctions génératrices vérifient toutes des q-équations de la même forme. Ils donnent alors une technique générale pour linéariser ce type de q-équation, puis fournissent une solution lorsque cette équation linéaire a une forme particulière.

Nous présentons ici une légère extension de la méthode ci-dessus en allégeant les restrictions portant sur l'équation linéaire obtenue. Le lemme que nous établissons est utilisé pour la plupart des exemples d'énumérations traités dans la section suivante. Rappelons tout d'abord le résultat de Prellberg et Brak [79].

Considérons la q-équation de la forme

$$G(x)G(xq) + a(x)G(x) + b(x)G(xq) + c(x) \quad = \quad 0 \ , \tag{5}$$

où $G(x) = G(x, q)$ désigne une série formelle de $K[[x, q]]$, et $a(x)$, $b(x)$ et $c(x)$ des polynômes de $K[x, q]$. Elle peut être linéarisée en utilisant la transformation

$$G(x) \quad = \quad \alpha \frac{H(xq)}{H(x)} - b(x) \ , \tag{6}$$

où $H(x) \in K[[x, q]]$, et α doit être choisi de sorte que les conditions initiales soient vérifiées. On obtient ainsi une équation fonctionnelle linéaire en $H(x)$,

$$\alpha^2 H(xq^2) + \alpha \left[a(x) - b(xq)\right] H(xq) + \left[c(x) - a(x)b(x)\right] H(x) \quad = \quad 0 \ . \tag{7}$$

Prellberg et Brak se restreignent alors à une équation fonctionnelle linéaire de la forme

$$0 = xH(xq) + \sum_{k=0}^{N} \alpha_k H(xq^k) \qquad \text{avec} \qquad \sum_{k=0}^{N} \alpha_k = 0 \ ,$$

et tous les α_k indépendants de x. La solution est donnée par

$$H(x) = \sum_{k=0}^{\infty} \frac{(-x)^n q^{\binom{n}{2}}}{\prod_{m=1}^{n} \Lambda(q^m)} \qquad \text{avec} \quad \Lambda(t) = \sum_{k=0}^{N} \alpha_k t^k \ .$$

Elle est obtenue simplement en résolvant la récurrence sur les coefficients des puissances de x dans $H(x)$.

Nous avons alors remarqué que la solution reste explicitement calculable avec des coefficients α_k de degré ≤ 1 en x.

Lemme 2.10 *Soit une équation fonctionnelle linéaire de la forme*

$$0 = \sum_{k=0}^{N} \alpha_k H(xq^k) + x \sum_{k=0}^{M} \beta_k H(xq^k) \qquad \textit{avec} \qquad \sum_{k=0}^{N} \alpha_k = 0 \ ,$$

et tous les α_k et β_k indépendants de x. Alors, la solution est donnée par

$$H(x) = \sum_{k=0}^{\infty} (-x)^n \prod_{m=1}^{n} \frac{\Lambda_\beta(q^{m-1})}{\Lambda_\alpha(q^m)} \qquad \textit{avec} \quad \Lambda_\alpha(t) = \sum_{k=0}^{N} \alpha_k t^k \quad \textit{et} \quad \Lambda_\beta(t) = \sum_{k=0}^{M} \beta_k t^k \ .$$

PREUVE :

Cette solution est de nouveau obtenue en résolvant la récurrence sur les coefficients des puissances de x dans $H(x)$. Si h_n est le coefficient de x^n dans $H(x)$, alors on obtient la récurrence

$$0 = h_n \sum_{k=0}^{N} \alpha_k q^{kn} + h_{n-1} \sum_{k=0}^{M} \beta_k q^{k(n-1)} \ ,$$

soit

$$h_n = -\frac{\sum_{k=0}^{M} \beta_k q^{k(n-1)}}{\sum_{k=0}^{N} \alpha_k q^{kn}} h_{n-1} \ .$$

Par itération, et en prenant $h_0 = 1$, on trouve le résultat.

□

Remarque : Beaucoup de fonctions génératrices d'objets combinatoires vérifient des systèmes de q-équations que l'on ne sait pas résoudre directement et explicitement. Mais ces q-équations permettent tout de même de déduire des récurrences sur les coefficients des séries génératrices. Nous verrons au Chapitre 4 que celles-ci sont très précieuses dans le domaine de la génération aléatoire uniforme d'objets combinatoires.

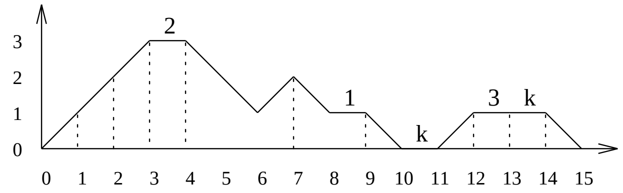


Figure 19: Un chemin de Motzkin k-coloré.

2.5 Quelques exemples d'énumération

2.5.1 Les chemins de Motzkin k-colorés

Un chemin de Motzkin a son origine et son extrémité sur l'axe des abscisses, et est formé de pas Nord-Est, Sud-Est et Est se trouvant tous au-dessus de cet axe. Un chemin de Motzkin k-coloré est un chemin de Motzkin dont les pas horizontaux (Est) sont colorés à l'aide de k couleurs. Un exemple d'un tel chemin est donné Figure 19 (pour des raisons d'impression en noir et blanc, les couleurs sont numérotées de 1 à k, et les pas horizontaux de couleur i sont étiquetés par i). La Figure 20 décrit une grammaire d'objets non ambiguë pour les chemins de Motzkin k-colorés.

Nous nous intéressons à l'énumération de ces objets selon l'aire relative aux pas montants (Nord-Est) et horizontaux, définie comme la somme des hauteurs des extrémités de ces pas par rapport à l'axe des abscisses. Le chemin de la Figure 19 est d'aire 15 et de longueur 15 (nombre total de pas).

On note C_k l'ensemble des chemins de Motzkin k-colorés. Soit $c \in C_k$, on note :

- $n(c)$ la longueur de c,
- $m(c)$ le nombre de pas montants et horizontaux de c,
- $a(c)$ l'aire de c.

La fonction génératrice des éléments de C_k selon ces trois paramètres est

$$M_k(x, z, q) = \sum_{c \in C_k} x^{m(c)} z^{n(c)} q^{a(c)} .$$

Figure 20: Une grammaire d'objets non ambiguë pour les chemins de Motzkin k-colorés.

Notre Théorème d'énumération 2.3 nous permet de déduire, de la grammaire d'objets Figure 20, la q-équation suivante :

$$M_k(x) = 1 + kxzM_k(x) + xz^2 qM_k(xq)M_k(x) .$$

Une simple application du Lemme 2.10 établi à la section précédente fournit la solution cette q-équation.

Proposition 2.11 *La fonction génératrice $M_k(x,z,q)$ des chemins de Motzkin k-colorés est donnée par :*

$$M_k(x) = \frac{T_1^k(x)}{T_0^k(x)} ,$$

avec

$$T_0^k(x) = \sum_{n\geq 0} x^n \frac{(-z)^n q^{\binom{n}{2}} \prod_{i=1}^n (k+zq^i)}{(q)_n} \qquad et \qquad T_1^k(x) = \sum_{n\geq 0} x^n \frac{(-z)^n q^{\binom{n+1}{2}} \prod_{i=1}^n (k+zq^i)}{(q)_n} .$$

Remarque : Barcucci, Del Lungo, Pergola et Pinzani calcule dans [7] la fonction génératrice des chemins de Motzkin k-colorés $S_1(x,y,q)$ selon la longueur (x), l'aire (q), et la longueur de la dernière descente (y). Une méthode de construction par opérateur leur permet d'obtenir une q-équation qu'ils résolvent grâce au lemme de Bousquet-Mélou [19]. Les deux techniques aboutissent à la même forme pour la série génératrice : nous avons en fait $M_k(1,z,q) = S_1(z,1,q)$. Cependant, le paramètre “nombre de pas montants et horizontaux”, exposant de la variable x dans notre formule, permet d'obtenir une forme totalement développée en x pour les séries formelles T_0^k et T_1^k.

Nous déduisons directement de la proposition précédente une formule d'énumération pour les chemins de Dyck selon les mêmes paramètres; en effet, les chemins de Dyck sont tout simplement des chemins de Motzkin k-colorés pour $k = 0$. Leur fonction génératrice est donc donnée par :

$$D(x) = M_0(x) = \frac{T_1^0(x)}{T_0^0(x)} ,$$

avec

$$T_0^0(x) = \sum_{n\geq 0} x^n \frac{(-z^2)^n q^{n^2}}{(q)_n} \qquad \text{et} \qquad T_1^0(x) = \sum_{n\geq 0} x^n \frac{(-z^2)^n q^{n(n+1)}}{(q)_n} .$$

Au lieu de l'aire relative aux pas montants et horizontaux, considérons maintenant l'aire standard, définie comme la somme des hauteurs des extrémités de tous les pas. Soit $M'_k(x,z,q)$ la fonction génératrice des chemins de Motzkin k-colorés,

$$M'_k(x,z,q) = \sum_{c\in C_k} x^{m(c)} z^{n(c)} q^{a'(c)} ,$$

où $a'(c)$ est l'aire standard du chemin c. La q-équation que nous obtenons n'est plus q-linéaire par rapport à la variable x, mais par rapport à z :

$$M'_k(z) = 1 + kxzM'_k(z) + xz^2qM'_k(zq)M'_k(z) ,$$

et nous pouvons la linéariser, mais pas la résoudre au moyen du Lemme 2.10. Le problème vient de la présence de z^2 dans un des coefficients qui induit une récurrence à deux termes au lieu d'un.

Nous pouvons tout de même déduire, directement de cette q-équation, un développement en fraction continue satisfait par $M'_k(x,z,q)$, résultat de Flajolet [54] pour $k = 0$ ou 1 et $x = 1$:

Proposition 2.12 *La fonction génératrice $M'_k(x,z,q)$ des chemins de Motzkin k-colorés satisfait :*

$$M'_k(x,z,q) = \cfrac{1}{1 - kxz - \cfrac{xz^2q}{1 - kxzq - \cfrac{xz^2q^3}{1 - kxzq^2 - \cfrac{xz^2q^5}{\cdots}}}} .$$

2.5.2 Les arbres planaires

La Figure 21 décrit une grammaire d'objets non ambiguë pour les arbres planaires.

Nous nous intéressons à l'énumération de ces objets selon la longueur du chemin interne, définie comme la somme des distances (en nombre d'arêtes) à la racine de tous les sommets. Nous considérons que la racine est à distance 0 d'elle-même. L'arbre de la Figure 22 a 11 sommets et la longueur de son chemin interne est 22.

On note A l'ensemble des arbres planaires. Soit $a \in A$, on note :

- $n(a)$ le nombre de sommets de a,
- $c(a)$ la longueur du chemin interne de a.

La fonction génératrice des éléments de A selon ces deux paramètres est

$$P(x,q) = \sum_{a \in A} x^{n(a)} q^{c(a)} .$$

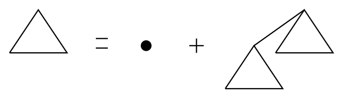


Figure 21: Une grammaire d'objets non ambiguë pour les arbres planaires.

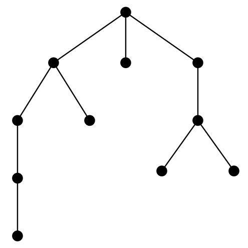


Figure 22: Un arbre planaire.

Notre Théorème d'énumération 2.3 nous permet de déduire, de la grammaire d'objets Figure 21, la q-équation suivante :

$$P(x) = x + P(xq)P(x) \ .$$

Si nous appliquons directement le Lemme 2.10, nous obtenons une formule de la forme :

$$P(x) = \frac{F_1(x)}{F_0(x)} \ , \qquad \text{avec} \qquad F_0(x) = \sum_{n\geq 0} \frac{x^n}{q^{\binom{n+1}{2}}(q)_n} \qquad \text{et} \qquad F_1(x) = \sum_{n\geq 0} \frac{x^n}{q^{\binom{n}{2}}(q)_n} \ ,$$

mais ce résultat n'est pas satisfaisant, car les fonctions F_0 et F_1 ne sont pas des séries formelles (nous ne pouvons pas développer par rapport à q les coefficients de x^n en une série entière). Nous nous sortons de ce mauvais pas en effectuant le changement de variable $P(x) = xY(x)$, et en appliquant le lemme à la q-équation sur $Y(x)$ ainsi obtenue,

$$Y(x) = 1 + xqY(xq)Y(x) \ .$$

Proposition 2.13 *La fonction génératrice $P(x, q)$ des arbres planaires est donnée par :*

$$P(x) = xY(x) = \frac{F_1(x)}{F_0(x)} \ ,$$

avec

$$F_0(x) = \sum_{n\geq 0} x^n \frac{(-1)^n q^{n^2}}{(q)_n} \qquad \textit{et} \qquad F_1(x) = \sum_{n\geq 1} x^n \frac{(-1)^{n-1} q^{n(n-1)}}{(q)_{n-1}} \ .$$

Remarque : Avec la même méthode que précédemment (voir remarque page 46), Barcucci, Del Lungo, Pergola et Pinzani [8] ont calculé la fonction génératrice des arbres planaires $A(s, x, y, q)$ selon la longueur de la branche la plus à droite (s), le nombre de sommets (x), le nombre de feuilles (y), et la longueur du chemin interne augmentée du nombre de sommets (q); en effet, ils considèrent la racine à distance 1 d'elle-même, et non pas 0. Si nous substituons s et y par 1 dans leur formule, nous obtenons :

$$A(1, x, 1, q) = \frac{H_1(x)}{H_0(x)} \ ,$$

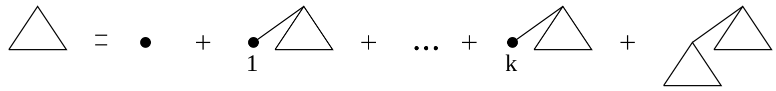


Figure 23: Une grammaire d'objets non ambiguë pour les arbres planaires k-colorés.

avec

$$H_0(x) = \sum_{n\geq 0} x^n \frac{(-1)^n q^{n(n+1)}}{(q)_n} \qquad \text{et} \qquad H_1(x) = \sum_{n\geq 1} x^n \frac{(-1)^{n-1} q^{n^2}}{(q)_{n-1}} \ .$$

Elle a la même forme que notre formule où l'on substitue x par xq pour tenir compte de la différence de définition pour la longueur du chemin interne, donc $A(1,x,1,q) = P(xq,q)$.

Définissons les arbres planaires k-colorés comme des arbres planaires dont les sommets internes sont non colorés, et dont les feuilles sont colorées, ou pas, à l'aide de k couleurs. Une exception est faite pour l'arbre à un seul sommet qui ne peut pas être coloré. La Figure 23 décrit une grammaire d'objets non ambiguë pour les arbres planaires k-colorés.

En notant $P_k(x,q)$ la fonction génératrice des arbres planaires k-colorés, nous obtenons à partir de la Figure 23 :

$$P_k(x) = x + kxqP_k(x) + P_k(xq)P_k(x) \ .$$

Pour les mêmes raisons que précédemment, nous effectuons le changement de variable $P_k = xY_k$ avant d'appliquer le lemme,

$$Y_k(x) = 1 + kxqY_k(x) + xqY_k(xq)Y_k(x) \ .$$

Proposition 2.14 *La fonction génératrice $P_k(x,q)$ des arbres planaires k-colorés est donnée par :*

$$P_k(x) = xY_k(x) = \frac{F_1^k(x)}{F_0^k(x)} \ ,$$

avec

$$F_0^k(x) = \sum_{n\geq 0} x^n \frac{(-1)^n q^{\binom{n+1}{2}} \prod_{i=0}^{n-1}(k+q^i)}{(q)_n}$$

et

$$F_1^k(x) = \sum_{n\geq 1} x^n \frac{(-1)^{n-1} q^{\frac{(n-1)(n+2)}{2}} \prod_{i=0}^{n-2}(k+q^i)}{(q)_{n-1}} \ .$$

Remarque : Si nous considérons que la distance à la racine d'une feuille colorée est diminuée de 1, alors nous définissons une longueur de chemin interne spéciale qui, pour un arbre a est $c'(a) =$

$c(a)-$nombre de feuilles colorées. Nous notons $P'_k(x,q)$ la fonction génératrice correspondante, et toujours à partir de la grammaire d'objets Figure 23, nous déduisons

$$P'_k(x) = x + kxP'_k(x) + P'_k(xq)P'_k(x) \ ,$$

et par le changement de variable $P'_k = xY'_k$,

$$Y'_k(x) = 1 + kxY'_k(x) + xqY'_k(xq)Y'_k(x) \ .$$

Nous remarquons alors que $Y'_k(x,q)$ et $M_k(x,1,q)$ vérifient la même q-équation, et donc

$$P'_k(x,q) = xM_k(x,1,q) \ .$$

Ainsi, il y a une bijection entre les chemins de Motzkin k-colorés ayant n pas montants et horizontaux et d'aire r et les arbres planaires k-colorés ayant $n+1$ sommets et de longueur du chemin interne spéciale r.

Chapitre 3

Bijections

Nous présentons dans ce chapitre des méthodes de construction de bijections entre ensembles d'objets, en utilisant des grammaires d'objets complètes, et surtout non ambiguës.

Prenons l'exemple de la bijection très classique entre les mots (ou chemins) de Dyck de longueur $2n$ et les arbres binaires complets ayant $2n+1$ sommets (n sommets internes et $n+1$ feuilles). Elle se décrit habituellement de la façon suivante : on parcourt les sommets de l'arbre dans l'ordre préfixe et on code les nœuds internes par x (ou un pas montant) et les feuilles, sauf la dernière, par $\overline{x}$ (ou un pas descendant). Cette bijection s'exprime très simplement à l'aide des deux grammaires d'objets non ambiguës Figure 24. Ces grammaires d'objets ont la même "structure" : un ensemble d'objets avec un objet terminal et une opération d'objets d'arité 2. Ainsi, nous pouvons établir une bijection entre les arbres de dérivation des deux grammaires en échangeant les opérations d'objets et les objets terminaux (voir Figure 25). Comme les grammaires d'objets sont non ambiguës, nous obtenons alors une bijection entre les objets. Des grammaires d'objets qui ont la même "structure" sont appelées *grammaires d'objets isomorphes*; c'est la notion fondamentale de ce chapitre.

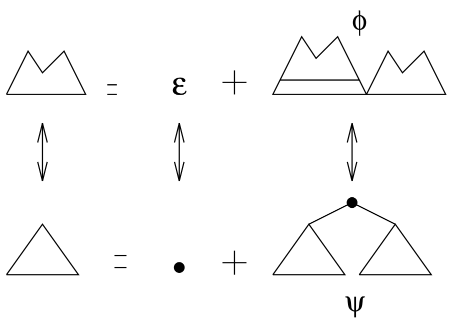


Figure 24: Grammaires d'objets pour les chemins de Dyck et les arbres binaires complets.

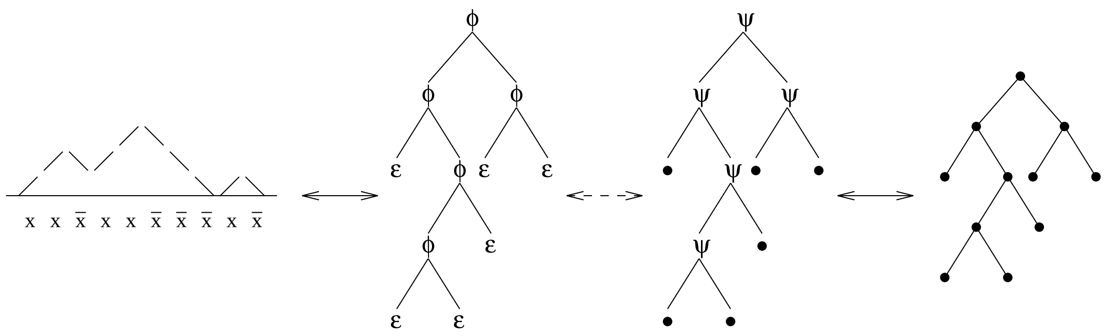


Figure 25: Bijection entre un chemin de Dyck et un arbre binaire complet.

Dans la première section, nous définissons le *système caractéristique de polynômes* associé à une grammaire d'objets qui nous permet de caractériser exactement la structure de la grammaire ; en particulier, pour une grammaire de dimension 1, le système est réduit à un polynôme où le coefficient de x^i est le nombre d'opérations d'objets d'arité i, et le terme constant le nombre d'objets terminaux. Ainsi, dans la seconde section, nous pouvons définir de manière précise la notion de grammaires d'objets isomorphes, puis celle de *classes* de grammaires d'objets qui sont isomorphes. La troisième section est consacrée à l'étude d'une transformation sur les grammaires d'objets, que nous nommons *S-itération*. Elle est l'analogue d'une transformation connue dans divers domaines comme les systèmes de réécritures, les techniques d'évaluation partielle et les transformations de programmes et de systèmes réguliers : le *dépliage* ("unfolding"). On peut par exemple se référer aux articles de Burstall et Darlington [24] et de Courcelle [30]. La S-itération nous permet de déduire le résultat central du chapitre : à partir de deux grammaires d'objets non ambiguës de dimension 1 ayant au moins une opération d'objets d'arité ≥ 2, on peut construire deux grammaires d'objets isomorphes engendrant les mêmes ensembles d'objets que les grammaires initiales. La grammaire classique des mots de Dyck, $D = \epsilon + xD\overline{x}D$, entre notamment dans ce cadre, et alors, pour tous les objets qui peuvent se décrire par une grammaire d'objets non ambiguë de dimension 1 ayant au moins une opération d'objets d'arité ≥ 2, nous savons construire une bijection avec les mots de Dyck. Ce résultat apporte ainsi une explication à la constatation du rôle si important joué par les mots de Dyck dans le codage des objets combinatoires, constatation soulignée notamment par Viennot dans [91]. Enfin, nous présentons dans la dernière section quelques directions de recherche nouvelles que nous souhaitons suivre.

3.1 Système caractéristique de polynômes

3.1.1 Préliminaires

Nous notons X_n l'ensemble de variables commutatives $\{x_1, \ldots, x_n\}$, et $\mathbb{N}[X_n]$ le demi-anneau de polynômes à variables dans X_n et à coefficients dans $\mathbb{N}$. Un élément de ce demi-anneau s'écrit

$$P(x_1, \ldots, x_n) = \sum_{i_1, \ldots, i_n} c_{i_1, \ldots, i_n} \, {x_1}^{i_1} \ldots {x_n}^{i_n}$$

où il n'y a qu'un nombre fini de coefficients $c_{i_1,\ldots,i_n}$ non nuls.
Dans la suite, nous noterons $\boldsymbol{i}$ le n-uplet $(i_1, \ldots, i_n)$ et $x^{\boldsymbol{i}} = {x_1}^{i_1} \ldots {x_n}^{i_n}$. Avec ces notations, nous avons

$$P(x_1, \ldots, x_n) = \sum_{\boldsymbol{i}} c_{\boldsymbol{i}} \, x^{\boldsymbol{i}}.$$

Nous noterons également $c_{\boldsymbol{i}} =< x^{\boldsymbol{i}}, P >$ le coefficient de $x^{\boldsymbol{i}}$ dans P.

Le *degré* en x_j de P, pour $j \in [1, \ldots, n]$, est le plus grand des entiers i_j pour tous les n-uplets $\boldsymbol{i}$ tels que $c_{\boldsymbol{i}} \neq 0$, et le *degré total* de P le plus grand des entiers $|\boldsymbol{i}| = i_1 + \cdots + i_n$ pour tous les n-uplets $\boldsymbol{i}$ tels que $c_{\boldsymbol{i}} \neq 0$. Le degré total de P est noté $d^\circ(P)$.

3.1.2 Définitions

Soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets de dimension n. Rappelons sa notation équationnelle :

$$\left\{ E_i = \sum_{e_i \in T_{E_i}} e_i \; + \sum_{dest(\phi) = E_i} \phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}}) \right\}_{i=1,\ldots,n}.$$

Nous associons à chaque ensemble d'objets une variable de X_n par une bijection $\sigma : \mathcal{E} \longrightarrow X_n$. Pour alléger les notations, supposons que $\sigma(E_i) = x_i$ pour tout i dans $[1, \ldots, n]$.

Définition 3.1 *On appelle <u>système caractéristique de polynômes</u> associé à* G*, et on note* $\mathcal{S}^\sigma_{cp}(G)$*, le système de couples* (x_i, P_i) *où* P_i *est le polynôme de* $\mathbb{N}[X_n]$ *construit de la façon suivante : on transforme la partie droite de l'équation définissant* E_i *dans* G *en remplaçant*

- *chaque objet terminal* e_i *par la valeur* 1*,*
- *chaque expression de la forme* $\phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}})$ *par le monôme* $x_{i_{1_\phi}} \ldots x_{i_{k_\phi}}$*.*

Nous obtenons alors le système caractéristique de polynômes

$$\mathcal{S}^\sigma_{cp}(G) \;=\; \left\{ (x_i, P_i) \;/\; P_i \;=\; |T_{E_i}| \;+ \sum_{dest(\phi) = E_i} x_{i_{1_\phi}} \ldots x_{i_{k_\phi}} \right\}_{i=1,\ldots,n}.$$

Remarque : C'est en fait un passage du non-commutatif au commutatif. Les informations sauvegardées pour chaque E_i sont les suivantes : le nombre d'objets terminaux, $|T_{E_i}| = < x^{\mathbf{o}}, P_i >$, et pour $\boldsymbol{j} = (j_1, \ldots, j_n) \neq \mathbf{o}$, le coefficient $< x^{\boldsymbol{j}}, P_i >$ est le nombre d'opérations d'objets ϕ avec $dest(\phi) = E_i$ dont le domaine de définition $dom(\phi)$ a pour composantes $E_{j_1}, \ldots, E_{j_n}$, sans tenir compte de l'ordre de ces composantes.

Remarque : Lorsque la grammaire d'objets G est de dimension 1, $\mathcal{S}^{\sigma}_{cp}(G)$ est alors réduit à un polynôme de $\mathbb{N}[X]$, où $X = X_1 = \{x\}$; $< x^0, \mathcal{S}^{\sigma}_{cp}(G) >$ est égal au nombre d'objets terminaux, et pour $j > 0$, $< x^j, \mathcal{S}^{\sigma}_{cp}(G) >$ est égal au nombre d'opérations d'objets d'arité j.

Exemple 3.1 : Le polynôme caractéristique de la grammaire d'objets des polyominos parallélogrammes Figure 16 est $1 + 2x + x^2$. En effet, cette grammaire d'objets compte un objet terminal ($\square$), deux opérations d'objets d'arité 1 (ϕ_1 et ϕ_2) et une opération d'objets d'arité 2 (ϕ_3).

Exemple 3.2 : Les polyominos convexes dirigés E_{pcd} sont engendrés par la grammaire d'objets Figure 18, que nous nommons $G_{E_{pcd}}$. Soient $X_3 = \{x_1, x_2, x_3\}$, et $\sigma(E_{pcd}) = x_1$, $\sigma(E_{pcdm}) = x_2$, $\sigma(E_{pp}) = x_3$. Le système caractéristique de polynômes associé est alors le suivant :

$$\begin{aligned}\mathcal{S}^{\sigma}_{cp}(G_{E_{pcd}}) \;=\; \{ \quad & (x_1\ ,\ 1 + x_1 + x_1x_3 + x_2), \\ & (x_2\ ,\ x_1 + x_2 + x_2x_3), \\ & (x_3\ ,\ 1 + 2x_3 + {x_3}^2) \quad \}\ .\end{aligned}$$

Définition 3.2 *Le <u>degré</u> de G est le n-uplet $(d_1, \ldots, d_n)$, où d_i est le degré total du polynôme P_i du couple (x_i, P_i) de $\mathcal{S}^{\sigma}_{cp}(G)$.*

3.2 Construction de bijections

La notion de grammaires d'objets *isomorphes* introduite ici permet de décrire simplement des bijections entre ensembles d'objets. Une nouvelle bijection entre les polyominos convexes dirigés et les chemins du Grand Dyck est alors décrite dans ce cadre. Enfin, nous donnons la définition de *classe de grammaires d'objets* qui sera ensuite étendue dans la prochaine section.

3.2.1 Grammaires d'objets isomorphes

Définition 3.3 *Soient $G = < \mathcal{E}, \mathcal{T}, \mathcal{P} >$ et $G' = < \mathcal{E}', \mathcal{T}', \mathcal{P}' >$ deux grammaires d'objets de même dimension n. On se donne $X_n = \{x_1, \ldots, x_n\}$ un ensemble de variables commutatives.*

On dit que G et G' sont <u>*isomorphes*</u> *s'il existe deux bijections $\sigma : \mathcal{E} \to X_n$ et $\sigma' : \mathcal{E}' \to X_n$ telles que*

$$\mathcal{S}_{cp}^{\sigma}(G) = \mathcal{S}_{cp}^{\sigma'}(G') \ .$$

Théorème 3.4 *Soient G et G' deux grammaires d'objets isomorphes. Si G et G' sont non ambiguës, alors on peut construire une bijection entre les objets engendrés par G et ceux engendrés par G'.*

Exemple 3.3 : A la Section 2.5 du Chapitre 2, nous avons mis en évidence une bijection entre l'ensemble des chemins de Motzkin k-colorés et celui des arbres planaires k-colorés en montrant l'égalité entre leur fonction génératrice respective selon certains paramètres. Et en effet, les deux grammaires d'objets qui les engendrent Figures 20 et 23 sont isomorphes.

La démonstration du Théorème 3.4 consiste à établir une bijection entre les arbres de dérivation des deux grammaires d'objets.

<u>PREUVE :</u>

Soient $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ et $G' =< \mathcal{E}', \mathcal{T}', \mathcal{P}' >$ deux grammaires d'objets non ambiguës isomorphes de dimension n :

$$G = \left\{ E_i = \sum_{e_i \in T_{E_i}} e_i \;+ \sum_{dest(\phi)=E_i} \phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}}) \right\}_{i=1,\ldots,n} ,$$

$$G' = \left\{ E_i' = \sum_{e_i' \in T_{E_i'}} e_i' \;+ \sum_{dest(\phi')=E_i'} \phi(E'_{i_{1_{\phi'}}}, \ldots, E'_{i_{k_{\phi'}}}) \right\}_{i=1,\ldots,n} .$$

Alors il existe deux bijections $\sigma : \mathcal{E} \to X_n$ et $\sigma' : \mathcal{E}' \to X_n$ telles que

$$\mathcal{S}_{cp}^{\sigma}(G) = \mathcal{S}_{cp}^{\sigma'}(G') \ .$$

Pour simplifier les notations, supposons que la numérotation est telle que, pour tout $i \in [1, n]$,

$$\sigma(E_i) = \sigma'(E_i') = x_i \text{ , de telle sorte que } \; P_i = P_i' \ .$$

Nous montrons alors que pour tout $i \in [1, n]$, il y a une bijection entre $\mathcal{O}_G(E_i)$ et $\mathcal{O}_{G'}(E_i')$. Il suffit pour cela d'établir pour chaque couple (E_i, E_i'), deux bijections : une entre les objets terminaux, $\Pi_i : T_{E_i} \to T'_{E_i'}$, et une entre les opérations d'objets, $\Theta_i : \mathcal{P}_{E_i} \to \mathcal{P}'_{E_i'}$ où $\mathcal{P}_{E_i}$ désigne l'ensemble $\{\phi \in \mathcal{P} \ / \ dest(\phi) = E_i\}$ et $\mathcal{P}'_{E_i'}$ l'ensemble $\{\phi' \in \mathcal{P}' \ / \ dest(\phi') = E_i'\}$. En effet, comme $P_i = P_i'$, nous avons :

- $|T_{E_i}| = < x^{\mathbf{o}}, P_i > = < x^{\mathbf{o}}, P'_i > = |T'_{E'_i}|$, il y a ainsi $|T_{E_i}|!$ façons de construire la bijection Π_i,

- pour $\boldsymbol{j} = (j_1, \ldots, j_n) \neq \mathbf{o}$, $< x^{\boldsymbol{j}}, P_i > = < x^{\boldsymbol{j}}, P'_i >$, nous mettons alors en bijection les opérations d'objets contribuant au même monôme dans P_i et P'_i; il y a donc encore là plusieurs manières de construire la bijection Θ_i.

Nous notons Π la réunion des Π_i et Θ la réunion des Θ_i, c'est à dire si $o \in T_{E_i}$ alors $\Pi(o) = \Pi_i(o)$, et si $\phi \in \mathcal{P}_{E_i}$ alors $\Theta(\phi) = \Theta_i(\phi)$.

Pour obtenir deux objets en bijection, il suffit de réaliser la correspondance entre les arbres de dérivation dans chacune des grammaires. Soit $o_i \in \mathcal{O}_G(E_i)$ et soit A son arbre de dérivation sur E_i dans G (non ambiguë). Nous transformons A en A' de la manière suivante :

- toute feuille, étiquetée par un objet terminal o de G, est alors étiquetée par l'objet terminal correspondant dans G', $\Pi(o)$,

- tout nœud interne, étiqueté par une opération d'objets ϕ de $\mathcal{P}$, est alors étiqueté par $\phi' = \Theta(\phi)$ de $\mathcal{P}'$, et l'ordre de ses fils est réorganisé selon la permutation qui lie les composantes des $dom(\phi)$ et $dom(\phi')$.

A' est alors un arbre de dérivation sur E'_i dans G' et $ev(A') = o'_i \in \mathcal{O}_{G'}(E'_i)$. Cette transformation sur l'arbre de dérivation étant bijective, $o_i \in \mathcal{O}_G(E_i)$ et $o'_i \in \mathcal{O}_{G'}(E'_i)$ sont en bijection. Et donc, $\mathcal{O}_G(E_i)$ est en bijection avec $\mathcal{O}_{G'}(E'_i)$ pour tout E_i de $\mathcal{E}$ et tout E'_i de $\mathcal{E}'$.

□

Exemple 3.4 : Dans les travaux de Delest et Fédou [40] (ou Delest et Viennot [42]) sur l'énumération des polyominos parallélogrammes selon la largeur et l'aire, se retrouve sous-jacente, la bijection avec les chemins de Dyck (non vides) exhibée Figure 26 par deux grammaires d'objets isomorphes. Ces deux grammaires d'objets permettent de lire très clairement cette correspondance : le nombre de colonnes d'un polyomino parallélogramme correspond au nombre de pics du chemin de Dyck correspondant, et la hauteur des colonnes à la hauteur des pics. Un exemple de bijection entre un polyomino parallélogramme et un chemin de Dyck est donné Figure 27.

Nous remarquons qu'une autre bijection est possible; en effet, les grammaires d'objets de la Figure 26 ont deux opérations d'objets d'arité 1, et nous pouvons échanger la correspondance que nous avons établie. Mais dans ce cas, la bijection ne conserve pas les paramètres cités ci-dessus.

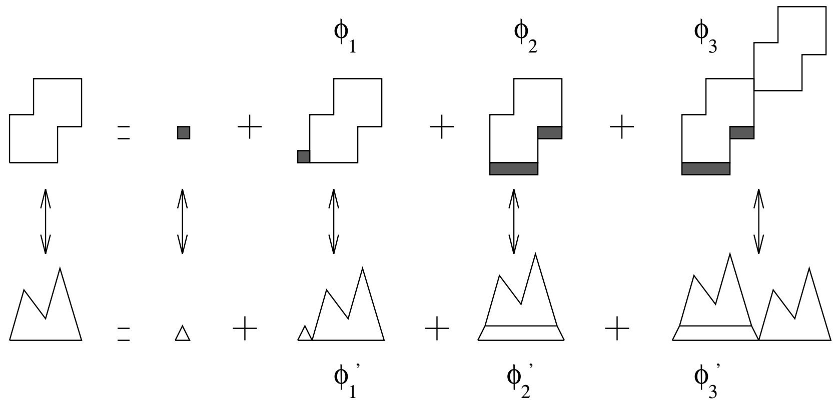


Figure 26: Polyominos parallélogrammes et chemins de Dyck.

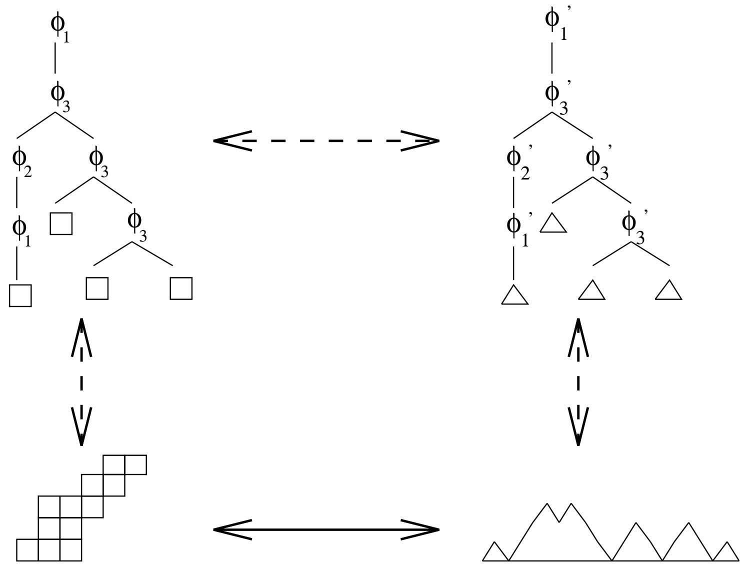


Figure 27: Bijection entre un polyomino parallélogramme et un chemin de Dyck.

3.2.2 Polyominos convexes dirigés et chemins du Grand Dyck

Dans [18], Bousquet-Mélou établit une bijection entre les polyominos convexes dirigés de périmètre $2n+4$ et les chemins du Grand Dyck de longueur $2n$, en utilisant une certaine décomposition récursive des polyominos. Nous établissons ici une nouvelle bijection entre les polyominos convexes dirigés et les chemins du Grand Dyck en exhibant deux grammaires d'objets isomorphes conservant la correspondance de ces paramètres. Notre bijection, comme celle de Bousquet-Mélou, permet de retrouver bijectivement le résultat étonnamment simple de Lin et Chang [72] : le nombre de polyominos convexes dirigés de périmètre $2n+4$ est égal au coefficient binomial $\binom{2n}{n}$.

Comme les chemins de Dyck, les chemins du Grand Dyck sont composés de pas Nord-Est et Sud-Est. Ils commencent et finissent également à la même hauteur, l'axe des abscisses, mais peuvent évoluer de part et d'autre de cet axe.

Proposition 3.5 *La Figure 28 décrit deux grammaires d'objets isomorphes non ambiguës pour les polyominos convexes dirigés (CD) et les chemins du Grand Dyck (GD), ce qui induit une bijection entre ces deux ensembles d'objets.*

PREUVE :
A la Section 1.7.3 du Chapitre 1, nous avons montré que la grammaire d'objets CD engendre de façon non ambiguë l'ensemble des polyominos convexes dirigés. Il reste donc à le faire pour les chemins du Grand Dyck GD.

Trois ensembles d'objets interviennent pour décrire ces chemins : les chemins de Grand Dyck eux-mêmes (E_{cgd}), les chemins terminant un pas en-dessous de l'axe des abscisses (E_{cm}) et les chemins de Dyck "retournés" (E_{cdr}).

Si C est un chemin du Grand Dyck, nous avons les cas disjoints suivants :

- soit C est le chemin vide,
- soit il termine par un pas montant, et le chemin sans ce pas appartient à E_{cm} : $C \in Im(\theta'_3)$,
- sinon, il termine par un pas descendant et,
  - soit il commence par un pas montant, et entre les deux on a un chemin de E_{cgd} : $C \in Im(\theta'_1)$,
  - soit il commence par un pas descendant et on coupe le chemin au premier pas qui repasse au-dessus de l'axe des abscisses obtenant ainsi un chemin de E_{cdr} suivi entre deux pas d'un chemin de E_{cgd} : $C \in Im(\theta'_2)$.

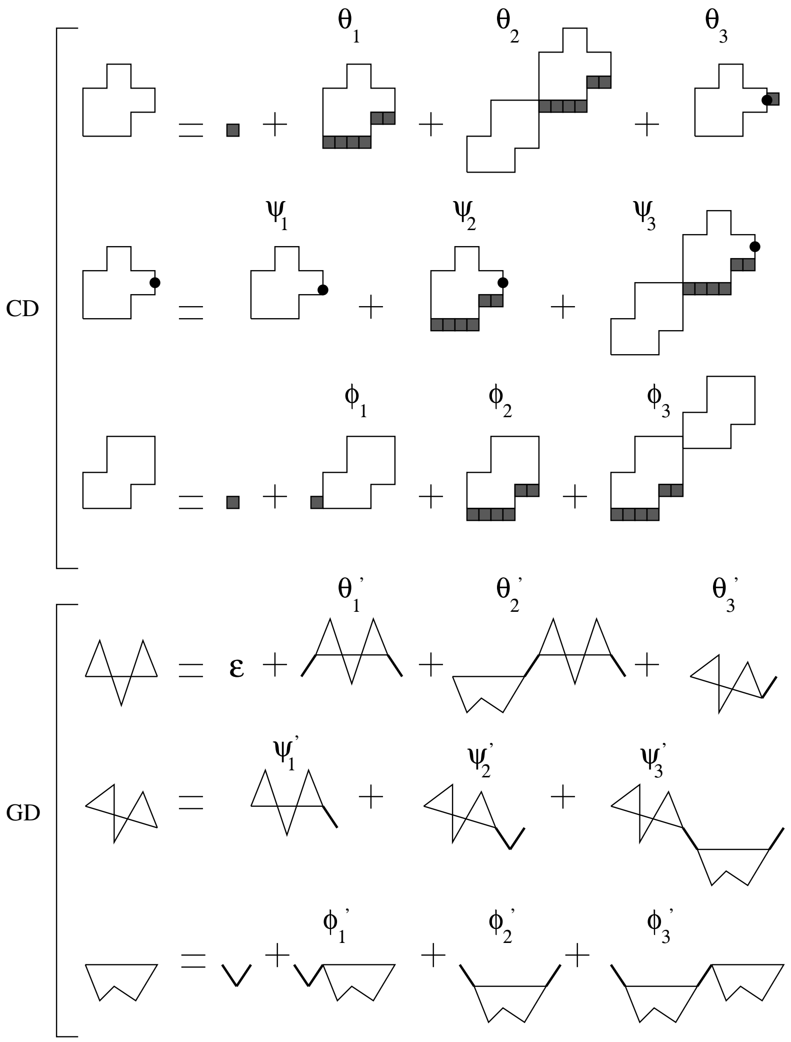


Figure 28: Polyominos convexes dirigés et chemins du Grand Dyck.

Ainsi, E_{cgd} est la réunion disjointe $\{\epsilon\} \bigcup Im(\theta'_1) \bigcup Im(\theta'_2) \bigcup Im(\theta'_3)$. Or, les opérations d'objets θ'_1, θ'_2, θ'_3 sont injectives : la première équation de la grammaire d'objets GD décrit cette décomposition non ambiguë.

Si M est un chemin terminant un pas en-dessous de l'axe des abscisses, nous avons les cas disjoints suivants :

- soit il termine par un pas descendant et le chemin sans ce pas appartient à E_{cgd} : $M \in Im(\psi'_1)$,
- sinon, il termine par un pas montant et,
  - soit ce pas est précédé d'un pas descendant et le chemin sans ces deux pas appartient à E_{cm} : $M \in Im(\psi'_2)$,
  - soit, on coupe le chemin au dernier pas descendant se trouvant à la même hauteur que ce pas montant, obtenant ainsi entre ces deux pas un chemin de E_{cdr} précédé d'un chemin de E_{cm} : $M \in Im(\psi'_3)$.

Ainsi, E_{cm} est la réunion disjointe $Im(\psi'_1) \bigcup Im(\psi'_2) \bigcup Im(\psi'_3)$. Or, les opérations d'objets ψ'_1, ψ'_2, ψ'_3 sont injectives : la deuxième équation la grammaire d'objets GD décrit cette décomposition non ambiguë.

La troisième équation la grammaire d'objets GD décrit la décomposition dite "de Dyck" non ambiguë bien connue pour les chemins de Dyck.

Les deux grammaires d'objets CD et GD Figure 28 sont non ambiguës et isomorphes, elle décrivent ainsi une bijection entre les polyominos convexes dirigés et les chemins du Grand Dyck.

$\square$

Corollaire 3.6 *Les deux grammaires d'objets Figure 28 fournissent une preuve bijective du résultat de Lin et Chang [72] : le nombre de polyominos convexes dirigés de périmètre $2n+4$ est égal au coefficient binomial $\binom{2n}{n}$.*

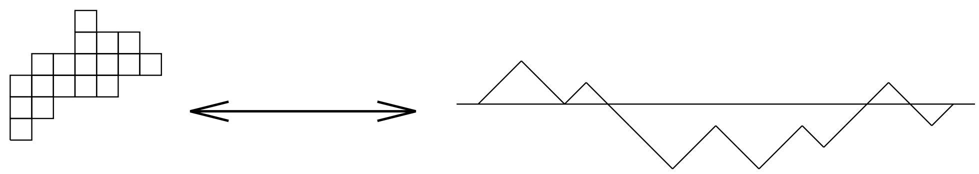


Figure 29: Bijection entre un polyomino convexe dirigé et un chemin du Grand Dyck.

PREUVE :
En examinant Figure 28 l'effet des opérations d'objets sur les paramètres (périmètre pour les polyominos et longueur pour les chemins), nous déduisons facilement que le périmètre d'un polyomino convexe dirigé est égal à la longueur du chemin du Grand Dyck qui lui correspond, augmentée de 4. Or, le nombre de chemins du Grand Dyck de longueur $2n$ est égal à $\binom{2n}{n}$. Donc, c'est aussi le nombre de polyominos convexes dirigés de périmètre $2n + 4$. Sur l'exemple de la Figure 29, le polyomino convexe dirigé est de périmètre 26 et le chemin de Dyck de longueur 22.
□

3.2.3 Classes de grammaires d'objets

La notion de grammaires d'objets isomorphes induit naturellement celle de *classe de grammaires d'objets* : famille de grammaires d'objets isomorphes non ambiguës. Le représentant d'une classe de grammaires d'objets C est un système de polynômes S_C de la forme :

$$S_C \quad = \quad \{(x_i, P_i) \ / \ P_i \in \mathbb{N}[X_n]\}_{i=1,\ldots,n} \ .$$

Une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ appartient à C s'il existe une bijection $\sigma : \mathcal{E} \to X_n$ telle que $S^{\sigma}_{cp}(G) = S_C$.

Nous pouvons alors construire des bijections entre les ensembles d'objets engendrés par des grammaires d'objets d'une même classe (Théorème 3.4). Nous donnons ci-dessous trois exemples de classes de grammaires d'objets : celle dite "de Dyck" de représentant $1 + x^2$, celle dite "de Dyck non vide" de représentant $1+2x+x^2$, et celle dite "de Motzkin" de représentant $1+x+x^2$.

Classe "de Dyck"

Les chemins de Dyck Figure 30 sont comptés par les nombres de Catalan $(1, 1, 2, 5, 14, 42, \ldots)$ selon la longueur. En fait, tous les ensembles d'objets engendrés par une grammaire isomorphe à celles des chemins de Dyck (un ensemble d'objets, un objet terminal, une opération d'objets d'arité 2, donc de polynôme caractéristique $1 + x^2$) sont en bijection et sont comptés par les nombres de Catalan selon un certain paramètre. Des exemples de tels objets sont donnés Figure 30 : les arbres binaires complets et les arbres planaires (nombre de sommets).

Classe "de Dyck non vide"

Les polyominos parallélogrammes et les chemins de Dyck non vides sont en bijection (Figure 26), et sont comptés par les nombres de Catalan $(1, 2, 5, 14, 42, \ldots)$ selon le périmètre pour les polyominos parallélogrammes, et selon la longueur pour les chemins de Dyck. En fait, tous les

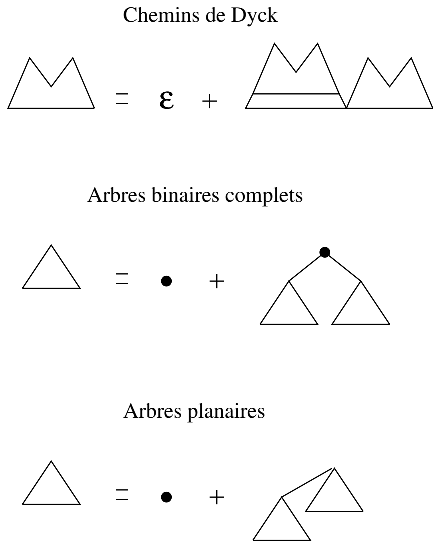


Figure 30: Quelques grammaires d'objets de la classe "de Dyck".

ensembles d'objets engendrés par une grammaire isomorphe à celles de la Figure 26 (un ensemble d'objets, un objet terminal, deux opérations d'arité 1 et une opération d'arité 2, donc de polynôme caractéristique $1 + 2x + x^2$) sont en bijection, et sont comptés par les nombres de Catalan selon un certain paramètre. Des exemples de tels objets sont donnés Figure 31 : les arbres binaires (nombre de sommets ou d'arêtes), les chemins de Motzkin bicolorés (longueur), les polyominos escalier (périmètre).

Classe "de Motzkin"

Tous les ensembles d'objets engendrés par une grammaire isomorphe à celle des chemins de Motzkin (un ensemble d'objets, un objet terminal, une opération d'arité 1 et une opération d'arité 2, donc de polynôme caractéristique $1+x+x^2$) sont comptés par les nombres de Motzkin $(1, 2, 4, 9, 21, \ldots)$ selon un certain paramètre. Des exemples de tels objets sont donnés Figure 32 : les chemins de Motzkin (longueur), les arbres 1-2-aires (nombre de sommets ou d'arêtes), les chemins de Dyck pour lesquels on interdit la suite de pas Nord-Est Sud-Est Nord-Est (longueur).

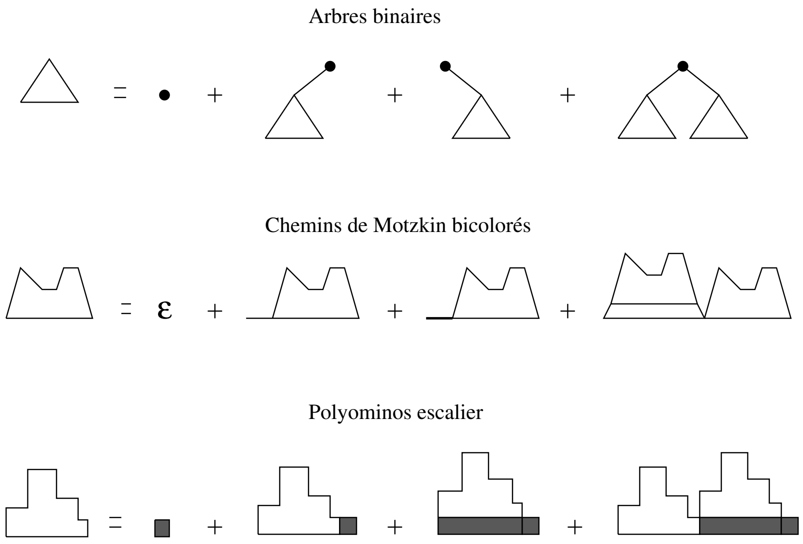


Figure 31: Quelques grammaires d'objets de la classe "de Dyck non vide".

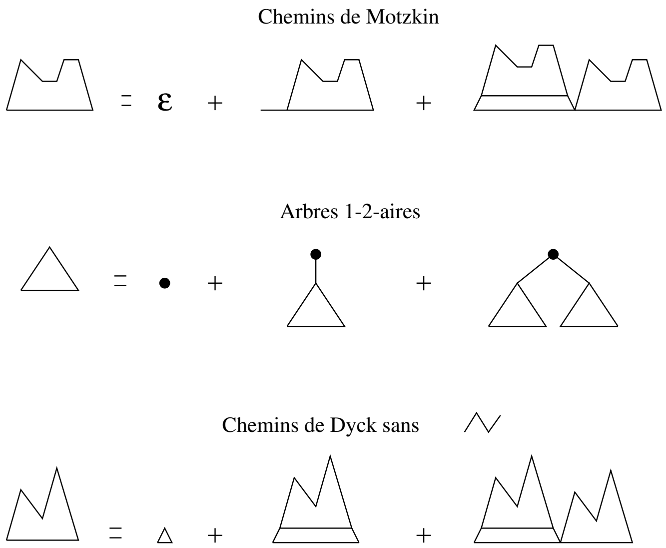


Figure 32: Quelques grammaires d'objets de la classe "de Motzkin".

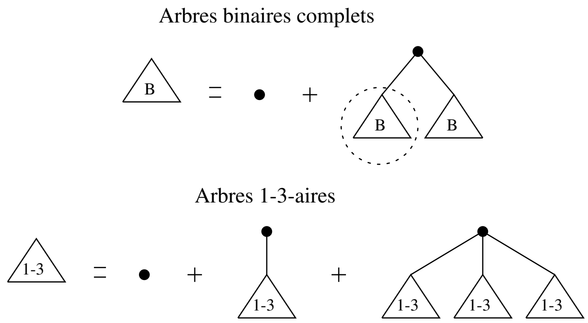


Figure 33: Arbres binaires complets et arbres 1-3-aires.

3.3 Itération de grammaires d'objets

3.3.1 Motivations

Nous venons de voir que des grammaires d'objets isomorphes induisent des constructions de bijections entre les ensembles d'objets qu'elles engendrent. Mais deux grammaires d'objets non isomorphes peuvent cacher des bijections sous-jacentes ... L'idée est de réaliser une suite de transformations des grammaires d'objets, qui conservent les ensembles d'objets engendrés, pour aboutir à des grammaires d'objets isomorphes.

Prenons l'exemple simple d'une grammaire d'objets pour les arbres binaires complets et d'une grammaire d'objets pour les arbres 1-3-aires représentées Figure 33. Ces deux grammaires d'objets sont de toute évidence non isomorphes. Nous transformons la grammaire d'objets des arbres binaires complets de la manière suivante : nous substituons l'instance de l'ensemble d'objets entourée de pointillés Figure 33 par l'équation toute entière, et en développant, nous obtenons la grammaire d'objets de la Figure 34. Cette nouvelle grammaire d'objets des arbres binaires complets est alors isomorphe à celle des arbres 1-3-aires Figure 33, ce qui induit une bijection entre ces deux ensembles d'objets. Nous nommons cette transformation sur les grammaires d'objets la *S-itération* (S pour Substitution). Dans d'autres domaines tels que les transformations de programmes ou de systèmes réguliers, une transformation analogue est connue sous le nom de *dépliage* ("unfolding") [24, 30]. Une question importante dans ces travaux est la correction

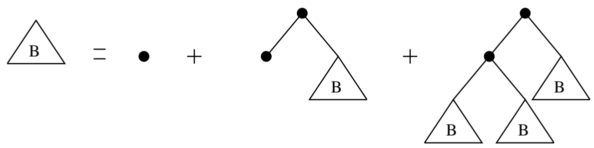


Figure 34: Une grammaire des arbres binaires complets isomorphe à celle des arbres 1-3-aires.

du système obtenu par dépliage, c'est à dire l'équivalence au système original. Ce genre de problèmes ne se pose pas pour la S-itération de grammaires d'objets car elle n'effectue qu'une simple réécriture de la grammaire d'objets initiale. Ainsi, les grammaires d'objets construites engendrent les mêmes ensembles d'objets, et sont donc équivalentes à la grammaire initiale.

Les résultats présentés dans la suite concernent uniquement la S-itération de grammaires d'objets de dimension 1 (une idée des problèmes soulevés par l'étude des grammaires d'objets de dimension ≥ 2 est donnée en Section 3.4). Le théorème fondamental de la section est le suivant :

Théorème 3.7 *Soient G_1 et G_2 deux grammaires d'objets non ambiguës de dimension 1 et de degré ≥ 2. Alors on peut construire, à partir de G_1 et G_2, deux grammaires d'objets isomorphes engendrant les mêmes ensembles d'objets que G_1 et G_2.*

Cela signifie donc que l'on peut construire des bijections entre tous les ensembles d'objets qui peuvent être engendrés par des grammaires d'objets non ambiguës de dimension 1 et de degré ≥ 2. Or, en particulier, c'est le cas pour un ensemble d'objets très connu et très important en combinatoire : l'ensemble des mots de Dyck, engendrés par la grammaire classique $D = \epsilon + xD\overline{x}D$. Ainsi, nous obtenons le corollaire suivant :

Corollaire 3.8 *On peut construire une bijection entre tout ensemble d'objets engendré par une grammaire d'objets non ambiguë de dimension 1 et de degré ≥ 2, et l'ensemble des mots de Dyck.*

Ce résultat apporte une explication à la constatation du rôle si important joué par les mots de Dyck dans le codage des objets combinatoires, constatation soulignée notamment par Viennot dans [91].

Dans la section suivante, nous décrivons précisément la S-itération de grammaires d'objets et de leur système caractéristique de polynômes associé. Nous détaillons plus spécialement le cas des grammaires d'objets de dimension 1 en énonçant les définitions et propriétés techniques utilisées par la suite. La Section 3.3.3 traite le cas particulier des grammaires d'objets de degré 1, tandis que la Section 3.3.4 est consacrée à l'élaboration d'une preuve constructive du Théorème 3.7. Enfin, des exemples d'application sont traités en Section 3.3.5, en particulier pour le Corollaire 3.8 (bijections avec les mots de Dyck).

3.3.2 La S-itération : définitions et propriétés

Description générale

La S-itération consiste à réécrire une grammaire d'objets en substituant, dans le domaine de définition d'une opération d'objets, une des instances d'un ensemble d'objets par l'équation qui le définit dans la grammaire.

Soit G une grammaire d'objets de dimension n,

$$G = \left\{ E_i = \sum_{e_i \in T_{E_i}} e_i + \sum_{dest(\phi)=E_i} \phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}}) \right\}_{i=1,\ldots,n} .$$

Soit ψ une opération d'objets de G avec $dest(\psi) = E_m$ et telle que l'une des composantes de $dom(\psi)$ est l'ensemble d'objets E_j. Effectuer la substitution de cette instance de E_j par l'équation qui le définit dans G consiste à remplacer dans l'équation qui définit E_m l'expression

$$\psi(E_{m_{1_\psi}}, \ldots, E_j, \ldots, E_{m_{k_\psi}})$$

par

$$\sum_{e_j \in T_{E_j}} \psi(E_{m_{1_\psi}}, \ldots, e_j, \ldots, E_{m_{k_\psi}}) + \sum_{dest(\phi)=E_j} \psi(E_{m_{1_\psi}}, \ldots, \phi(E_{j_{1_\phi}}, \ldots, E_{j_{k_\phi}}), \ldots, E_{m_{k_\psi}}) .$$

La grammaire d'objets G' ainsi obtenue engendre le même ensemble d'objets que G pour chaque E_i : $\mathcal{O}_{G'}(E_i) = \mathcal{O}_G(E_i)$ pour tout $i \in [1, n]$.

Nous appelons *itérée de la grammaire d'objets* G une grammaire d'objets obtenue par une succession finie de S-itérations à partir de G.

Rappelons que notre but est d'itérer des grammaires d'objets pour déterminer si parmi leurs itérées, il en existe qui sont isomorphes ; si c'est le cas, on dit que ce sont des grammaires d'objets *S-isomorphes*.

Ainsi, il suffit de ramener notre étude aux systèmes caractéristiques de polynômes et de reporter l'action de la S-itération sur ces systèmes de polynômes. Soit $\mathcal{S}^{\sigma}_{cp}(G)$ le système caractéristique de polynômes associé à G ($\sigma(E_i) = x_i$),

$$\mathcal{S}^{\sigma}_{cp}(G) = \left\{ (x_i, P_i) \ / \ P_i = |T_{E_i}| + \sum_{dest(\phi)=E_i} x_{i_{1_\phi}} \ldots x_{i_{k_\phi}} \right\}_{i=1,\ldots,n} .$$

Nous obtenons le système caractéristique de polynômes associé à G', $\mathcal{S}^{\sigma}_{cp}(G')$, en substituant, dans le polynôme P_m de $\mathcal{S}^{\sigma}_{cp}(G)$, la variable x_j du monôme $x_{m_{1_\psi}} \ldots x_j \ldots x_{m_{k_\psi}}$ par le polynôme P_j, puis en développant afin d'avoir de nouveau une somme de monômes.

Reprenons l'exemple des arbres binaires complets vu précédemment. Le système caractéristique de polynômes de leur grammaire d'objets Figure 33 est réduit au polynôme $1+x^2$. Nous calculons le polynôme caractéristique de leur grammaire d'objets Figure 34 en remplaçant la variable x correspondante du monôme x^2 par $1 + x^2$,

$$1 + (1 + x^2)x = 1 + x + x^3 .$$

Remarque : Nous pouvons voir sur l'exemple précédent que remplacer la "première" instance de x dans le monôme x^2 ou la "deuxième" est équivalent, car les variables sont commutatives. Ceci n'est pas le cas pour les grammaires d'objets, remplacer la première instance de l'ensemble d'objets ou la seconde aboutit à des grammaires d'objets différentes (car les opérations d'objets sont différentes), bien que isomorphes.

Grammaires d'objets de dimension 1 et polynômes caractéristiques

Les grammaires d'objets de dimension 1 que nous considérons comprennent au moins un objet terminal et au moins une opération d'objets. En effet, une grammaire d'objets n'ayant pas d'objet terminal engendre l'ensemble vide.

Etant donné que seul le caractère isomorphe nous intéresse ici, nous ramenons l'étude des grammaires d'objets de dimension 1 à celle de leurs polynômes caractéristiques. Ainsi, les polynômes considérés sont de terme constant non nul (au moins un objet terminal) et de degré ≥ 1 (au moins une opération d'objets). Dans la suite, nous notons $\mathbb{N}[X]_{GO}$ l'ensemble de ces polynômes. L'étude de la S-itération des polynômes de $\mathbb{N}[X]_{GO}$ nécessite quelques définitions et propriétés techniques énoncées ci-dessous.

Définition 3.9 *Soit P un polynôme de $\mathbb{N}[X]_{GO}$. Pour $i > 0$, on définit l'application suivante :*

$$\begin{array}{rcrcl} S_i^P & : & \{K \in \mathbb{N}[X]_{GO} / < x^i, K > \neq 0\} & \longrightarrow & \mathbb{N}[X]_{GO} \\ & & K & \longmapsto & K - x^i + x^{i-1}P\ . \end{array}$$

Le polynôme $S_i^P(K)$ est le résultat de la substitution d'une occurrence de la variable x d'un monôme x^i de K par le polynôme P.

Remarque : Soient G et G' deux grammaires d'objets respectivement associées à K et P engendrant le même ensemble d'objets E. Il y a alors plusieurs choix possibles pour construire une grammaire d'objets associée au polynôme $S_i^P(K)$; il faut choisir :

1. une opération d'objets ψ de G d'arité i,
2. une des i instances de l'ensemble d'objets E dans $dom(\psi)$ pour lui substituer l'équation définie par G'.

Ainsi, pour le seul polynôme $S_i^P(K)$, le nombre de grammaires d'objets possibles est ($i *$ nombre d'opérations d'objets de G d'arité i), c'est à dire $(i* < x^i, K >)$.

Nous déduisons alors deux propositions qui découlent directement de la définition précédente.

Proposition 3.10 *Si k le degré de K et p le degré de P, alors le degré de $S_i^P(K)$ est égal à $max(k, p+i-1)$.*

Proposition 3.11 *Si K est tel qu'il existe $i > 0$ et $j > 0$ avec $< x^i, K > \neq 0$ et $< x^j, K > \neq 0$, alors*

$$S_i^P(S_j^P(K)) = S_j^P(S_i^P(K)) \ .$$

Notations : Pour plus de lisibilité, on notera $S_i^P(S_j^P(K)) = S_i^P S_j^P(K)$. De même, l'application de r fois S_i^P au polynôme K (si $< x^i, K > \geq r$) sera notée

$$\underbrace{S_i^P \dots S_i^P}_{r\ fois}(K) = (S_i^P)^r(K) \ .$$

Nous pouvons alors définir la relation d'*itéré* entre deux polynômes.

Définition 3.12 *On dit que Q est un <u>P-itéré direct</u> de K, et on note $K \prec_P^0 Q$, s'il existe $i > 0$ tel que*

$$< x^i, K > \neq 0 \quad et \quad Q = S_i^P(K) \ .$$

Définissons la fermeture transitive de la relation précédente.

Définition 3.13 *On dit que Q est un <u>P-itéré</u> de K, et on note $K \prec_P Q$, s'il existe des polynômes $K_0, K_1, \dots, K_m$ de $\mathbb{N}[X]_{GO}$ tels que*

$$K = K_0 \prec_P^0 K_1 \prec_P^0 \dots \prec_P^0 K_m = Q \ .$$

On dit alors que Q est un <u>P-itéré de rang m</u> de K, et on pourra noter $K \prec_P^m Q$.

Dans la suite, nous nous intéressons au cas particulier où $P = K$. Nous introduisons alors la notation suivante :

Notations : On note $\underline{K \prec Q}$ lorsque $K \prec_K Q$, et on dit que Q est un <u>itéré</u> de K.

C'est cette notion sur les polynômes qui traduit celle, vue précédemment, de grammaire d'objets *itérée* d'une autre.

La relation $\prec$ satisfait deux propriétés très importantes qui sont à la base de nos démonstrations.

Propriété 3.14 *La relation $\prec$ est degré-croissante : si $K \prec Q$, alors $d°(K) \leq d°(Q)$.*

<u>PREUVE :</u>

La démonstration de cette propriété découle directement de la Proposition 3.10.

□

Remarque : Dans le cas particulier où $d°(K) = 1$, nous avons aussi $d°(Q) = 1$.

Propriété 3.15 *La relation $\prec$ est transitive.*

PREUVE :

Montrons pour cela que si $K \prec_K W$ et $W \prec_W Q$, alors $K \prec_K Q$.

$K \prec_K W \implies \exists K_0, K_1, \ldots, K_m \ / \ K = K_0 \prec^0_K K_1 \prec^0_K \ldots \prec^0_K K_m = W$.

$W \prec_W Q \implies \exists W_0, W_1, \ldots, W_n \ / \ W = W_0 \prec^0_W W_1 \prec^0_W \ldots \prec^0_W W_n = Q$.

Nous pouvons exprimer cela d'une autre façon :

$\exists i_1, \ldots, i_m \ / \ K_r = S^K_{i_r}(K_{r-1})$ pour $r \in [1, m]$, et alors $W = S^K_{i_m} \ldots S^K_{i_1}(K)$.

$\exists j_1, \ldots, j_n \ / \ W_r = S^W_{j_r}(W_{r-1})$ pour $r \in [1, n]$, et alors $Q = S^W_{j_n} \ldots S^W_{j_1}(W)$.

Ainsi, $S^W_{j_r}$ peut s'écrire $S^W_{j_r} = S^K_{j_r+i_m-1} \ldots S^K_{j_r+i_1-1} S^K_{j_r}$, et donc

$Q = \underbrace{S^K_{j_n+i_m-1} \ldots S^K_{j_n+i_1-1} S^K_{j_n}}_{S^W_{j_n}} \ldots\ldots \underbrace{S^K_{j_1+i_m-1} \ldots S^K_{j_1+i_1-1} S^K_{j_1}}_{S^W_{j_1}} \underbrace{S^K_{i_m} \ldots S^K_{i_1}(K)}_{W}.$

Et donc nous avons $K \prec_K Q$.

□

Enfin, nous traduisons la notion de grammaires d'objets S-isomorphes sur les polynômes.

Définition 3.16 *On dit que K est <u>S-isomorphe</u> à Q, et on note $K \equiv_S Q$, s'il existe un polynôme T de $\mathbb{N}[X]_{GO}$ tel que*

$$K \prec T \quad et \quad Q \prec T \ .$$

Les deux sections suivantes sont consacrées à la détermination des différentes classes de la relation $\equiv_S$. En effet, nous avons vu plus haut que les itérés des polynômes de degré 1 sont aussi de degré 1. Ainsi, nous traitons séparément le cas particulier des polynômes de degré 1 (Section 3.3.3) et celui des polynômes de degré ≥ 2 (Section 3.3.4).

3.3.3 Classification des grammaires d'objets de degré 1

La détermination des classes de grammaires d'objets S-isomorphes de dimension 1 et de degré 1 est ramenée à la classification des polynômes de $\mathbb{N}[X]_{GO}$ de degré 1 selon la relation $\equiv_S$.

Théorème 3.17 *Soient K et Q deux polynômes de $\mathbb{N}[X]_{GO}$ de degré 1, $K = k_0 + k_1 x$ et $Q = q_0 + q_1 x$ avec $k_0, k_1, q_0, q_1 \geq 1$. Alors, on a*

$$K \equiv_S Q \quad \Longleftrightarrow \quad \frac{k_1 - 1}{k_0} = \frac{q_1 - 1}{q_0} \ .$$

PREUVE :

Par définition, nous avons $K \equiv_S Q$ si et seulement si il existe deux entiers positifs ou nuls i_K et i_Q tels que

$$\underbrace{(S^K_1)^{i_K}(K)}_{P_K} = \underbrace{(S^Q_1)^{i_Q}(Q)}_{P_Q} \ .$$

En calculant explicitement P_K et P_Q, et nous trouvons :

$$\begin{cases} P_K &= (i_K+1)k_0 + ((i_K+1)(k_1-1)+1)\,x \ , \\ P_Q &= (i_Q+1)q_0 + ((i_Q+1)(q_1-1)+1)\,x \ . \end{cases}$$

Pour satisfaire la condition $P_K = P_Q$, nous devons vérifier alors les égalités suivantes :

$$\begin{cases} (i_K+1)\,k_0 &= (i_Q+1)\,q_0 \\ (i_K+1)(k_1-1) &= (i_Q+1)(q_1-1) \ . \end{cases}$$

Comme k_0, q_0, (i_K+1) et (i_Q+1) sont différents de 0, nous pouvons faire le rapport des deux égalités, et obtenir la condition :

$$\frac{k_1-1}{k_0} = \frac{q_1-1}{q_0} \ .$$

Nous construisons effectivement le polynôme $P_K = P_Q$ à partir de K et Q en donnant à i_K et i_Q les plus petites valeurs entières qui conviennent :

$$\begin{cases} i_K &= \frac{ppcm(k_0,q_0)}{k_0} - 1 &= \frac{ppcm(k_1-1,q_1-1)}{k_1-1} - 1 \\ i_Q &= \frac{ppcm(k_0,q_0)}{q_0} - 1 &= \frac{ppcm(k_1-1,q_1-1)}{q_1-1} - 1 \end{cases}$$

où *ppcm* désigne le plus petit commun multiple de deux entiers.

□

Remarque : Chaque nombre rationnel positif correspond à une classe de la relation $\equiv_S$ pour les polynômes de $\mathbb{N}[X]_{GO}$ de degré 1, et inversement.

3.3.4 Classification des grammaires d'objets de degré ≥ 2

Nous donnons ici une preuve du Théorème 3.7, énoncé en Section3.3.1, concernant les grammaires d'objets de dimension 1 et de degré ≥ 2. Pour cela, nous montrons un théorème équivalent sur les polynômes de $\mathbb{N}[X]_{GO}$ de degré ≥ 2.

Théorème 3.18 *Tous les polynômes de $\mathbb{N}[X]_{GO}$ de degré ≥ 2 sont S-isomorphes deux à deux.*

Avant de démontrer le Théorème 3.18, nous prouvons un lemme qui permet ensuite de restreindre l'étude aux polynômes de degré ≥ 2 dont tous les coefficients sont non nuls, et de minimiser ainsi les contraintes d'application des substitutions.

Lemme 3.19 *Tous les polynômes de $\mathbb{N}[X]_{GO}$ de degré ≥ 2 admettent un itéré dont tous les coefficients sont non nuls.*

PREUVE :

Nous voulons donc montrer que pour tout $K \in \mathbb{N}[X]_{GO}$ de degré ≥ 2, il existe un polynôme Q tel que $K \prec Q$ avec $< x^i, Q > \neq 0$ pour tout $i \in [0, d^\circ(Q)]$. Pour cela, nous procédons en deux étapes :

1. il existe W avec $< x, W > \neq 0$ tel que $K \prec W$,

2. il existe Q avec $< x^i, Q > \neq 0$, pour $i \in [0, d°(Q)]$ tel que $W \prec Q$,

et donc, par la Propriété 3.15, nous avons $K \prec Q$.

1. Soit $K = k_0 + k_1 x + \cdots + k_n x^n$. Soit $j \geq 1$ tel que $k_1 = \cdots = k_{j-1} = 0$ et $k_j \geq 1$. Nous montrons alors par récurrence sur j que, pour $P \in \mathbb{N}[X]_{GO}$ donné, il existe W avec $< x, W > \neq 0$ tel que $K \prec_P W$. Notons $p_0 =< x^0, P >$.

   Si $j = 1$, $k_1 \geq 1$ et c'est fini, $W = K$.

   Supposons la propriété vraie jusqu'à $j = r - 1$.

   Si $j = r$, calculons le polynôme $T = S_r^P(K)$:

   $$S_r^P(K) = k_0 + x^{r-1}.P + (k_r - 1)x^r + k_{r+1}x^{r+1} + \cdots + k_n x^n ,$$

   nous avons $< x^{r-1}, T >= p_0 \geq 1$, donc par hypothèse de récurrence, il existe un polynôme W avec $< x, W > \neq 0$ tel que $T \prec_P W$. Or $K \prec_P T$, donc par transitivité, nous avons $K \prec_P W$.

   Il suffit de remplacer P par K et alors $K \prec W$, avec la construction $W = S_2^K \ldots S_j^K(K)$.

2. La démonstration de la deuxième étape ressemble à celle de la première.
   Soit $W = w_0 + w_1 x + \cdots + w_m x^m$ ($w_1 \geq 1$). Soit $j \geq 2$ tel que $w_{j-1} = 0$ et $w_j, \ldots, w_m \geq 1$. Nous montrons alors par récurrence sur j que, pour $P \in \mathbb{N}[X]_{GO}$ donné avec $< x, P > \neq 0$, il existe Q avec $< x^i, Q > \neq 0$ pour $i \in [0, d°(Q)]$ tel que $W \prec_P Q$. Notons $p_0 =< x^0, P >$ et $p_1 =< x, P >$.

   Si $j = 2$, tous les coefficients de W sont non nuls, et c'est fini, $Q = W$.

   Supposons la propriété vraie jusqu'à $j = r - 1$.

   Si $j = r$, calculons le polynôme $T = S_r^P(W)$:

   $$S_r^P(W) = w_0 + \cdots + w_{r-2}x^{r-2} + x^{r-1}.P + (w_r - 1)x^r + w_{r+1}x^{r+1} + \cdots + w_m x^m ,$$

   nous avons $< x^{r-1}, T >= p_0 \geq 1$ et $< x^r, T) >= w_r - 1 + p_1 \geq 1$, donc par hypothèse de récurrence, il existe Q avec $< x^i, Q > \neq 0$ pour $i \in [0, d°(Q)]$ tel que $T \prec_P Q$. Or $W \prec_P T$, donc par transitivité, nous avons $W \prec_P Q$.

   Il suffit de remplacer P par W et alors $W \prec Q$, avec la construction $Q = S_3^W \ldots S_j^W(W)$.

□

Nous pouvons maintenant démontrer le Théorème 3.18.

PREUVE :

La démonstration est constructive : pour deux polynômes K et Q donnés, nous montrons que K et Q sont S-isomorphes en calculant explicitement un polynôme qui est à la fois un itéré de K et un itéré de Q. Nous donnons les suites de substitutions qu'il faut appliquer à K et à Q pour cela.

Grâce au Lemme 3.19 et à la Propriété 3.15, nous pouvons restreindre l'étude à des polynômes de degré ≥ 2 de coefficients tous non nuls. Ainsi, la seule contrainte à respecter pour appliquer une substitution S_i à un polynôme est de vérifier que i est inférieur ou égal au degré de ce polynôme. Nous éliminons cette dernière contrainte en appliquant les substitutions selon l'ordre croissant sur i : une suite de S_1, puis une suite de S_2, etc ...

Alors, pour montrer que $K \equiv_S Q$, nous prouvons qu'il existe deux entiers r_K et r_Q, et des entiers $N_1, \ldots, N_{r_K}$ et $M_1, \ldots, M_{r_Q}$ tels que :

$$\underbrace{(S_{r_K}^K)^{N_{r_K}} \ldots (S_1^K)^{N_1}(K)}_{P_K} = \underbrace{(S_{r_Q}^Q)^{M_{r_Q}} \ldots (S_1^Q)^{M_1}(Q)}_{P_Q} \ .$$

Lorsque le degré de K ou de Q ou des deux est égal à 2, nous devons prendre une précaution supplémentaire pour que la solution que nous donnons plus bas soit valide : si $d°(K) = 2$, alors nous devons avoir $< x, Q > \geq 2$, et inversement, si $d°(Q) = 2$, nous devons avoir $< x, K > \geq 2$. Nous réglons ce détail technique en remplaçant Q par $S_2^Q(Q)$ (resp. K par $S_2^K(K)$). Lorsque les degrés sont strictement plus grands que 2, la condition des coefficients tous non nuls suffit.

Soient K, de degré n, et Q, de degré m, de tels polynômes,

$$K = \sum_{i=0}^{n} k_i x^i \quad \text{et} \quad Q = \sum_{i=0}^{m} q_i x^i \ .$$

Nous montrons que la solution suivante convient : $r_K = m + 1$, $r_Q = n + 1$, et pour $z \geq 1$,

$$\begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ \vdots \\ N_{m+1} \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + z \begin{pmatrix} q_0 \\ q_1 - 1 \\ q_2 \\ \vdots \\ q_m \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n+1} \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + z \begin{pmatrix} k_0 \\ k_1 - 1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} .$$

Les polynômes P_K et P_Q sont clairement de degré $n + m$. Pour montrer que ces polynômes sont égaux, nous déterminons les expressions de $< x^i, P_K >$ et $< x^i, P_Q >$, pour $i \in [0, n + m]$. Prenons l'exemple de K (on suppose $n \geq m$). Voici la succession des substitutions appliquées à

K pour obtenir P_K.

		$n+m$		$n+1$	n		$m+1$	m		2	1	0	
		0	$\ldots$	0	k_n	$\ldots$	$\ldots$	k_m	$\ldots$	k_2	k_1	k_0	K
$+$	$N_1\times$	0	$\ldots$	0	k_n	$\ldots$	$\ldots$	k_m	$\ldots$	k_2	k_1-1	k_0	$(S_1^K)^{N_1}$
$+$	$N_2\times$	0	$\ldots$	k_n	k_{n-1}	$\ldots$	$\ldots$	k_{m-1}	$\ldots$	k_1-1	k_0		$(S_2^K)^{N_2}$
$+$													$\vdots$
$+$	$N_{m+1}\times$	k_n	$\ldots$	k_{n-m+1}	k_{n-m}	$\ldots$	k_1-1	k_0					$(S_{m+1}^K)^{N_{m+1}}$
		$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	P_K

Nous étendons K et Q au degré $n+m$:

$$K=\sum_{i=0}^{n+m} k_i x^i \quad \text{où } k_i = 0 \text{ pour } i>n, \qquad \text{et} \qquad Q=\sum_{i=0}^{n+m} q_i x^i \quad \text{où } q_i=0 \text{ pour } i>m.$$

Nous nous donnons les notations suivantes :

$$\text{pour} j\in[0,n+m]\ , \quad K_j=\begin{cases} k_1-1 & \text{si } j=1\\ k_j & \text{sinon}\end{cases} \qquad \text{et} \qquad Q_j=\begin{cases} q_1-1 & \text{si } j=1\\ q_j & \text{sinon}\end{cases}$$

ainsi,

$$\text{pour} j\in[0,n+m]\ , \quad N_{j+1}=\begin{cases} zq_0-1 & \text{si } j=0\\ zQ_j & \text{sinon}\end{cases} \qquad \text{et} \qquad M_{j+1}=\begin{cases} zk_0-1 & \text{si } j=0\\ zK_j & \text{sinon}\end{cases}$$

Alors, pour $i\in[0,n+m]$, nous avons :

$$\begin{cases} < x^i, P_K >= k_i+\displaystyle\sum_{j=0}^{i} N_{j+1}K_{i-j} & (1)\\ < x^i, P_Q >= q_i+\displaystyle\sum_{j=0}^{i} M_{j+1}Q_{i-j} & (2)\end{cases}$$

Montrons que (1) et (2) sont égaux pour $i\in[0,n+m]$.

cas $i=0$ **:**

$$(1)=k_0+N_1K_0=k_0+(zq_0-1)k_0=zk_0q_0$$
$$(2)=q_0+M_1Q_0=q_0+(zk_0-1)q_0=zk_0q_0$$

cas $i\geq 1$ **:** nous réécrivons (1) et (2)

$$\begin{aligned}(1) &= k_i+N_1K_i+N_{i+1}K_0+\sum_{j=1}^{i-1}N_{j+1}K_{i-j}\\ &= k_i+(zq_0-1)K_i+zQ_ik_0+\sum_{j=1}^{i-1}N_{j+1}K_{i-j}\\ &= k_i-K_i+\underbrace{zq_0K_i+zQ_ik_0+\sum_{j=1}^{i-1}N_{j+1}K_{i-j}}_{\alpha} \quad = \quad k_i-K_i+\alpha\ ,\end{aligned}$$

$$\begin{aligned}
(2) &= q_i + M_1 Q_i + M_{i+1} Q_0 + \sum_{j=1}^{i-1} M_{j+1} Q_{i-j} \\
&= q_i + (zk_0 - 1) Q_i + zK_i q_0 + \sum_{j=1}^{i-1} K_j N_{i-j+1} \\
&= q_i - Q_i + \underbrace{zk_0 Q_i + zK_i q_0 + \sum_{j=1}^{i-1} N_{j+1} K_{i-j}}_{\alpha} \quad = \quad q_i - Q_i + \alpha \;.
\end{aligned}$$

Or pour $i = 1$, $k_i - K_i = q_i - Q_i = 1$,
et pour $i \geq 2$, $k_i - K_i = q_i - Q_i = 0$.

Donc (1) = (2) pour tout $i \in [0, n+m]$, et $P_K = P_Q$, ce qui montre que $K \equiv_S Q$.

□

Exemple 3.5 : Appliquons la construction du Théorème 3.18 aux deux polynômes suivants : $P_D = 1 + 2x + x^2$, le polynôme caractéristique d'une grammaire d'objets pour les chemins de Dyck (non vides), et $P_M = 1 + x + x^2$, celui d'une grammaire d'objets pour les chemins de Motzkin (voir Figure 35).

Comme P_D et P_M sont tous les deux de degré 2 et que $< x, P_M >= 1$, nous devons prendre

$$K = P_D \quad \text{et} \quad Q = S_2^{P_M}(P_M) = 1 + 2x + x^2 + x^3.$$

La solution est alors la suivante :

$$(S_4^K)^{N_4} \ldots (S_1^K)^{N_1}(K) = (S_3^Q)^{M_3} \ldots (S_1^Q)^{M_1}(Q)$$

avec

$$\begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + z \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} + z \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} .$$

Prenant $z = 1$, nous obtenons le polynôme

$$S_4^K S_3^K S_2^K(K) = S_3^Q S_2^Q(Q) = 1 + 3x + 3x^2 + 3x^3 + 2x^4 + x^5.$$

Ainsi, nous avons $P_D \equiv_S P_M$.

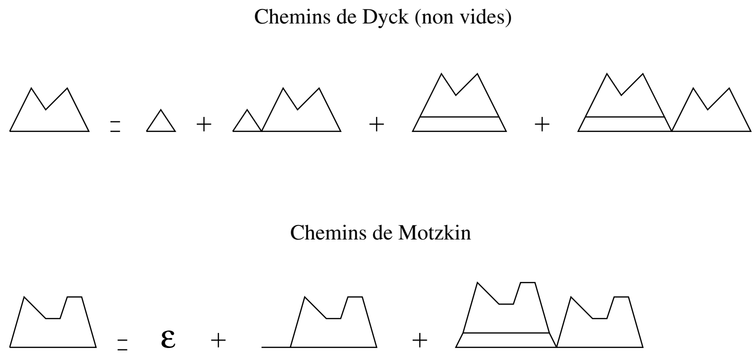


Figure 35: Grammaires d'objets classiques pour les chemins de Dyck et les chemins de Motzkin.

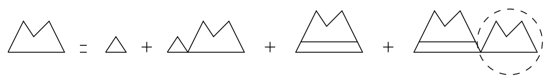
3.3.5 Bijections et mots de Dyck

Grâce à la preuve du Théorème 3.18 que nous avons établie à la section précédente, nous savons construire effectivement des grammaires d'objets isomorphes à partir de grammaires d'objets quelconques de dimension 1 et de degré ≥ 2; et ainsi, nous obtenons le Théorème 3.7. En effet, cette preuve fournit les suites de substitutions à effectuer pour chacun des polynômes caractéristiques, il suffit alors de les transposer sur les grammaires d'objets. Pour cela, rappelons que la substitution $S_i^P(K)$ signifie qu'une opération d'objets d'arité i est choisie dans la grammaire d'objets associée au polynôme K, et que l'une des i composantes de son domaine de définition est substitutée par l'équation définie par la grammaire d'objets associée au polynôme P.

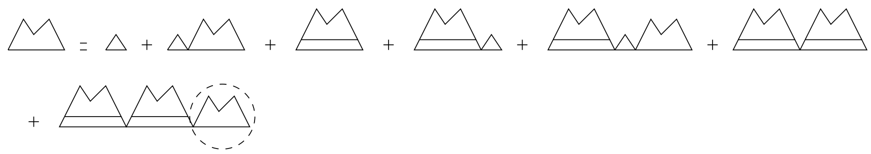
Exemple 3.6 : Reprenons l'exemple précédent des chemins de Dyck (non vides) et des chemins de Motzkin, et construisons deux grammaires d'objets isomorphes en appliquant les suites de substitutions que nous avons trouvées. Le cas des chemins de Dyck est traité en détail Figure 36. Le polynôme $P_D = K = 1 + 2x + x^2$ est le polynôme caractéristique de la grammaire d'objets initiale. Nous contruisons alors successivement des grammaires d'objets associées aux polynômes $K_1 = S_2^K(K) = 1 + 3x + 2x^2 + x^3$, puis $K_2 = S_3^K(K_1) = 1 + 3x + 3x^2 + 2x^3 + x^4$, et enfin $K_3 = S_4^K(K_2) = 1 + 3x + 3x^2 + 3x^3 + 2x^4 + x^5$. Pour cela, à chaque étape, nous choisissons de remplacer, par la grammaire d'objets de départ, l'instance de l'ensemble d'objets entourée de pointillés. En faisant de même pour les chemins de Motzkin, nous obtenons une grammaire d'objets Figure 37 isomorphe à celle construite pour les chemins de Dyck Figure 36 et reproduite Figure 37.

Le Théorème 3.18 signifie également que les polynômes de $\mathbb{N}[X]_{GO}$ de degré ≥ 2 forment une unique classe de la relation $\equiv_S$, dont le plus petit représentant est le polynôme $1 + x^2$. Nous obtenons alors le corollaire suivant, équivalent au Corollaire 3.8 pour les grammaires d'objets.

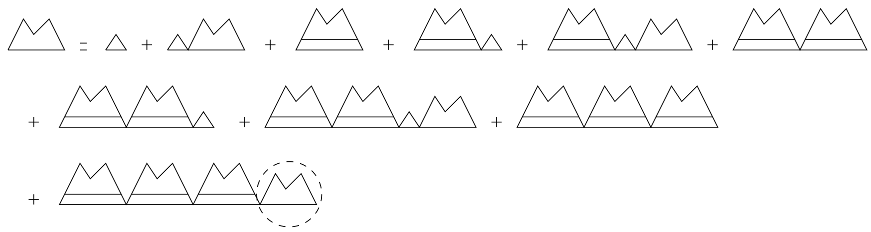
$P_D = K = 1 + 2x + x^2$



$K_1 = S_2^K(K) = 1 + 3x + 2x^2 + x^3$



$K_2 = S_3^K(K_1) = 1 + 3x + 3x^2 + 2x^3 + x^4$



$K_3 = S_4^K(K_2) = 1 + 3x + 3x^2 + 3x^3 + 2x^4 + x^5$

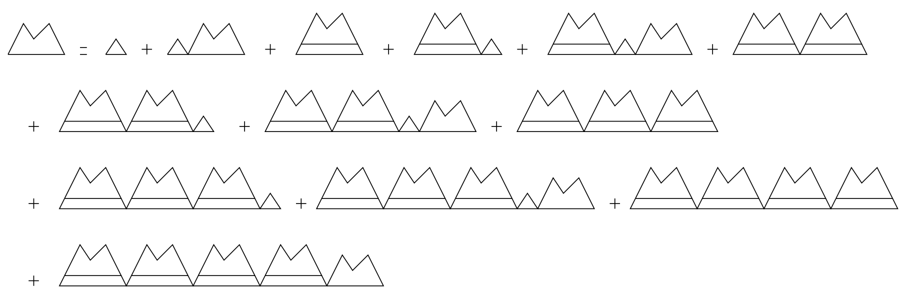


Figure 36: Suite de substitutions pour la grammaire d'objets des chemins de Dyck (non vides).

Corollaire 3.20 *Tous les polynômes de $\mathbb{N}[X]_{GO}$ de degré ≥ 2 sont S-isomorphes au polynôme $1 + x^2$, polynôme caractéristique de la grammaire d'objets classique pour les mots de Dyck : $D = \epsilon + xD\overline{x}D$.*

Ainsi, pour tous les objets qui peuvent se décrire par une grammaire d'objets non ambiguë de dimension 1 et de degré ≥ 2, nous savons construire une bijection avec les mots de Dyck en exhibant des grammaires d'objets isomorphes. Ce résultat donne un élément de réponse à cette remarque interrogative faite notamment par Viennot dans [91] : "Pourquoi les mots de Dyck jouent-ils un rôle si important dans le codage des objets combinatoires ?". En effet, dans la littérature combinatoire, le codage par les mots de Dyck (ou des mots très proches) est

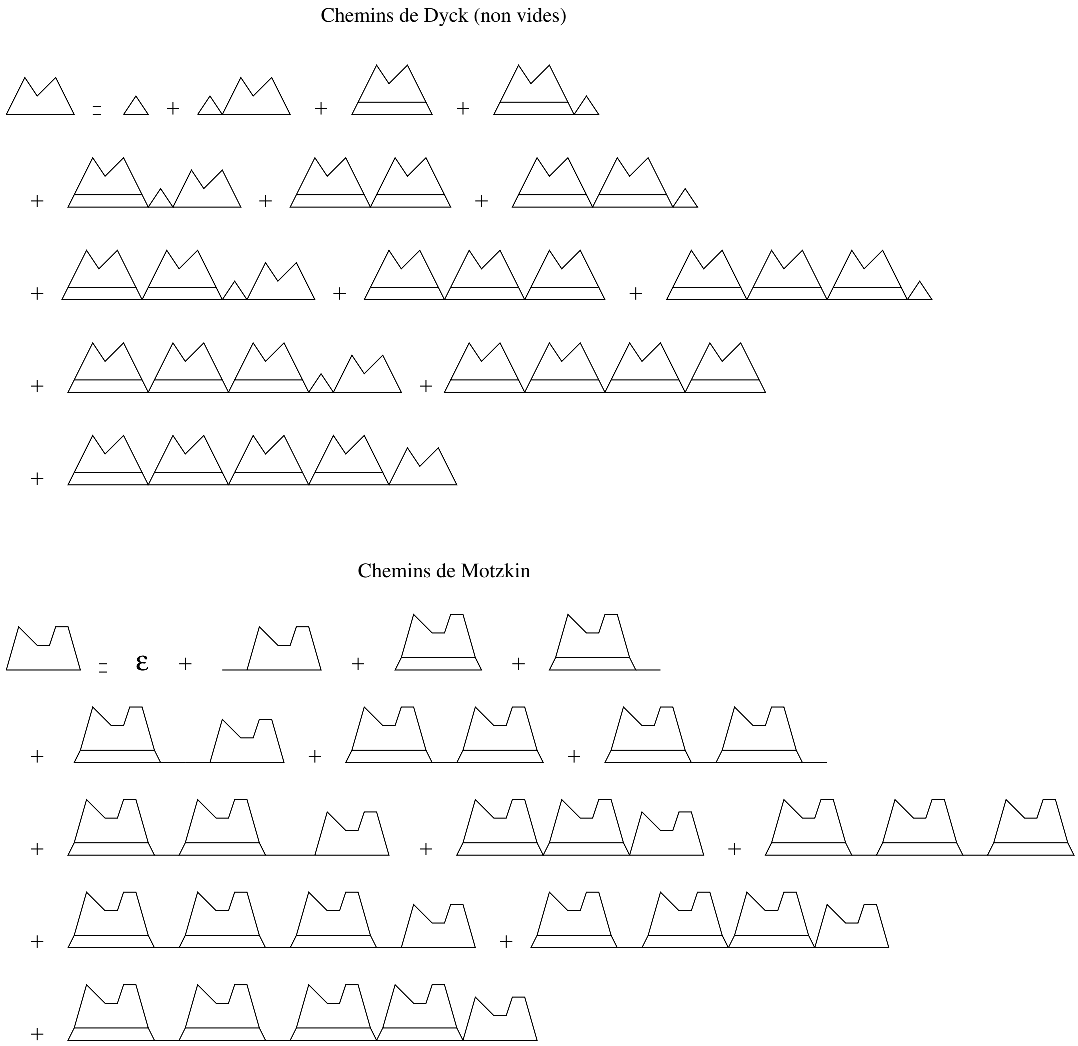


Figure 37: Grammaires d'objets isomorphes pour les chemins de Dyck (non vides) et les chemins de Motzkin.

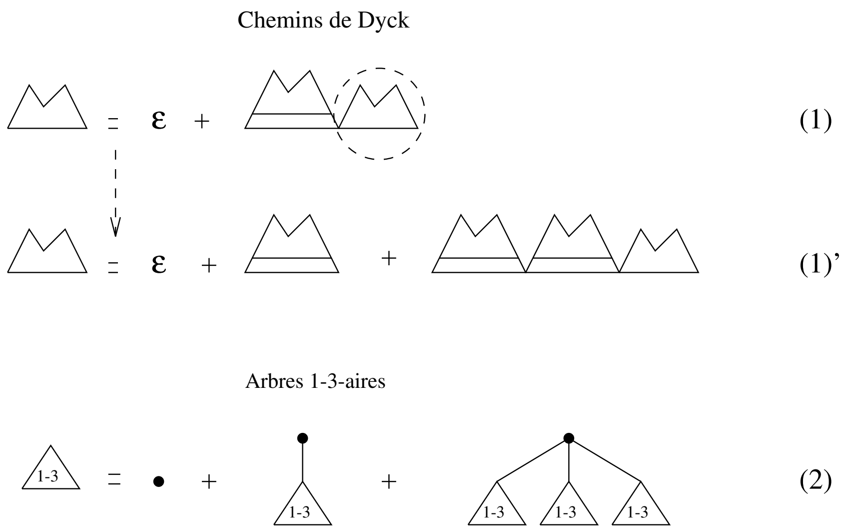


Figure 38: Grammaires d'objets isomorphes pour les chemins de Dyck et les arbres 1-3-aires.

très souvent évoqué pour différentes classes d'objets : des arbres, des polyominos, des cartes planaires, etc. Notre corollaire vient effectivement confirmer le fait que beaucoup d'objets sont étroitement liés aux mots de Dyck. Nous illustrons ceci en donnant trois exemples de construction de bijection avec les mots de Dyck : les arbres 1-3-aires, les mots de Motzkin, et les polyominos parallélogrammes.

Exemple 3.7 : Les chemins de Dyck et les arbres 1-3-aires sont engendrés respectivement par les grammaires d'objets (1) et (2) de la Figure 38, de polynôme caractéristique $P_D = 1 + x^2$ et $P_A = 1 + x + x^3$. Ces polynômes ne sont pas de coefficients tous non nuls, donc nous devons appliquer le Lemme 3.19. Or, la première étape de calcul pour le polynôme P_D nous amène à

$$S_2^{P_D}(P_D) = 1 + x + x^3 = P_A\ .$$

Donc, une seule substitution suffit pour obtenir des grammaires d'objets isomorphes, les grammaires d'objets $(1)'$ et (2) de la Figure 38. C'est en fait le même exemple que l'exemple introductif de la Section 3.3.1 avec les arbres binaires complets.

Remarque : Les grammaires d'objets isomorphes de la Figure 38 permettent de déduire simplement le passage de deux paramètres par la bijection :

le nombre de pics du chemin de Dyck est égal au nombre de feuilles de l'arbre 1-3-aire qui ne sont pas fils droit d'un sommet d'arité 3,

la somme des hauteurs des pics du chemin est égale à la somme des hauteurs de ces feuilles où les arêtes droites ne sont pas comptabilisées.
Pour les deux exemples suivants, nous ne donnons pas explicitement de grammaires d'objets isomorphes, mais seulement les suites de substitutions à effectuer pour les construire.

Exemple 3.8 : Considérons le cas des chemins de Motzkin $(1 + x + x^2)$ et des chemins de Dyck $(1 + x^2)$. Les suites de substitutions à réaliser pour obtenir des grammaires d'objets isomophes sont un peu plus compliquées que lorsque l'on s'intéresse aux chemins de Dyck qui peuvent être vides (voir Figure 37). Il faut tout d'abord ramener le polynôme $P_D = 1 + x^2$ à un polynôme de coefficients tous non nuls en utilisant le Lemme 3.19 :

1. $S_2^{P_D}(P_D) = 1 + x + x^3 = W$,
2. $S_2^W S_3^W(W) = 1 + 2^x + x^2 + x^3 + x^4 + x^5$.

Le Théorème 3.18 est alors appliqué avec $K = 1 + 2^x + x^2 + x^3 + x^4 + x^5$ et $Q = 1 + x + x^2$. La solution est alors la suivante : $(S_3^K)^{N_3} \dots (S_1^K)^{N_1}(K) = (S_6^Q)^{M_6} \dots (S_1^Q)^{M_1}(Q)$ avec

$$\begin{pmatrix} N_1 \\ N_2 \\ N_3 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} + z \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + z \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Prenant $z = 1$, nous obtenons le polynôme

$$S_3^K(K) = S_6^Q S_5^Q S_4^Q S_3^Q S_2^Q(Q) = 1 + 2x + 2x^2 + 3x^3 + 2x^4 + 2x^5 + x^6 + x^7.$$

Exemple 3.9 : Etablissons les suites de substitutions nécessaires pour construire des grammaires d'objets isomorphes pour les chemins de Dyck $(1+x^2)$ et les polyominos parallélogrammes, engendrés par la grammaire d'objets classique Figure 16 $(1 + 2x + x^2)$. Comme dans l'exemple précédent, nous ramenons le polynôme caractéristique des chemins de Dyck au polynôme $K = 1+2x+x^2+x^3+x^4+x^5$. Nous appliquons le Théorème 3.18 aux polynômes K et $Q = 1 + 2x + x^2$, et prenant $z = 1$, nous obtenons le polynôme

$$S_3^K S_2^K(K) = S_6^Q S_5^Q S_4^Q S_3^Q S_2^Q(Q) = 1 + 3x + 3x^2 + 3x^3 + 3x^4 + 3x^5 + 2x^6 + x^7.$$

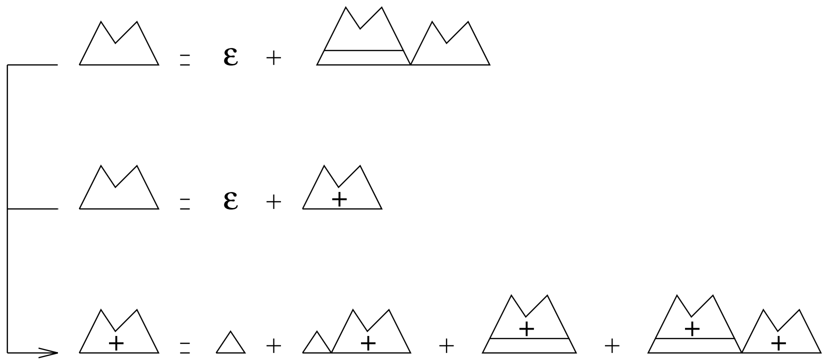


Figure 39: Une R-itération de la grammaire d'objets des chemins de Dyck.

3.4 Conclusions et perspectives

La R-itération

L'exemple précédent donne la construction d'une bijection entre les polyominos parallélogrammes et les chemins de Dyck qui aboutit à des grammaires d'objets de degré 7! Or, nous avons vu qu'une bijection très simple existe entre les polyominos parallélogrammes et les chemins de Dyck non vides (voir Figure 26 en Section 3.2.1). Pour l'obtenir, il suffit d'utiliser une autre itération de grammaires d'objets, que nous appelons la R-itération. Contrairement à la S-itération qui est une simple réécriture d'une grammaire d'objets, la R-itération ne conserve pas l'ensemble d'objets engendré par la grammaire initiale. Celle-ci consiste à retirer un objet terminal d'un des ensembles d'objets de la grammaire d'objets; on obtient ainsi une nouvelle grammaire d'objets engendrant un nouvel ensemble d'objets. Par conséquent, si l'application de suites de R-itérations à deux grammaires d'objets aboutit à des grammaires isomorphes, cela induit une bijection entre les deux ensembles d'objets engendrés par les grammaires d'objets de départ à un nombre fini près d'objets.

Prenons l'exemple de la grammaire d'objets engendrant les chemins de Dyck (grammaire (1) Figure 38). Nous appliquons la R-itération à cette grammaire d'objets en retirant l'objet terminal ϵ et en remplaçant de toutes les façons possibles les instances de l'ensemble d'objets par ce terminal. Nous obtenons alors la grammaire d'objets de la Figure 39, qui engendre donc les chemins de Dyck non vides. Et cette grammaire est isomorphe à celle des polyominos parallélogrammes, son polynôme caractéristique est $1 + 2x + x^2$. Par comparaison à la S-itération (exemple 3.9), nous obtenons des grammaires d'objets isomorphes bien plus simples, et en une seule R-itération. Certes, la bijection ne porte pas exactement sur les chemins de Dyck, mais sur les chemins de Dyck au chemin vide près. Cependant, cette bijection est plus exploitable que celle de l'exemple 3.9.

Nous avons donc l'idée d'étudier une combinaison des deux itérations de grammaires d'objets pour voir si, en acceptant des bijections à un nombre fini près d'objets, nous obtenons en général des grammaires d'objets simples.

Les grammaires d'objets de dimension ≥ 2

Après l'étude complète du cas des grammaires d'objets de dimension 1, il reste bien sûr celle beaucoup plus générale des grammaires d'objets de dimension ≥ 2. Nous n'avons pas encore exploré ce vaste domaine, mais une première analyse de quelques exemples permet de penser que les réslutats ne seront pas aussi simples et spectaculaires qu'en dimension 1.
Par exemple, les deux systèmes de polynômes suivants deviennent égaux si les variables x_1 et x_2 du second sont échangées, et les grammaires d'objets qu'ils représentent sont alors isomorphes.

$$\begin{cases} x_1, 1+{x_1}^2 \\ x_2, 1+x_1x_2 \end{cases} \qquad \begin{cases} x_1, 1+x_1x_2 \\ x_2, 1+{x_2}^2 \end{cases}$$

Ainsi, pour tester l'égalité de deux systèmes de polynômes à chaque étape d'une S-itération (ou d'une R-itération), il faut tenir compte de toutes les permutations de variables.
Une autre remarque est que l'espoir d'un théorème en dimension ≥ 2 équivalent au Théorème 3.18 s'envole avec l'exemple simple suivant :

$$\begin{cases} x_1, 1+{x_1}^2 \\ x_2, 1+x_1x_2 \end{cases} \qquad \begin{cases} x_1, 1+x_1x_2 \\ x_2, 1+x_1{x_2}^2 \end{cases}$$

Quelles que soient les S-itérations effectuées, ces deux systèmes de polynômes ne peuvent pas être S-isomorphes, car le premier polynôme du premier système, $1+{x_1}^2$, ne contiendra jamais la variable x_2 alors que les deux polynômes du second la contiendront toujours.

Les paramètres

Dans la plupart des bijections classiques en combinatoire, les paramètres préservés par la bijection sont des paramètres naturels sur les objets. Par exemple,

- la longueur, le nombre de pics, ou la somme des hauteurs des pics pour les chemins ou les mots ,
- le nombre de sommets, de feuilles, ou d'arêtes pour les arbres,
- le périmètre, le nombre de colonnes, ou l'aire pour les polyominos.

Peut-être pouvons-nous espérer que les nouvelles bijections que nous sommes en mesure de construire à l'aide de notre Théorème 3.18 vont nous donner la possibilité de mettre en avant des paramètres plus "exotiques" !? L'un des principaux obstacles à la détermination des paramètres conservés à travers ces bijections est la taille imposante des grammaires d'objets isomorphes obtenues, en nombre d'opérations d'objets et en degré. Il suffit pour s'en convaincre de regarder la Figure 37.

Une première idée est de tenter de minimiser le nombre de S-itérations successives nécessaires à l'obtention de grammaires d'objets isomorphes. Par exemple, il faudrait trouver une solution qui évite de se ramener systématiquement à des polynômes de coefficients tous non nuls, car ce Lemme 3.19 fait considérablement grossir les polynômes : pour les mots de Dyck de polynôme caractéristique $1 + x^2$, nous avons vu que le polynôme produit par le lemme est $1 + 2x + x^2 + x^3 + x^4 + x^5$.

Une autre idée est de voir si une combinaison avec la R-itération permettrait de rendre les grammaires d'objets obtenues plus simples, et ainsi plus exploitables pour la détermination des paramètres.

Il reste donc beaucoup de choses à découvrir ...

Chapitre 4

Génération aléatoire

Le travail présenté dans ce chapitre se situe dans le cadre de la *méthode récursive*, procédé de génération formalisé par Nijenhuis et Wilf [77]. Récemment, Flajolet, Zimmermann et Van Cutsem [61] ont donné une approche systématique de cette méthode pour des objets (étiquetés ou non) *décomposables*, c'est à dire que l'on peut décrire récursivement à partir d'*atomes* et à l'aide des opérations d'*union*, de *produit*, de construction d'*ensembles*, de *suites* et de *cycles*. Ils montrent qu'à partir d'une telle description, qu'ils nomment *spécification*, on peut construire automatiquement un programme de génération aléatoire uniforme des objets considérés selon le nombre d'atomes. L'implémentation des calculs avancés développés dans [61] a été réalisée dans *Gaïa*, un “package” *Maple* conçu par Zimmermann [99], puis adapté par Murray à la dernière version de *Maple* (*Maple V Release 3*) sous le nom de *CombStruct*. Concrètement, étant donnée une spécification, *CombStruct* fournit automatiquement les procédures d'énumération et les procédures de génération uniforme selon la taille (nombre d'atomes) des structures. Générer une structure de taille n coûte $O(n \log n)$ opérations arithmétiques dans le pire des cas, après un calcul préalable des coefficients d'énumération en $O(n^2)$.

Le champ d'application des grammaires d'objets est à la fois plus restreint et plus large que celui des spécifications. Les grammaires d'objets ne comprennent pas les opérations de construction d'ensembles, de suites ou de cycles, ce ne sont pas des opérations d'objets. Elle utilisent uniquement des opérations d'*union* et de *produit*, mais ce sont des *produits “étendus”*. En effet, une unique notion de taille est associée aux spécifications, le nombre d'atomes, et lorsque un non-terminal est défini par le produit de deux non-terminaux, sa fonction génératrice est égale au produit des fonctions génératrices des deux non-terminaux. Or, pour les grammaires d'objets, nous pouvons associer à une grammaire diverses notions de tailles définies par différentes valuations d'objets ; de plus, ces valuations peuvent être q-linéaires. Si $E = \phi(E_1, E_2)$, la valeur de la fonction génératrice de E en fonction de celles de E_1 et E_2 dépend des valuations d'objets considérées; pour des valuations d'objets q-linéaires quelconques, elle est de la forme

$$V_E(E) = \lambda_\phi V_{E_1}(E_1)\Big|_{x \leftarrow xq^{A_1^\phi}} V_{E_2}(E_2)\Big|_{x \leftarrow xq^{A_2^\phi}} .$$

Ainsi, nous pouvons atteindre, pour divers ensembles d'objets, des paramètres non algébriques qui n'ont pas encore été exploités dans le domaine de la génération aléatoire.

Nous développons dans ce chapitre une étude inspirée de celle de Flajolet, Zimmermann et Van Cutsem [61] sur les spécifications; nous l'adaptons aux grammaires d'objets auxquelles sont associées des valuations d'objets linéaires et q-linéaires. Pour garantir une génération aléatoire uniforme des objets, nous nous intéressons naturellement ici à des grammaires d'objets non ambiguës. Une grammaire d'objets ambiguë favoriserait le tirage des objets ayant plusieurs arbres de dérivation dans la grammaire. La forme utilisée pour décrire les grammaires d'objets est la forme 1-2. La première section fournit les transformations à effectuer sur l'ensemble de valuations d'objets associé à une grammaire d'objets en parallèle à la mise en forme 1-2 de la grammaire. Dans la deuxième section, nous établissons les critères qui permettent de tester si un couple (grammaire d'objets,ensemble de valuations d'objets) est *bien fondé* ou non. Enfin, les troisième et quatrième sections présentent les procédures d'énumération et de génération, ainsi que leurs complexités : le calcul du nombre d'objets ayant comme valeur de valuation ${x_1}^{k_1} \dots {x_n}^{k_n} {q_1}^{l_1} \dots {q_r}^{l_r}$ est dans le cas le pire en $O(k_1^2 \dots k_n^2\ l_1^2 \dots l_r^2)$, et générer un objet ayant cette même valeur de valuation coûte $O(k_1 \dots k_n\ l_1 \dots l_r(k_1 + \dots + k_n + l_1 + \dots + l_r))$ opérations. Nous avons concrétisé ce travail en réalisant le programme *Maple qAlGO*, pour lequel la technique d'implémentation reste naturellement proche de celle de *CombStruct*. A partir d'une grammaire d'objets non ambiguë G et d'un ensemble de valuations d'objets G-linéaire ou G-q-linéaire, *qAlGO* construit automatiquement, pour tous les ensembles d'objets de G, les procédures d'énumération et les procédures de génération uniforme selon les valuations d'objets. Le caractère automatique du logiciel *qAlGO* en fait un outil très utile qui facilite l'étude expérimentale de diverses statistiques sur des objets combinatoires (voir des exemples en Section 4.6 et Annexe A). *qAlGO* vient en complément de *CombStruct*. En effet, *CombStruct* permet le tirage aléatoire d'objets construits à l'aide d'*unions*, de *produits*, de constructions de *cycles*, d'*ensembles*, ou de *suites*, selon le paramètre *nombre d'atomes*. De ce point de vue, *qAlGO* traite uniquement les unions et les produits (opérations d'objets), mais il accepte diverses valuations d'objets pour les mêmes grammaires d'objets, et surtout des valuations q-linéaires, qui rendent ainsi accessible la génération aléatoire d'objets combinatoires selon des paramètres non algébriques : par exemple l'*aire* pour les différentes classes de polyominos (voir Section 4.6).

4.1 Grammaires d'objets en forme 1-2

La forme 1-2 des grammaires d'objets est à la base de la méthode de génération aléatoire présentée dans ce chapitre. Nous avons vu au Chapitre 1 comment transformer une grammaire d'objets quelconque en une grammaire d'objets en forme 1-2 équivalente. La première partie de cette section décrit la manière dont on peut modifier un ensemble de valuations d'objets associé à une grammaire d'objets pour en obtenir un *équivalent* associé à la grammaire en forme 1-2; *équivalent* signifie que ce nouvel ensemble de valuations d'objets conserve la valeur des valuations des objets engendrés par les ensembles d'objets de la grammaire initiale, et ainsi les fonctions génératrices sont les mêmes. La seconde partie présente une notation particulière des grammaires d'objets sur laquelle s'appuient les techniques de preuve mises en œuvre dans le reste du chapitre.

4.1.1 Forme 1-2 et valuations d'objets

Nous considérons de nouveau les mêmes notations qu'au Chapitre 2 : X et Q désignent deux ensembles disjoints de variables, $X = \{x_1, \ldots, x_n\}$ et $Q = \{q_1, \ldots, q_r\}$, et $K[X, Q]$ est l'anneau de polynômes à variables dans X et Q et à coefficients dans K.

Proposition 4.1 *Soient $G = < \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets et ν un ensemble de valuations d'objets G-q-linéaire associé à G : $\nu = \{V_E : E \longrightarrow K[X, Q], E \in \mathcal{E}\}$. Il existe un ensemble de valuations d'objets G-q-linéaire équivalent associé la forme 1-2 de G, que l'on peut construire à partir de ν.*

PREUVE :
On rappelle que pour transformer G en forme 1-2, on découpe les domaines de définition de toutes les opérations d'objets d'arité > 2 en rajoutant des ensembles d'objets et des opérations d'objets identité (voir la preuve de la Proposition 1.17). Pour ϕ d'arité > 2 telle que $\phi : E_1 \times \ldots \times E_k \longrightarrow E$, on transforme le système d'équations de G :

$$\left\{ \begin{array}{lcl} \vdots & & \\ E & = & \cdots + \phi(E_1, \ldots, E_k) + \cdots \\ \vdots & & \end{array} \right. \quad \text{devient} \quad \left\{ \begin{array}{lcl} \vdots & & \\ E & = & \cdots + \phi_{E_1}(E_1, F_{E_2}) + \cdots \\ \vdots & & \\ F_{E_2} & = & \phi_{E_2}(E_2, F_{E_3}) \\ & \vdots & \\ F_{E_{k-1}} & = & \phi_{E_{k-1}}(E_{k-1}, E_k) \end{array} \right.$$

Ainsi, il faut définir de nouvelles valuations d'objets pour ces nouveaux ensembles d'objets et opérations d'objets. Si on a

$$V_E(\phi(E_1,\ldots,E_k)) = \lambda_\phi \prod_{i=1}^{k} V_{E_i}(E_i)\big|_{x \leftarrow xq^{A_i^\phi}},$$

alors on définit :

$$\begin{cases} V_{F_{E_i}}(F_{E_i}) = V_{F_{E_i}}(\phi_{E_i}(E_i, F_{E_{i+1}})) = V_{E_i}(E_i)\big|_{x \leftarrow xq^{A_i^\phi}} V_{F_{E_{i+1}}}(F_{E_{i+1}}) \quad , \ i = 2\ldots k-2 \\ V_{F_{E_{k-1}}}(F_{E_{k-1}}) = V_{F_{E_{k-1}}}(\phi_{E_{k-1}}(E_{k-1}, E_k)) = V_{E_{k-1}}(E_{k-1})\Big|_{x \leftarrow xq^{A_{k-1}^\phi}} V_{E_k}(E_k)\big|_{x \leftarrow xq^{A_k^\phi}} \\ V_E(\phi_{E_1}(E_1, F_{E_2})) = \lambda_\phi \ V_{E_1}(E_1)\big|_{x \leftarrow xq^{A_1^\phi}} V_{F_{E_2}}(F_{E_2}) \end{cases}$$

soit

- $\lambda_{\phi_{E_1}} = \lambda_\phi$, et $\lambda_{\phi_{E_i}} = 1$ pour $i = 2\ldots k-1$,
- $A_{E_{k-1}}^{\phi_{E_{k-1}}} = A_{k-1}^\phi$, $A_{E_k}^{\phi_{E_{k-1}}} = A_k^\phi$ et, $A_{E_i}^{\phi_{E_i}} = A_i^\phi$ et $A_{F_{E_{i+1}}}^{\phi_{E_i}} = 0$ pour $i = 1\ldots k-2$.

□

4.1.2 Notation en forme développée

Définition 4.2 *Une grammaire d'objets est en forme développée si tout ensemble d'objets E est défini par une équation du type*

(1) soit $E = e$,

(2) soit $E = E_1 + \cdots + E_m$,

(3) soit $E = \phi(E_{i_{1_\phi}},\ldots,E_{i_{k_\phi}})$.

Nous avons alors la propriété suivante.

Propriété 4.3 *Toute grammaire d'objets peut être transformée en une grammaire équivalente en forme développée.*

En effet, soit $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets de dimension n :

$$\left\{ E_i = \sum_{e_i \in T_{E_i}} e_i \ + \sum_{dest(\phi)=E_i} \phi(E_{i_{1_\phi}},\ldots,E_{i_{k_\phi}}) \right\}_{i=1,\ldots,n}.$$

La notation en forme développée est une simple réécriture de la grammaire d'objets qui consiste à isoler les objets terminaux et les opérations d'objets de la manière suivante : on remplace l'équation de chaque ensemble d'objets E_i par l'ensemble d'équations ci-dessous.

$$\left\{\begin{array}{lcl} & \vdots & \\ F_{e_i} & = & e_i \\ & \vdots & \end{array}\right\}\text{pour } e_i \in T_{E_i}$$

$$\left\{\begin{array}{lcl} & \vdots & \\ F_{\phi} & = & \phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}}) \\ & \vdots & \end{array}\right\}\text{pour } \phi \text{ telle que } dest(\phi) = E_i$$

$$E_i = \sum_{e_i \in T_{E_i}} F_{e_i} + \sum_{dest(\phi)=E_i} F_{\phi}$$

Remarque : Dans la dernière équation, celle qui définit E_i, les opérations d'objets appliquées aux nouveaux ensembles d'objets F_{e_i} et F_{ϕ} sont des opérations identité. Pour ne pas alourdir les notations, on se permet de ne pas les écrire explicitement.

On peut aussi redécouper l'équation de type (2) pour avoir seulement des opérations + binaires :

$$E = E_1 + F_{E_2}, \ldots, F_{E_{m-1}} = E_{m-1} + E_m.$$

Alors, pour une grammaire d'objets en forme 1-2 développée, l'équation définissant un ensemble d'objets E est du type

(1) soit $E = e$,

(2) soit $E = E_1 + E_2$,

(3) soit $E = \phi(E_1)$,

(4) soit $E = \phi(E_1, E_2)$.

Exemple 4.1 : Reprenons l'exemple des polyominos parallélogrammes E_{pp} engendrés par la grammaire d'objets non ambiguë (cf Chapitre 1)

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

et écrivons la sous forme 1-2 développée :

$$\left\{\begin{array}{lclclcl} E_{pp} & = & E_1 + E_2 & \quad & E_1 & = & \square \\ E_2 & = & E_3 + E_4 & & E_3 & = & \phi_1(E_{pp}) \\ E_4 & = & E_5 + E_6 & & E_5 & = & \phi_2(E_{pp}) \\ E_6 & = & \phi_3(E_{pp}, E_{pp}) & & & & \end{array}\right.$$

Dans la suite du chapitre, nous considérons les grammaires d'objets écrites sous cette forme.

4.2 Couple (G, ν) bien fondé

Nous adaptons ici aux grammaires d'objets le caractère *bien fondé* des spécifications défini par Zimmermann [98]. Toutes les grammaires d'objets G et tous les ensembles d'applications associés ν que l'on peut définir ne se prêtent pas à la génération aléatoire. Certains couples (G, ν) sont *dégénérés*, les autres sont dits *bien fondés*. Les cas dégénérés que nous voulons éviter sont des ensembles d'objets qui n'engendrent aucun objet (non productifs), ou encore des ensembles d'objets qui engendrent une infinité d'objets ayant la même valuation, et qui font que par conséquent les fonctions génératrices ne sont pas définies. Nous établissons alors la définition suivante.

Définition 4.4 *Soient $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets non ambiguë et ν un ensemble d'applications : $\nu = \{V_E : E \longrightarrow K[X, Q], E \in \mathcal{E}\}$. Le couple (G, ν) est dit <u>bien fondé</u> si pour chaque ensemble d'objets $E \in \mathcal{E}$:*

(1) E est productif (Définition 1.10),

(2) V_E est une valuation d'objets (Définition 2.1).

Dans toute la suite du chapitre, notre étude est restreinte aux cas où les applications V_E ont pour valeurs des polynômes réduits à un monôme. C'est en fait la situation la plus fréquemment rencontrée pour les objets et les paramètres "usuels" (mots, chemins, arbres, ...). Il serait cependant intéressant de traiter le cas général pour justement tenter d'étudier des paramètres moins courants.
Notations : $Mon[X, Q]$ désigne l'ensemble des polynômes de $K[X, Q]$ réduits à un monôme.

Théorème 4.5 *Soient G une grammaire d'objets non ambiguë et ν un ensemble G-q-linéaire d'applications à valeurs dans $Mon[X, Q]$. Le caractère bien fondé du couple (G, ν) est décidable.*

La suite de cette section est consacrée à la preuve du théorème ci-dessus.

Pour vérifier la condition 1 de la Définition 4.4 (chaque ensemble d'objets est productif), il suffit d'utiliser l'algorithme exhibé au Chapitre 1 (voir la preuve du Lemme 1.14 page 23). Le cas de la condition 2 (chaque V_E est une valuation d'objets) est traité dans la section suivante.

4.2.1 ν est un ensemble de valuations d'objets

Il s'agit de vérifier que, pour tout ensemble d'objets E de la grammaire, le nombre d'objets de E ayant la même valuation est fini pour chaque valeur de valuation, c'est à dire :

$$\forall (k_1, \ldots, k_n, l_1, \ldots, l_r) \in \mathbb{N}^n \times \mathbb{N}^r \ ,$$
$$\{e \in E / V_E(e) = {x_1}^{k_1} \ldots {x_n}^{k_n} {q_1}^{l_1} \ldots {q_r}^{l_r}\} \text{ est un ensemble fini.}$$

Pour cela, dans le même esprit que Greene [64] et Zimmermann [98], nous définissons une relation d'ordre partiel sur les ensembles d'objets.

Définition 4.6 *Soient G une grammaire d'objets et ν un ensemble G-q-linéaire d'applications à valeurs dans $Mon[X, Q]$. On définit une relation d'ordre partiel entre les ensembles d'objets de G : on a $E_1 \prec E$ lorsque*

- *soit $E = E_1 + E_2$,*
- *soit $E = \phi(E_1)$ avec $\lambda_\phi = 1$ et $A_1^\phi = 0$,*
- *soit $E = \phi(E_1, E_2)$ avec $\lambda_\phi = 1$, $A_1^\phi = 0$, et il existe $e \in \mathcal{O}_G(E_2)$ tel que $V_{E_2}(e) = 1$.*

Remarque : Pour déterminer si un ensemble d'objets engendre un objet de valuation 1, on utilise un algorithme très similaire à celui mis en œuvre pour la preuve du Lemme 1.14 (décidabilité de "E est productif"). On fait la construction suivante :

- $\mathcal{E}_0 = \{E \in \mathcal{E} \ / \ \exists e \in T_E, V_E(e) = 1\}$,
- $\mathcal{E}_i = \mathcal{E}_{i-1} \bigcup \{E \in \mathcal{E} \ / \ \exists \phi \in \mathcal{P}$ avec $dest(\phi) = E$, chaque composante de $dom(\phi) \in \mathcal{E}_{i-1}$ et $\lambda_\phi = 1\}$, pour $i > 0$,
- $\mathcal{E}_{total} = \bigcup_{i \geq 0} \mathcal{E}_i$.

$\mathcal{E}_{total}$ est alors la famille d'ensembles d'objets qui engendrent un objet de valuation 1, et il est calculable en un nombre borné d'opérations.

La relation $E_1 \prec E$ traduit le fait qu'au moins un objet de E de valuation donnée est obtenu à partir d'un objet de E_1 de même valuation. Ainsi, il y a dégénérescence lorsque l'ordre partiel $\prec$ contient un cycle.

Proposition 4.7 *Soient G une grammaire d'objets dont tous les ensembles d'objets sont productifs et ν un ensemble G-q-linéaire d'applications à valeurs dans $Mon[X, Q]$. Alors les deux assertions suivantes sont équivalentes :*

(i) il existe un ensemble d'objets engendrant une infinité d'objets de même valuation,

(ii) il existe un cycle dans l'ordre partiel $\prec$.

PREUVE :

On considère que la grammaire d'objets est sous forme 1-2 développée. Il est évident que lorsque (ii) est vraie, (i) l'est aussi : le cycle génère un nombre infini d'arbres de dérivation pour une valuation fixée.

$(i) \Longrightarrow (ii)$: supposons que (i) est vraie. Soit E un ensemble d'objets engendrant un ensemble infini d'objets de valuation $x^{\boldsymbol{k}}q^{\boldsymbol{l}} = x_1^{k_1}\dots x_n^{k_n}q_1^{l_1}\dots q_r^{l_r}$ (où $\boldsymbol{k} = (k_1,\dots,k_n)$ et $\boldsymbol{l} = (l_1,\dots,l_r)$) : on note $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E) = \infty$. On peut choisir $\boldsymbol{k}$ et $\boldsymbol{l}$ "minimums" pour lesquels il existe un tel ensemble d'objets E : on prend le plus petit $k_1 = k_1^0$ tel qu'il existe F avec $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(F) = \infty$, puis parmi ces $(k_1^0, k_2, \dots, k_n, l_1, \dots, l_r)$, on prend le plus petit $k_2 = k_2^0$ tel qu'il existe F avec $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(F) = \infty$, et ainsi de suite jusqu'à l_r. Les objets de l'ensemble d'objets E sont obtenus par

soit $E = E_1 + E_2$ **:** alors on a forcément $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E_1) = \infty$ ou $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E_2) = \infty$,

soit $E = \phi(E1)$ **:** $\boldsymbol{k}$ et $\boldsymbol{l}$ sont "minimums", alors on a forcément $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E_1) = \infty$ (et donc $\lambda_\phi = 1$, et $A_1^\phi = 0$ si $\boldsymbol{k} \neq \mathbf{o}$),

soit $E = \phi(E1, E_2)$ **:** $\boldsymbol{k}$ et $\boldsymbol{l}$ sont "minimums", alors on a forcément
soit $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E_1) = \infty$ et il existe $e \in \mathcal{O}_G(E_2)$ tel que $V_{E_2}(e) = 1$, (et donc $\lambda_\phi = 1$, et $A_1^\phi = 0$ si $\boldsymbol{k} \neq \mathbf{o}$),
soit $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E_2) = \infty$ et il existe $e \in \mathcal{O}_G(E_1)$ tel que $V_{E_1}(e) = 1$ (et donc $\lambda_\phi = 1$, et $A_2^\phi = 0$ si $\boldsymbol{k} \neq \mathbf{o}$).

Nous venons ainsi de montrer que si $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(E) = \infty$, alors il existe F tel que $\mathcal{O}_G^{(\boldsymbol{k},\boldsymbol{l})}(F) = \infty$ et $F \prec E$. Le nombre d'ensembles d'objets de G étant fini, l'ordre partiel $\prec$ admet nécessairement un cycle.

$\square$

Exemple 4.2 : Reprenons l'exemple précédent de la grammaire d'objets en forme 1-2 développée des polyominos parallélogrammes E_{pp}. Considérons l'application "largeur", pour tout E dans $\{E_{pp}, E_1, \dots, E_6\}$,

$$\begin{array}{rccl} V_E : & E & \longrightarrow & \mathbb{N}[x] \\ & e & \longmapsto & x^{largeur(e)} \end{array}$$

Le graphe de dépendance de la relation d'ordre partiel $\prec$ construit Figure 40 admet un cycle ($E_{pp} \prec E_5 \prec E_4 \prec E_2 \prec E_{pp}$), donc les applications V_E ne sont pas des valuations d'objets. En effet, l'opération d'objets ϕ_2 ne modifie pas la largeur du polyomino passé en paramètre; alors, nous avons des ensembles infinis d'objets de même largeur, par exemple $\{\square, \phi_2(\square), \phi_2(\phi_2(\square)), \dots\}$ qui sont tous de largeur 1 et ont donc tous la même valuation x.

Si à la place de l'application "largeur", nous considérons l'application "largeur et aire",

$$\begin{array}{rccl} V'_E : & E & \longrightarrow & \mathbb{N}[x,q] \\ & e & \longmapsto & x^{largeur(e)}\, q^{aire(e)} \end{array}$$

nous obtenons alors un graphe de dépendance de $\prec$ sans cycle (l'arête (E_{pp}, E_5) disparaît), impliquant que les applications V'_E sont des valuations d'objets.

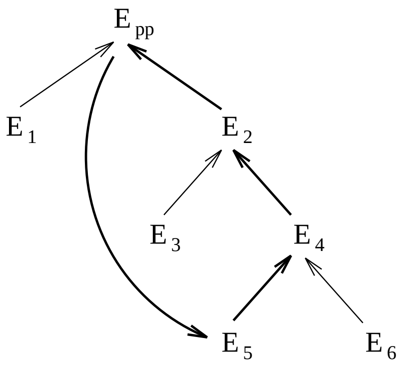


Figure 40: Graphe de dépendance de $\prec$ pour les polyominos parallélogrammes et l'application "largeur".

4.2.2 Le caractère bien fondé de (G, ν)

Nous pouvons maintenant prouver le Théorème 4.5 en donnant un algorithme de décision du caractère bien fondé d'un couple (G, ν).

Algorithme BienFondé :
Entrée : un couple (G, ν).
Sortie : **vrai** si bien fondé, **faux** sinon.

$\star$ Calculer l'ensemble des ensembles d'objets productifs (Lemme 1.14)
Si un des ensembles d'objets n'est pas productif **Alors** Retourner **faux**.
$\star$ Calculer le graphe de dépendance de la relation $\prec$.
Si ce graphe admet un cycle **Alors** Retourner **faux**.
$\star$ Retourner **vrai**.

4.3 Procédures d'énumération

Pour pouvoir générer des objets aléatoirement de manière uniforme selon une certaine valuation d'objets, il est nécessaire de savoir les dénombrer selon cette valuation.

Nous présentons dans cette section un algorithme de calcul pour les coefficients des fonctions génératrices des objets. Il utilise les récurrences obtenues simplement à partir de la grammaire d'objets. Nous montrons alors que le calcul du nombre d'objets de valuation $x_1^{k_1} \dots x_n^{k_n} q_1^{l_1} \dots q_r^{l_r}$ est dans le pire des cas en $O(k_1^2 \dots k_n^2\, l_1^2 \dots l_r^2)$ opérations.

4.3.1 Quelques notations

Fixons dès maintenant quelques notations pour faciliter l'écriture des différents algorithmes qui suivent dans ce chapitre.

Si P est un un monôme de $Mon[X,Q]$, soit $P = x^{\boldsymbol{k}} q^{\boldsymbol{l}} = x_1^{k_1} \dots x_n^{k_n} q_1^{l_1} \dots q_r^{l_r}$, alors on note P_x le n-uplet $\boldsymbol{k} = (k_1, \dots, k_n)$ et P_q le r-uplet $\boldsymbol{l} = (l_1, \dots, l_r)$.

On peut considérer que le n-uplet $\boldsymbol{k}$ (resp. r-uplet $\boldsymbol{l}$) est un vecteur de taille n (resp. r). Ainsi, pour deux vecteurs de même taille u et v, on note $u+v$ (resp. $u-v$) le vecteur obtenu en effectuant la somme (resp. différence) composante à composante des vecteurs u et v.

Pour $v = (v_i)$ un vecteur et $A = (a_{ij})$ une matrice, $1 \leq i \leq n$, $1 \leq j \leq r$, on note $v.A$ le vecteur $w = (w_j)$, $1 \leq j \leq r$, obtenu en effectuant le produit de v par A : $w_j = v_1 a_{1j} + \dots + v_n a_{nj}$ pour $1 \leq j \leq r$.

4.3.2 Algorithme de calcul des coefficients

Soient une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ et un ensemble de valuations d'objets G-q-linéaire associé à G, $\nu = \{V_E : E \longrightarrow Mon[X,Q], E \in \mathcal{E}\}$, de telle sorte que le couple (G, ν) soit bien fondé.

La fonction génératrice d'un ensemble d'objets E est notée

$$V_E(E) = \sum_{K_1,\dots,K_n,L_1,\dots,L_r=0}^{\infty} c_E(K_1,\dots,K_n,L_1,\dots,L_r)\, x_1^{K_1} \dots x_n^{K_n} q_1^{L_1} \dots q_r^{L_r},$$

ou plus simplement

$$V_E(E) = \sum_{\boldsymbol{K},\boldsymbol{L}=0}^{\infty} c_E(\boldsymbol{K},\boldsymbol{L})\, x^{\boldsymbol{K}} q^{\boldsymbol{L}},$$

où $\boldsymbol{K}$ est le n-uplet $(K_1,\dots,K_n)$ et $\boldsymbol{L}$ le r-uplet $(L_1,\dots,L_r)$.

Nous considérons que la grammaire d'objets est donnée sous forme 1-2 développée. Alors, les formules pour calculer les coefficients $c_E(\boldsymbol{K},\boldsymbol{L})$ pour un ensemble d'objets E sont les suivantes :

- soit $E = e$, alors

$$\text{si } V_E(e) = x^{\boldsymbol{K}} q^{\boldsymbol{L}} \text{ alors } c_E(\boldsymbol{K},\boldsymbol{L}) = 1 \text{ sinon } 0,$$

- soit $E = E_1 + E_2$, alors

$$c_E(\boldsymbol{K},\boldsymbol{L}) = c_{E_1}(\boldsymbol{K},\boldsymbol{L}) + c_{E_2}(\boldsymbol{K},\boldsymbol{L}),$$

- soit $E = \phi(E_1)$ avec $V_E(\phi(E_1)) = \lambda_\phi V_{E_1}(E_1)\big|_{x \leftarrow x q^{A_1^\phi}}$, alors

$$c_E(\boldsymbol{K},\boldsymbol{L}) = c_{E_1}(\boldsymbol{K} - \lambda_{\phi x}, \boldsymbol{L} - \lambda_{\phi q} - (\boldsymbol{K} - \lambda_{\phi x}).A_1^\phi),$$

- soit $E = \phi(E_1, E_2)$ avec $V_E(\phi(E_1,E_2)) = \lambda_\phi V_{E_1}(E_1)\big|_{x \leftarrow xq^{A_1^\phi}} V_{E_2}(E_2)\big|_{x \leftarrow xq^{A_2^\phi}}$, alors

$$c_E(\boldsymbol{K},\boldsymbol{L}) = \sum_{\boldsymbol{i}=\mathbf{o}}^{\boldsymbol{K}} \sum_{\boldsymbol{j}=\mathbf{o}}^{\boldsymbol{L}} c_{E_1}(\boldsymbol{i},\boldsymbol{j}).c_{E_2}(\boldsymbol{K}-\lambda_{\phi x}-\boldsymbol{i}, \boldsymbol{L}-\lambda_{\phi q}-\boldsymbol{i}.A_1^\phi-(\boldsymbol{K}-\lambda_{\phi x}-\boldsymbol{i}).A_2^\phi-\boldsymbol{j}).$$

L'algorithme présenté ici permet de calculer les coefficients des fonctions génératrices des ensembles d'objets de la grammaire jusqu'à l'ordre $(\boldsymbol{k},\boldsymbol{l}) = (k_1,\ldots,k_n,l_1,\ldots,l_r)$.

Algorithme CalcCoeffs :

Entrée : un couple (G,ν) bien fondé (G en forme 1-2),
et un $(n+r)$-uplet d'entiers $(\boldsymbol{k},\boldsymbol{l})$.

Sortie : pour chaque ensemble d'objets E, les coefficients $c_E(\boldsymbol{K},\boldsymbol{L})$,
pour $\mathbf{o} \leq \boldsymbol{K} \leq \boldsymbol{k}$ et $\mathbf{o} \leq \boldsymbol{L} \leq \boldsymbol{l}$.

$\mathcal{L} \longleftarrow$ liste des ensembles d'objets triés selon l'ordre partiel $\prec$
Pour $K_1 = 0$ **jusqu'à** k_1 **Faire**
  $\ddots$
    Pour $K_n = 0$ **jusqu'à** k_n **Faire**
      Pour $L_1 = 0$ **jusqu'à** l_1 **Faire**
        $\ddots$
          Pour $L_r = 0$ **jusqu'à** l_r **Faire**
            $E_0 \longleftarrow \mathcal{L}$
            Tant que $E_0 \neq \{\}$ **Faire**
              $E \longleftarrow$ premier(E_0)
              Calculer $c_E(K_1,\ldots,K_n,L_1,\ldots,L_r)$ (formules ci-dessus)
              $E_0 \longleftarrow E_0 - \{E\}$
            Fin Tant que
          Fin Pour
        $\iddots$
      Fin Pour
    Fin Pour
  $\iddots$
Fin Pour

4.3.3 Complexité

Théorème 4.8 *La complexité de l'algorithme* **CalcCoeffs** *est de* $O(|G|\ k_1^2 \dots k_n^2\ l_1^2 \dots l_r^2)$ *opérations arithmétiques sur les coefficients (additions, soustractions, multiplications, divisions), où* $|G|$ *est la dimension de la grammaire d'objets en forme 1-2 développée.*

PREUVE :

Pour chaque ensemble d'objets E, déterminons le coût du calcul de $c_E(\boldsymbol{K}, \boldsymbol{L})$. Dans les trois premiers cas ($E = e$, $E = E_1 + E_2$, $E = \phi(E_1)$), le coût est de $O(1)$. Pour le dernier cas ($E = \phi(E_1, E_2)$), le coût est de $O(K_1 \dots K_n L_1 \dots L_r)$ opérations. Ainsi, pour chaque ensemble d'objets E, le calcul de $c_E(\boldsymbol{K}, \boldsymbol{L})$ nécessite au plus $O(K_1 \dots K_n L_1 \dots L_r)$ opérations. Donc, le coût de la boucle la plus interne calculant ce coefficient pour tous les ensembles d'objets est $O(|G| K_1 \dots K_n L_1 \dots L_r)$. Le coût du calcul de tous les coefficients jusqu'à l'ordre $(k_1, \dots, k_n, l_1, \dots, l_r)$ est alors $O(|G|\ k_1^2 \dots k_n^2\ l_1^2 \dots l_r^2)$.

□

4.4 Procédures de génération

Nous exhibons ici des procédures de génération aléatoire pour les ensembles d'objets de grammaires d'objets en forme 1-2. La structure de la procédure de génération associée à un ensemble d'objets dépend du type d'équation le définissant dans la grammaire. Grâce à la section précédente, les tables de coefficients d'énumération des ensembles d'objets de la grammaire sont calculées à l'avance, et une fois pour toutes, jusqu'au rang $(\boldsymbol{k}, \boldsymbol{l})$, en un temps $O(k_1^2 \dots k_n^2\ l_1^2 \dots l_r^2)$. La complexité des procédures de génération est alors en $O(k_1 \dots k_n\ l_1 \dots l_r(k_1 + \dots + k_n + l_1 + \dots + l_r))$.

4.4.1 Procédures de base

Soient une grammaire d'objets $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ et un ensemble de valuations d'objets G-q-linéaire associé à G, $\nu = \{V_E : E \longrightarrow Mon[X, Q], E \in \mathcal{E}\}$, de telle sorte que le couple (G, ν) soit bien fondé, et G est sous forme 1-2 développée.

Nous associons à chaque ensemble d'objets E de la grammaire une procédure g_E qui prend comme paramètres un n-uplet $\boldsymbol{k}$ et un r-uplet $\boldsymbol{l}$, et génére aléatoirement et uniformément un objet de E de valuation $x^{\boldsymbol{k}} q^{\boldsymbol{l}}$. Elle donne en fait l'arbre de dérivation de l'objet dans la grammaire. Cette procédure g_E dépend de la forme de l'équation qui définit E dans la grammaire d'objets.

- Soit $E = e$. La procédure g_E est alors triviale :

g_E :=**Proc**($\boldsymbol{k}$,$\boldsymbol{l}$)

Si $V_E(e) = x^{\boldsymbol{k}} q^{\boldsymbol{l}}$ **Alors** Retourner e

Fin Proc

- Soit $E = E_1 + E_2$. La procédure g_E doit générer un objet de E_1 ou de E_2. La probabilité pour que cet objet appartienne à E_1 est égal à $c_{E_1}(\boldsymbol{k},\boldsymbol{l})/c_E(\boldsymbol{k},\boldsymbol{l})$:

g_E :=**Proc**($\boldsymbol{k}$,$\boldsymbol{l}$)

U := Uniforme([0, 1]);

Si $U < c_{E_1}(\boldsymbol{k},\boldsymbol{l})/c_E(\boldsymbol{k},\boldsymbol{l})$ **Alors** Retourner $g_{E_1}(\boldsymbol{k},\boldsymbol{l})$

Sinon Retourner $g_{E_2}(\boldsymbol{k},\boldsymbol{l})$

Fin Proc

- Soit $E = \phi(E_1)$, avec $V_E(\phi(E_1)) = \lambda_\phi V_{E_1}(E_1)|_{x \leftarrow xq^{A_1^\phi}}$. La procédure g_E est alors très simple. Elle retourne un objet de E_1 ayant pour valuation $x^{\boldsymbol{k}-\lambda_{\phi x}} q^{\boldsymbol{l}-\lambda_{\phi q}-(\boldsymbol{k}-\lambda_{\phi x}).A_1^\phi}$:

g_E :=**Proc**($\boldsymbol{k}$,$\boldsymbol{l}$)

$\boldsymbol{kk} := \boldsymbol{k} - \lambda_{\phi x}$; $\boldsymbol{ll} := \boldsymbol{l} - \lambda_{\phi q}$;

Retourner $\phi(g_{E_1}(\boldsymbol{kk}, \boldsymbol{ll} - \boldsymbol{kk}.A_1^\phi))$

Fin Proc

- Soit $E = \phi(E_1, E_2)$, avec $V_E(\phi(E_1,E_2)) = \lambda_\phi V_{E_1}(E_1)|_{x \leftarrow xq^{A_1^\phi}} V_{E_2}(E_2)|_{x \leftarrow xq^{A_2^\phi}}$. La procédure g_E doit générer un objet de E obtenu par ϕ à partir d'un objet de E_1 et d'un objet de E_2 qui respectent la définition des valuations. La probabilité pour que l'objet de E_1 soit de valuation $x^{\boldsymbol{K}} q^{\boldsymbol{L}}$ et celui de E_2 de valuation $x^{\boldsymbol{k}-\lambda_{\phi x}-\boldsymbol{K}} q^{\boldsymbol{l}-\lambda_{\phi q}-\boldsymbol{K}.A_1^\phi-(\boldsymbol{k}-\lambda_{\phi x}-\boldsymbol{K}).A_2^\phi-\boldsymbol{L}}$ est $c_{E_1}(\boldsymbol{K},\boldsymbol{L}).c_{E_2}(\boldsymbol{k}-\lambda_{\phi x}-\boldsymbol{K}, \boldsymbol{l}-\lambda_{\phi q}-\boldsymbol{K}.A_1^\phi-(\boldsymbol{k}-\lambda_{\phi x}-\boldsymbol{K}).A_2^\phi-\boldsymbol{L})/c_E(\boldsymbol{k},\boldsymbol{l})$. D'où la procédure :

g_E :=**Proc**($\boldsymbol{k}$,$\boldsymbol{l}$)

$\boldsymbol{kk} := \boldsymbol{k} - \lambda_{\phi x}$; $\boldsymbol{ll} := \boldsymbol{l} - \lambda_{\phi q}$;

$\boldsymbol{K} := \mathbf{o}$; $\boldsymbol{L} := \mathbf{o}$; $\boldsymbol{T} := \boldsymbol{ll} - \boldsymbol{kk}.A_2^\phi$;

$S := c_{E_1}(\mathbf{o},\mathbf{o}).c_{E_2}(\boldsymbol{kk},\boldsymbol{T})/c_E(\boldsymbol{k},\boldsymbol{l})$;

U := Uniforme([0, 1]);

Tant que $U > S$ **Faire**

$K_1 := K_1 + 1$;

$K_2 := 0$;

Tant que $U > S$ **Et** $K_2 \leq k_2$ **Faire**

$\ddots$

$K_n := 0;$
Tant que $U > S$ **Et** $K_n \leq k_n$ **Faire**
$K_n := K_n + 1;$
$\boldsymbol{T} := \boldsymbol{ll} - \boldsymbol{K}.A_1^\phi - (\boldsymbol{kk} - \boldsymbol{K}).A_2^\phi;$
$L_1 := 0;$
Tant que $U > S$ **Et** $L_1 \leq l_1$ **Faire**
$\ddots$
$L_r := 0;$
Tant que $U > S$ **Et** $L_r \leq l_r$ **Faire**
$L_r := L_r + 1;$
$S := S + c_{E_1}(\boldsymbol{K}, \boldsymbol{L}).c_{E_2}(\boldsymbol{kk} - \boldsymbol{K}, \boldsymbol{T} - \boldsymbol{L})/c_E(\boldsymbol{k}, \boldsymbol{l});$
Fin Tant que
$\iddots$
Fin Tant que
Fin Tant que
$\iddots$
Fin Tant que
Fin Tant que
Retourner $\phi(g_{E_1}(\boldsymbol{K}, \boldsymbol{L}), g_{E_2}(\boldsymbol{kk} - \boldsymbol{K}, \boldsymbol{T} - \boldsymbol{L}))$
Fin Proc

4.4.2 Complexité

Théorème 4.9 *La complexité des procédures de génération associées aux ensembles d'objets d'une grammaire d'objets G (G en forme 1-2 et (G, ν) bien fondé) est, dans le pire des cas, en $O(k_1 \ldots k_n\, l_1 \ldots l_r(k_1 + \cdots + k_n + l_1 + \cdots + l_r))$.*

PREUVE :
Une procédure de génération consiste à construire récursivement un arbre de dérivation dans G, qui est binaire car G est en forme 1-2. Or, la taille de l'arbre de dérivation d'un objet de valuation $x^{\boldsymbol{k}}q^{\boldsymbol{l}}$ est proportionnelle à $k_1 + \cdots + k_n + l_1 + \cdots + l_r$. La génération d'un sommet de l'arbre coûte au pire un nombre d'opérations arithmétiques en $O(k_1 \ldots k_n\, l_1 \ldots l_r)$ (boucles imbriquées de la procédure précédente). Ainsi, la complexité de la génération de l'arbre total est dans le pire des cas en $O(k_1 \ldots k_n\, l_1 \ldots l_r(k_1 + \cdots + k_n + l_1 + \cdots + l_r))$.

□

4.4.3 Algorithme de génération aléatoire uniforme

Nous avons maintenant tous les ingrédients qui permettent de décrire un algorithme de génération aléatoire uniforme pour les objets d'une grammaire d'objets G selon un certain ensemble de valuations d'objets G-q-linéaire ν (à condition que le couple (G,ν) soit bien fondé).

Algorithme GénérationAléatoire :
Entrée : un couple (G,ν).
Sortie : procédures pour générer aléatoirement les objets engendrés par G.

- ⋆ Transformer G en forme 1-2 (Section 4.1) $\Longrightarrow$ (G_0,ν_0).
- ⋆ Vérifier que (G_0,ν_0) est bien fondé (Section 4.2), sinon erreur.
- ⋆ Pour chaque ensemble d'objets E de G_0, créer la procédure d'énumération c_E, puis calculer, une fois pour toutes, tous les coefficients jusqu'au rang $(\boldsymbol{k},\boldsymbol{l})$ (Section 4.3).
- ⋆ Pour chaque ensemble d'objets E de G_0, créer la procédure de génération g_E comme nous avons vu ci-dessus.

Les procédures de génération aléatoire que nous obtenons fournissent les arbres de dérivation des objets dans G_0, et non pas G. Une simple transformation linéaire en la taille des arbres $(O(k_1+\cdots+k_n+l_1+\cdots+l_r))$ donne les arbres de dérivation dans G. Cette dernière manipulation n'affecte donc pas la complexité des procédures de génération aléatoire. De plus, il est aisé, lors de l'implémentation, de supprimer cette dernière étape et d'effectuer les transformations au fur et à mesure de la génération.
Remarque : Dans le cas des spécifications, Flajolet, Zimmermann et Van Cutsem [61] ont montré que la génération d'une structure de taille n est en $O(n\log n)$ en utilisant un algorithme *boustrophédon* au lieu de $O(n^2)$ pour un algorithme séquentiel. Il serait intéressant de faire une analyse similaire pour voir si l'application du principe *boustrophédon* peut également améliorer la complexité de notre algorithme de génération.

4.5 Implémentation : le programme Maple qAlGO

Nous avons réalisé en *Maple* [25] le programme *qAlGO* qui met en œuvre tout ce que nous venons de développer dans les sections précédentes. L'implémentation de *qAlGO* est inspirée de celle de *CombStruct* (ou *Gaïa*) [99].

A partir d'une grammaire d'objets non ambiguë G et d'un ensemble de valuations d'objets G-linéaire ou G-q-linéaire, *qAlGO* construit automatiquement, pour tous les ensembles d'objets de G, les procédures d'énumération et les procédures de génération uniforme selon les valuations d'objets. Le caractère automatique du logiciel *qAlGO* en fait un outil très utile qui facilite l'étude expérimentale de diverses statistiques sur des objets combinatoires (voir des exemples Section 4.6 et Annexe A). Une restriction est toutefois faite sur les valuations d'objets : *qAlGO* gère seulement les valuations sur une ou deux variables (1 variable x ou 2 variables x et y pour le cas G-linéaire, et 2 variables x et q pour le cas G-q-linéaire). Etant donnée la complexité des algorithmes décrits plus haut, l'implémentation au-delà de deux variables serait quasiment inutilisable.

Nous décrivons dans la suite de cette section deux points particuliers concernant l'implémentation de *qAlGO*. Nous présentons tout d'abord un algorithme qui calcule la *valuation minimale* pour chaque ensemble d'objets d'une grammaire d'objets; ce calcul sert à la fois à tester si tous les ensembles d'objets sont productifs et à éviter certains calculs inutiles dans les procédures d'énumération et de génération. Puis, nous débattons sur les problèmes posés par les valuations q-linéaires lorsque nous voulons ramener automatiquement l'énumération et la génération à un seul paramètre, l'exposant de la variable q (par exemple l'aire pour les polyominos).

4.5.1 Calcul des valuations minimales

L'algorithme présenté dans cette section n'est effectivement implémenté dans *qAlGO* que pour des valuations d'objets sur une ou deux variables, comme tous les autres vus précédemment. Mais il se décrit très bien dans le cadre plus général que nous avons considéré jusqu'à maintenant.

Soient $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ une grammaire d'objets et ν un ensemble G-q-linéaire d'applications : $\nu = \{V_E : E \longrightarrow Mon[X,Q], E \in \mathcal{E}\}$. On appelle *valuation minimale* d'un ensemble d'objets E dans $\mathcal{E}$, et on note $vm(E)$, le monôme qui a pour exposant de chaque variable la valeur minimale de cet exposant pour tout $V_E(e)$, $e \in E$.

Remarque : La valuation minimale $vm(E)$ ne correspond pas forcément à la valuation d'un objet de E. Par exemple, si E est un ensemble réduit à deux objets, $E = \{e_1, e_2\}$, et si $V_E(e_1) = xq^2$ et $V_E(e_2) = x^2q$, alors $vm(E) = xq$ est différent à la fois de $V_E(e_1)$ et $V_E(e_2)$.

Proposition 4.10 *Soient G une grammaire d'objets et ν un ensemble G-q-linéaire d'applications à valeurs dans $Mon[X,Q]$. On peut calculer les valuations minimales des ensembles d'objets de G.*

PREUVE :

L'algorithme suivant calcule la valuation minimale de chaque ensemble d'objets. C'est une extension d'un algorithme de Zimmermann qui détermine "la valuation des non-terminaux d'une spécification" ([98], algorithme A page 28). Chaque $vm(E)$ est représenté par un vecteur de taille $n + r$, dont les composantes sont respectivement l'exposant des variables $x_1, \ldots, x_n, q_1, \ldots, q_r$.

Dans l'algorithme, nous séparons les calculs pour les exposants des x_i (vm_x), et ceux des q_i (vm_q). Nous désignons par $min(v_1, v_2)$ le vecteur obtenu en effectuant le minimum composante à composante des vecteurs v_1 et v_2.

Algorithme ValMin :

Entrée : un couple (G, ν) (G en forme 1-2).

Sortie : la valuation minimale de chaque ensemble d'objets E.

Pour chaque E tel que $E = e$ **Faire** $v_x^0(E) \leftarrow V_E(e)_x$ et $v_q^0(E) \leftarrow V_E(e)_q$

Pour chaque autre E **Faire** $v_x^0(E) \leftarrow \infty$ et $v_q^0(E) \leftarrow \infty$

$j \leftarrow 0$

Répéter

  $j \leftarrow j + 1$

  Pour chaque E **Faire**

    Si $E = e$ **Alors** $v_x^j(E) \leftarrow v_x^{j-1}(E)$ et $v_q^j(E) \leftarrow v_q^{j-1}(E)$

    Si $E = E_1 + E_2$ **Alors**

      $v_x^j(E) \leftarrow min(v_x^{j-1}(E_1), v_x^{j-1}(E_2))$

      $v_q^j(E) \leftarrow min(v_q^{j-1}(E_1), v_q^{j-1}(E_2))$

    Si $E = \phi(E_1)$ **Alors**

      $v_x^j(E) \leftarrow \lambda_{\phi x} + v_x^{j-1}(E_1)$

      $v_q^j(E) \leftarrow \lambda_{\phi q} + v_q^{j-1}(E_1) + v_x^{j-1}(E_1).A_1^\phi$

    Si $E = \phi(E_1, E_2)$ **Alors**

      $v_x^j(E) \leftarrow \lambda_{\phi x} + v_x^{j-1}(E_1) + v_x^{j-1}(E_2)$

      $v_q^j(E) \leftarrow \lambda_{\phi q} + v_q^{j-1}(E_1) + v_q^{j-1}(E_2) + v_x^{j-1}(E_1).A_1^\phi + v_x^{j-1}(E_2).A_2^\phi$

  Fin Pour

Jusqu'à ce que $v_x^j(E) = v_x^{j-1}(E)$ et $v_q^j(E) = v_q^{j-1}(E)$ pour tout $E \in \mathcal{E}$

Pour chaque E **Faire** $vm_x(E) \leftarrow v_x^j(E)$ et $vm_q(E) \leftarrow v_q^j(E)$

L'algorithme **ValMin** se termine. En effet, pour chaque j et E, nous avons, composante à composante, $v_x^j(E) \leq v_x^{j-1}(E)$ (resp. $v_q^j(E) \leq v_q^{j-1}(E)$) à cause de la croissance des fonctions min et $+$; et $v_x^j(E)$ (resp. $v_q^j(E)$) est minoré par le vecteur $(0, \ldots, 0)$.

□

Nous pouvons alors énoncer immédiatement le résultat simple suivant :

Corollaire 4.11 *Pour chaque ensemble d'objets E, toutes les composantes de $vm(E)$ calculées par* **ValMin** *sont finies si et seulement si E est productif.*

Ainsi, dans l'algorithme **BienFondé** de la Section 4.2, au lieu de calculer l'ensemble des ensembles d'objets productifs avec l'algorithme du Chapitre 1, nous utilisons l'algorithme **ValMin** pour déterminer les valuations minimales et nous vérifions qu'elles sont toutes finies.

Connaître les valuations minimales sert également à éviter des calculs inutiles dans les procédures d'énumération et de génération, en changeant notamment les bornes des indices des boucles.

Exemple 4.3 : En Section 4.3.2, nous donnons des formules permettant de calculer les coefficients $c_E(\boldsymbol{K}, \boldsymbol{L})$ pour un ensemble d'objets E. Dans le cas où l'ensemble E est défini par $E = \phi(E_1, E_2)$, les bornes des sommes peuvent être modifiées :

$$\sum_{\boldsymbol{i}=vm_x(E_1)}^{\boldsymbol{K}-vm_x(E_2)} \sum_{\boldsymbol{j}=vm_q(E_1)}^{\boldsymbol{L}-vm_q(E_2)} c_{E_1}(\boldsymbol{i},\boldsymbol{j}).c_{E_2}(\boldsymbol{K}-\lambda_{\phi x}-\boldsymbol{i}, \boldsymbol{L}-\lambda_{\phi q}-\boldsymbol{i}.A_1^{\phi}-(\boldsymbol{K}-\lambda_{\phi x}-\boldsymbol{i}).A_2^{\phi}-\boldsymbol{j}).$$

4.5.2 Cas des valuations d'objets q-linéaires

Introduisons le problème à l'aide d'exemples. Si nous voulons tirer aléatoirement des polyominos parallélogrammes selon l'aire, nous devons fournir à *qAlGO* la grammaire d'objets

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

et, sous une forme adéquate (voir l'Annexe A pour la syntaxe exacte), l'équation

$$f(x,q) = xq + xqf(x,q) + f(xq,q) + f(xq,q)f(x,q)$$

vérifiée par la fonction génératrice

$$f(x,q) = \sum_{e \in E_{pp}} x^{largeur(e)}\ q^{aire(e)}\ .$$

En effet, nous avons besoin de la largeur pour interpréter l'aire des polyominos sur les opérations d'objets.

Ainsi, les procédures que nous avons vues jusqu'à maintenant fournissent les procédures d'énumération $c(k,l)$ et de génération aléatoire $g(k,l)$ des polyominos parallélogrammes selon l'aire (l) *et la largeur* (k).

Pour obtenir l'énumération et la génération aléatoire seulement selon l'aire, il faut connaître, pour une aire donnée m, toutes les largeurs possibles, ou tout au moins une borne pour ces largeurs, $B(m)$.

Effectivement, le nombre total de polyominos parallélogrammes d'aire m est alors donné par

$$C(m) = \sum_{k=1}^{B(m)} c(k,m)\ .$$

Et pour générer aléatoirement un polyomino parallélogramme d'aire m, il suffit de tirer uniformément une valeur k_0 pour la largeur, et d'utiliser la procédure $g(k,l)$. L'algorithme est le suivant :

$k_0 := 1;$ (largeur minimale calculée avec **ValMin**)
$S := c(k_0, m)/C(m);$
$U :=$ Uniforme$([0,1]);$
Tant que $U > S$ **Faire**
  $k_0 := k_0 + 1;$
  $S := S + c(k_0, m)/C(m);$
Fin Tant que
$g(k_0, m);$

En fait, dans le cas des polyominos parallélogrammes, la borne $B(m)$ est tout simplement égale à m. Mais ce n'est pas vrai pour tous les objets et valuations.

Reprenons l'exemple des chemins de Motzkin (1-colorés), nous avons vu au Chapitre 2 que leur fonction génératrice selon la longueur (x) et l'aire(q) vérifie l'équation

$$M(x,q) = 1 + xM(x,q) + x^2 qM(xq,q)M(x,q) \ .$$

La borne $B(m)$ pour la longueur des chemins d'aire m est $B(m) = 2m$.

Dans le cas des arbres planaires (Chapitre 2), leur fonction génératrice selon le nombre de sommets (x) et la longueur du chemin interne (q) vérifie

$$P(x,q) = x + P(xq,q)P(x,q) \ ,$$

et la borne $B(m)$ pour le nombre de sommets des arbres ayant m comme longueur du chemin interne est $B(m) = m + 1$.

Ainsi, suivant les objets et les valuations, la nature de cette borne peut être très différente.

Pour l'instant, *qAlGO* ne gère pas la détermination automatique de cette borne ; l'utilisateur doit fournir la fonction de m qui permet de la calculer, et *qAlGO* crée alors les procédures que nous venons de voir.

Il reste donc à faire une étude pour un calcul automatique de la borne, ou d'une majoration raisonnable de celle-ci, si la détermination exacte s'avère trop coûteuse.

4.6 Etudes expérimentales et statistiques

Il existe dans la littérature de nombreux algorithmes de codage de polyominos par des mots de langages algébriques (souvent Dyck ou Motzkin), où le périmètre du polyomino correspond

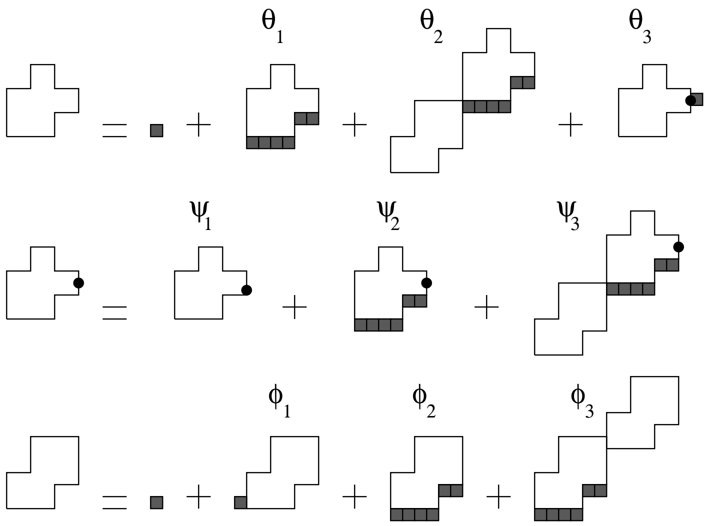


Figure 41: Une grammaire d'objets non ambiguë pour les polyominos convexes dirigés.

à la longueur du mot. Ainsi, la génération aléatoire de diverses classes de polyominos selon le périmètre est facilement accessible dès que l'on sait générer les mots. C'est le cas pour les polyominos escaliers, parallélogrammes et convexes dirigés. La technique diffère un peu pour les polyominos convexes (voir [69, 6]). En ce qui concerne la génération de polyominos selon l'aire, quelques cas isolés ont été traités par des méthodes diverses : diagrammes de Ferrers ou partitions [77], polyominos verticalement convexes dirigés [10], animaux dirigés [9]. Une collection des algorithmes connus de génération aléatoire de polyominos est implémentée au sein du logiciel POLYLAB développé par Bertoli, Del Lungo et Ulivi [16].

Nous présentons ici un exemple d'études expérimentales réalisé avec *qAlGO* (et aussi *Maple* et *CalICo*) sur la génération aléatoire selon l'aire des polyominos parallélogrammes et des polyominos convexes dirigés.

Pour cela, nous donnons en entrée à *qAlGO* la grammaire d'objets des polyominos convexes dirigés représentée Figure 41 (elle contient celle des polyominos parallélogrammes), ainsi que les valuations d'objets qui permettent d'obtenir l'énumération et la génération selon la largeur (x) et l'aire (q). Les équations sur les fonctions génératrices sont les suivantes

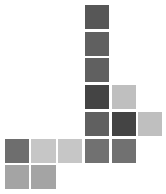
$$\begin{cases} CD(x,q) &= xq + CD(xq,q) + P(x,q)CD(xq,q) + xqCDm(x,q) \\ CDm(x,q) &= CD(x,q) + CDm(xq,q) + P(x,q)CDm(xq,q) \\ P(x,q) &= xq + xqP(x,q) + P(xq,q) + P(xq,q)P(x,q) \end{cases}$$

où $CD(x,q)$, $CDm(x,q)$, et $P(x,q)$ sont respectivement les séries génératrices des polyominos convexes dirigés, convexes dirigés marqués et parallélogrammes.

Nous pouvons alors générer aléatoirement selon l'aire ces objets; en fait, *qAlGO* fournit l'arbre de dérivation associé à l'objet dans la grammaire, par exemple

$$\theta_3(\psi_3(\phi_3(\phi_1(\square),\square),\psi_1(\theta_1(\theta_3(\psi_1(\theta_1(\theta_1(\theta_1(\square)))))))).$$

Il suffit alors d'évaluer cet arbre selon les opérations d'objets, et d'écrire le polyomino sous une forme comprise par l'interface *XAnimal* de *CalICo* [41, 83] ; elle nous permet ainsi de visualiser le polyomino ci-dessous.



C'est grâce à cette simple manipulation que nous avons constaté l'allure éffilée des polyominos convexes dirigés et parallélogrammes générés uniformément selon l'aire. Ils sont d'aspect bien plus compact lorsque le paramètre considéré est le périmètre. Nous donnons des exemples de génération de ces polyominos selon l'aire et le périmètre Figures 42, 43, 44 et 45.

Pour rendre compte de cette apparence mince et allongée, nous calculons expérimentalement la valeur moyenne de deux paramètres : la *hauteur d'une colonne* et le *nombre de contacts entre deux colonnes*, c'est à dire le nombre de cellules par lesquelles deux colonnes adjacentes sont en contact. En effectuant 1000 tirages de polyominos convexes dirigés d'aire 100, et 1000 tirages de polyominos parallélogrammes d'aire 200, nous obtenons comme valeurs moyennes :

- $2,37$ pour la hauteur d'une colonne,
- $1,37$ pour le nombre de contacts entre deux colonnes.

Ces chiffres viennent effectivement confirmer l'allure effilée des polyominos.
Remarque : Le résultat pour la hauteur moyenne d'une colonne $(2,37)$ coïncide exactement avec celui de Bender [12] calculé par des méthodes d'analyse asymptotique.

La différence de 1 entre ces deux paramètres moyens s'explique très simplement en remarquant que cette différence pour un polyomino tend vers le rapport de la hauteur et de la largeur du polyomino, et que par symétrie des objets, ce rapport tend lui-même vers 1.[1]

En plus de leur allure effilée, les polyominos convexes dirigés générés selon l'aire ont une autre caractéristique tout à fait remarquable : la petite taille du "coin" du polyomino qui fait que ce n'est pas un parallélogramme (voir Figure 44). Il serait intéressant de faire une étude pour expliquer ce fait.

[1] Merci à Viennot de nous avoir éclairés sur ce point!

4.7 Conclusions et perspectives

Nous venons de développer une approche systématique de la génération aléatoire uniforme d'objets combinatoires selon divers paramètres, et spécialement des paramètres non algébriques. Son implémentation en *Maple* sous le nom de *qAlGO* nous permet de montrer que notre méthode est effective. Le souci principal de notre travail n'est pas d'optimiser la complexité des algorithmes mis en œuvre ou de fournir des algorithmes linéaires, comme c'est souvent le cas dans le domaine de la génération aléatoire. L'avantage crucial de notre méthode est son caractère automatique et générique qui facilite énormément la diversification des cadres d'applications.

Dans la section précédente, nous avons, à titre d'exemple, réalisé une ébauche d'études expérimentales concernant des valeurs moyennes de paramètres relatifs aux polyominos ; nous pouvons imaginer bien d'autres études statistiques sur des chemins, des arbres, etc. De nombreuses conjectures sont donc à portée de main !

Une autre idée d'utilisation de notre programme *qAlGO* est plus liée à la notion de grammaires d'objets isomorphes et à leurs applications dans le domaine des bijections étudiées au Chapitre 3. Notamment, la bijection entre les objets de deux grammaires isomorphes s'établit sur les arbres de dérivation en échangeant les opérations d'objets et les terminaux. Rappelons que le principe de *qAlGO* est de générer les arbres de dérivation de la grammaire d'objets non ambiguë fournie par l'utilisateur, qui peut ensuite évaluer ces arbres (à l'aide de fonctions *Maple*) pour obtenir les objets sous le format qu'il désire.
Ainsi, si l'utilisateur dispose de plusieurs grammaires d'objets isomorphes pour différents objets, il peut, à partir d'une des grammaires, réaliser "gratuitement" la génération de tous les objets en bijection. Il lui suffit de générer les arbres de derivation d'une grammaire, et d'évaluer ces arbres avec les diverses fonctions correspondant aux opérations d'objets des différentes grammaires d'objets.
De plus, cela offre une perspective supplémentaire pour les problèmes de détermination des paramètres préservés par ces bijections que nous avons évoqués à la dernière section du Chapitre 3. En effet, le fait de pouvoir très facilement générer des objets en bijection et les visualiser à l'aide de *CalICo* nous permet d'imaginer quelques études systématiques et visuelles ; nous pensons notamment à une utilisation judicieuse des couleurs sous *CalICo* qui peut être un atout important pour mettre en évidence le passage des paramètres.

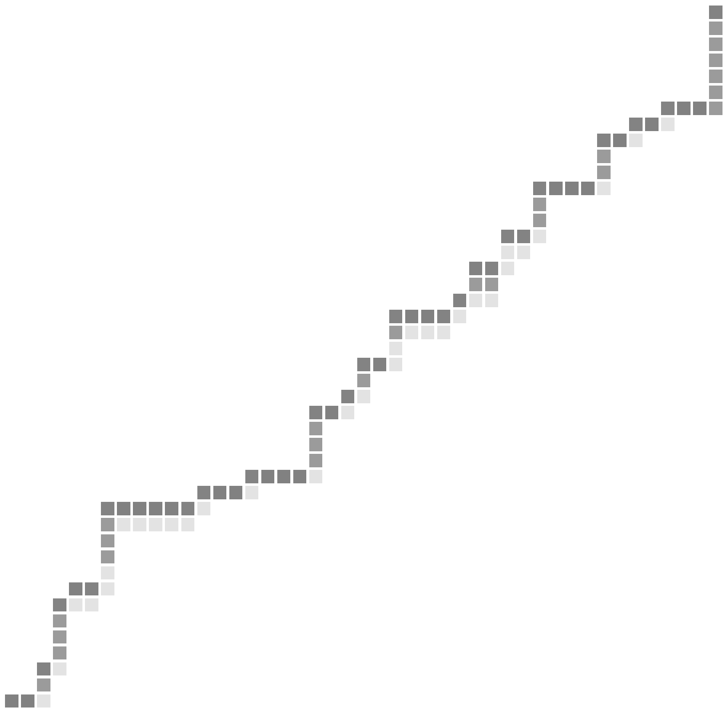


Figure 42: Un polyomino parallélogramme aléatoire d'aire 100.

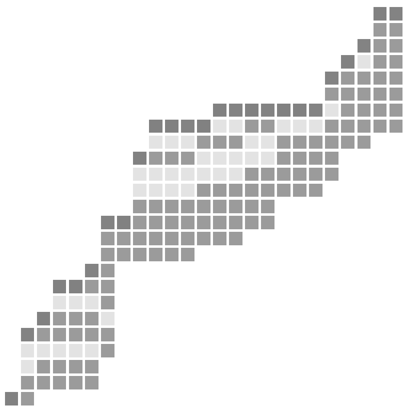


Figure 43: Un polyomino parallélogramme aléatoire de périmètre 100.

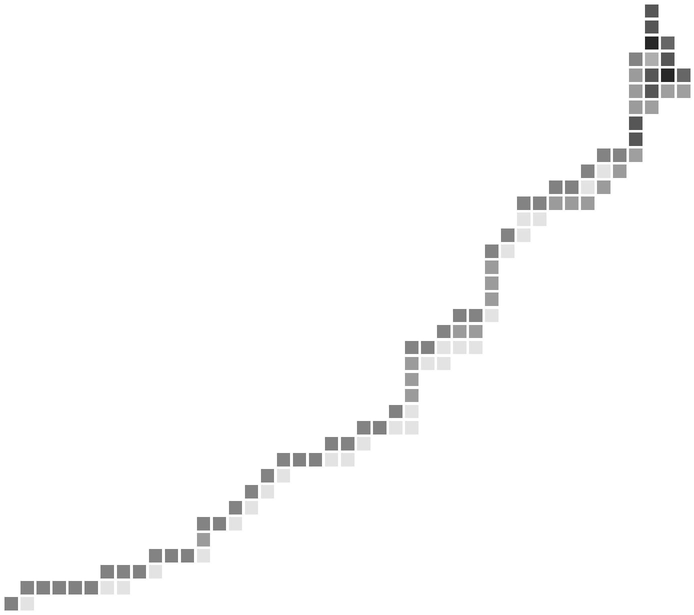


Figure 44: Un polyomino convexe dirigé aléatoire d'aire 100.

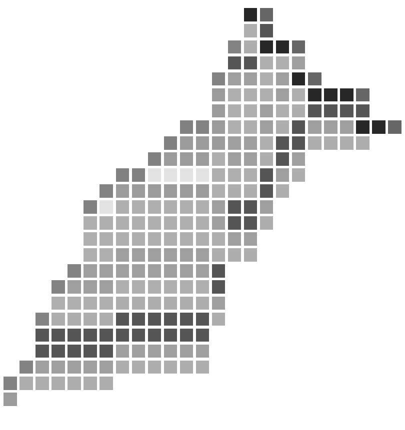


Figure 45: Un polyomino convexe dirigé aléatoire de périmètre 100.

Annexe A

Le Programme Maple qAlGO

Le programme *qAlGO* est un "package" *Maple* qui permet de compter et générer aléatoirement divers objets combinatoires selon différents paramètres, notamment non algébriques. A partir d'une grammaire d'objets non ambiguë G et d'un ensemble de valuations d'objets G-linéaire ou G-q-linéaire, *qAlGO* construit automatiquement, pour tous les ensembles d'objets de G, les procédures d'énumération et les procédures de génération uniforme selon les valuations d'objets.
Remarque : Pour décrire les fonctionnalités de *qAlGO*, nous nous appuyons régulièrement sur le même exemple : celui des polyominos parallélogrammes qui, rappelons le, sont engendrés par la grammaire d'objets non ambiguë Figure 46.

Installation

qAlGO peut être utilisé avec les deux versions de *Maple* [25] : *Maple V Release 2* ou *Release 3*.

Le fichier `qalgo.map` contient les sources *Maple* du programme *qAlGO*. Pour créer le "package" `qalgo.m`, il suffit de taper la commande ci-dessous.

```
% maple -s -q < qalgo.map
```

Alors, pour pouvoir charger facilement le "package" *qAlGO* depuis *Maple*, il faut ajouter la ligne suivante au fichier `.mapleinit` du répertoire d'accueil.

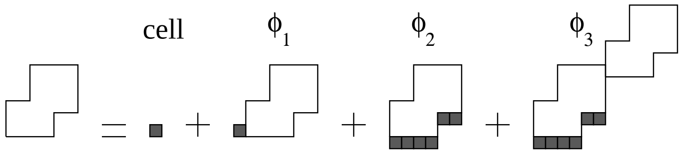


Figure 46: Grammaire d'objets non ambiguë pour les polyominos parallélogrammes.

```
libname := '/usr/labri/dutour/qAlGO',libname:
```

Ainsi, la simple commande `with(qalgo)` permet de charger le “package”.

```
% maple
> with(qalgo):
> ...
```

Syntaxe des grammaires d'objets et des valuations d'objets

Les grammaires d'objets

L'utilisateur fournit les grammaires d'objets à *qAlGO* sous la forme de leur notation équationnelle, c'est à dire un système d'équations de la forme

$$\left\{E_i = \sum_{e_i \in T_{E_i}} e_i \;+ \sum_{dest(\phi)=E_i} \phi(E_{i_{1_\phi}}, \ldots, E_{i_{k_\phi}})\right\}_{i=1,\ldots,n}.$$

Pour les polyominos parallélogrammes, on écrit

```
> paral := {P = cell + phi1(P) + phi2(P) + phi3(P,P)}:
```

où `P`, `cell`, `phi1`, `phi2` et `phi3` sont des variables formelles de *Maple*, c'est à dire des variables auxquelles on n'a pas encore affectées de valeur.

Les valuations d'objets

qAlGO accepte trois types de valuations : `linear`, `linear2` et `qlinear`. Le type `linear` correspond aux valuations linéaires sur une variable (x), le type `linear2` aux valuations linéaires sur deux variables (x et y) et le type `qlinear` aux valuations q-linéaires sur deux variables (x et q).

Les valuations d'objets sont fournies à *qAlGO* sous la forme d'une séquence d'“éléments” décrivant l'action des valuations sur chaque objet terminal et chaque opération d'objets de la grammaire à laquelle elles s'appliquent. Par exemple, pour les polyominos parallélogrammes, elles s'expriment de la manière suivante :

```
> valparal := [[cell,...], [phi1,...], [phi2,...], [phi3,...]]:
```

Les `...` sont différents selon que les valuations sont de type `linear`, `linear2` ou `qlinear`.

Le type `linear`

C'est une valuation d'objets du type $V : e \mapsto x^{param(e)}$, telle que

pour un objet terminal `o`, on a $V(\mathtt{o}) = x^{\mathtt{a}}$, et on écrit

```
[o, a] ,
```

et pour une opération d'objets phi avec $dom(\mathtt{phi}) = E_1 \times \ldots \times E_n$, on a $V(\mathtt{phi}(E_1, \ldots, E_n)) = x^{\mathtt{b}}\, V(E_1) \ldots V(E_n)$, et on écrit

```
[phi, b] .
```

Par exemple, pour les polyominos parallélogrammes, si on considère la valuation "périmètre", $V_1 : e \mapsto x^{périmètre(e)}$, on écrit

```
> valparal1 := [[cell, 4], [phi1, 2], [phi2, 2], [phi3, 0]]:
```

Le type `linear2`

C'est une valuation d'objets du type $V : e \mapsto x^{param1(e)} y^{param2(e)}$, telle que

pour un objet terminal o, on a $V(\mathtt{o}) = x^{\mathtt{a1}} y^{\mathtt{a2}}$, et on écrit

```
[o, a1, a2] ,
```

et pour une opération d'objets phi avec $dom(\mathtt{phi}) = E_1 \times \ldots \times E_n$, on a $V(\mathtt{phi}(E_1, \ldots, E_n)) = x^{\mathtt{b1}} y^{\mathtt{b2}}\, V(E_1) \ldots V(E_n)$, et on écrit

```
[phi, b1, b2] .
```

Par exemple, pour les polyominos parallélogrammes, si on considère la valuation "largeur et hauteur", $V_2 : e \mapsto x^{largeur(e)} y^{hauteur(e)}$, on écrit

```
> valparal2 := [[cell, 1, 1], [phi1, 1, 0], [phi2, 0, 1], [phi3, 0, 0]]:
```

Le type `qlinear`

C'est une valuation d'objets du type $V : e \mapsto x^{param1(e)} q^{param2(e)}$, telle que

pour un objet terminal o, on a $V(\mathtt{o}) = x^{\mathtt{k}} q^{\mathtt{l}}$, et on écrit

```
[o, k, l] ,
```

et pour une opération d'objets phi avec $dom(\mathtt{phi}) = E_1 \times \ldots \times E_n$, on a $V(\mathtt{phi}(E_1, \ldots, E_n)) = x^{\mathtt{k}} q^{\mathtt{l}}\; V(E_1)|_{x \leftarrow xq^{\mathtt{m1}}} \ldots V(E_n)|_{x \leftarrow xq^{\mathtt{mn}}}$, et on écrit

```
[phi, k, l, [m1,...,mn]] .
```

Par exemple, pour les polyominos parallélogrammes, si on considère la valuation "largeur et aire", $V_3 : e \mapsto x^{largeur(e)} q^{aire(e)}$, on écrit

```
> valparal3 := [[cell, 1, 1], [phi1, 1, 1, [0]], [phi2, 0, 0, [1]],
               [phi3, 0, 0, [1,0]]]:
```

Utilisation des fonctions disponibles

Quatre fonctions du "package" *qAlGO* sont disponibles pour l'utilisateur : `compile`, `countgo`, `drawgo` et `drawgoall`. Nous décrivons ici le rôle de ces fonctions et la manière de les utiliser.

La fonction `compile`

Elle construit, à partir d'une grammaire d'objets et d'une valuation d'objets données, les procédures d'énumération et les procédures de génération aléatoire.
Important : La fonction `compile` doit être exécutée avant toute utilisation, pour les mêmes données, des trois autres fonctions.
La fonction `compile` prend comme arguments

- une grammaire d'objets,
- un ensemble de valuations d'objets,
- le type des valuations (`linear`, `linear2` ou `qlinear`),
- et, si le type est `qlinear`, une fonction qui permet de calculer la *borne* (se référer à la Section 4.5.2 du Chapitre 4).

Prenons l'exemple des polyominos parallélogrammes, les commandes *Maple* sont :

```
> compile(paral,valparal1,linear):
> compile(paral,valparal2,linear2):
> compile(paral,valparal3,qlinear,Identity):
```

où la fonction `Identity`, définie par défaut au sein du "package" *qAlGO*,

```
Identity := proc(m) m end:
```

est la simple *fonction identité*.

La fonction `countgo`

Elle permet de compter (selon des valuations d'objets données) les objets engendrés par un ensemble d'objets d'une grammaire d'objets. Elle prend comme arguments

- une grammaire d'objets,
- un ensemble de valuations d'objets,
- le type des valuations (`linear`, `linear2` ou `qlinear`),
- un ensemble d'objets de la grammaire d'objets,
- et une ou deux valeurs numériques pour les valuations selon qu'elles sont de type `linear`, `linear2` ou `qlinear` :
  - `linear` : une valeur,
  - `linear2` : deux valeurs,
  - `qlinear` : deux valeurs, ou une valeur (pour le paramètre exposant de q) lorsqu'on veut la somme sur toutes les valeurs possibles du paramètre exposant de x.

Considérons de nouveau l'exemple des polyominos parallélogrammes. Voici les commandes pour obtenir
le nombre de polyominos parallélogrammes de périmètre 50 :

```
> countgo(paral,valparal1,linear,P,50);
                              1289904147324
```

le nombre de polyominos parallélogrammes de largeur 20 et de hauteur 25 :

```
> countgo(paral,valparal2,linear2,P,20,25);
                          56386538045363054709180
```

le nombre de polyominos parallélogrammes de largeur 20 et d'aire 40 :

```
> countgo(paral,valparal3,qlinear,P,20,40);
                              9845061662866
```

le nombre de polyominos parallélogrammes d'aire 40 :

```
> countgo(paral,valparal3,qlinear,P,40);
                             102664213847468
```

Remarque : Le calcul des coefficients n'est effectué qu'une seule fois pour chaque valeur des paramètres. En effet, lorsque `countgo(...,n)` (ou `countgo(...,n,m)`) est calculé, sa valeur est stockée grâce à l'option `remember` de *Maple* pour les procédures ; ainsi, l'appel suivant à `countgo(...,n)` (ou `countgo(...,n,m)`) demande un temps constant.

La fonction `drawgo`

Elle permet de générer aléatoirement et uniformément (selon des valuations d'objets données) les objets engendrés par un ensemble d'objets d'une grammaire d'objets. Elle prend en entrée les mêmes arguments que la fonction précédente `countgo`.

Par exemple, générons des polyominos parallélogrammes
de périmètre 20 :

```
> drawgo(paral,valparal1,linear,P,20);
          phi2(phi2(phi3(phi2(phi2(cell)), phi1(phi2(cell)))))
```

de largeur 7 et d'aire 15 :

```
> drawgo(paral,valparal3,qlinear,P,7,15);
phi1(phi1(phi3(phi3(phi2(phi2(cell)), phi1(cell)), phi3(phi2(cell), cell))))
```

d'aire 20 :

```
> drawgo(paral,valparal3,qlinear,P,20);
phi1(phi1(phi1(phi1(
 phi1(phi1(phi1(phi3(phi1(phi2(cell)),
                phi3(phi3(phi2(cell), phi2(cell)), cell)))))
 ))))
```

La fonction `drawgoall`

Elle permet de générer *tous* les objets engendrés par un ensemble d'objets d'une grammaire d'objets ayant des valeurs données pour les valuations d'objets. Elle prend en entrée les mêmes arguments que les fonctions précédentes `countgo` et `drawgo`.

Si nous voulons générer tous les polyominos parallélogrammes d'aire 4, qui sont au nombre de 9,

```
> countgo(paral,valparal3,qlinear,P,4);
                                        9
```

nous tapons la commande

```
> drawgoall(paral,valparal3,qlinear,P,4);
[phi2(phi2(phi2(cell))), phi1(phi2(phi2(cell))), phi2(phi1(cell)),
    phi3(cell, phi2(cell)), phi3(phi2(cell), cell), phi1(phi1(phi2(cell))),
    phi1(phi3(cell, cell)), phi3(cell, phi1(cell)), phi1(phi1(phi1(cell)))]
```

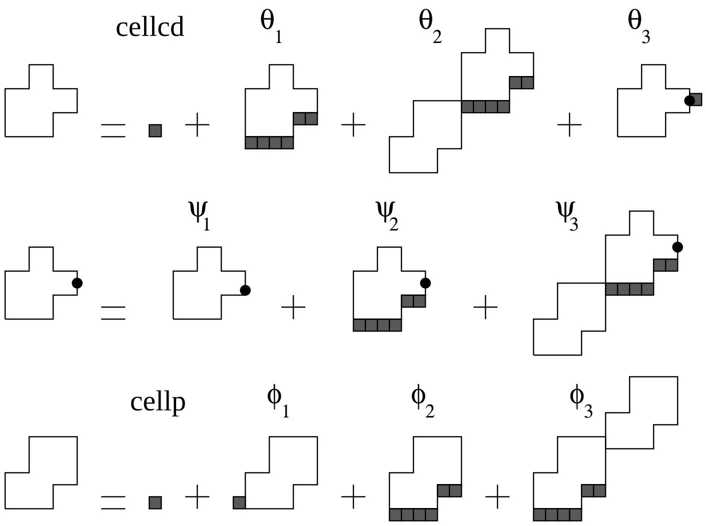


Figure 47: Une grammaire d'objets non ambiguë pour les polyominos convexes dirigés.

Un exemple détaillé

Nous détaillons ici un exemple de calcul expérimental réalisé à l'aide de *qAlGO* décrit à la Section 4.6 du Chapitre 4 : le calcul de la valeur moyenne de la *hauteur d'une colonne* pour les polyominos convexes dirigés tirés aléatoirement selon l'aire (grammaire d'objets Figure 47).

Tout d'abord, nous écrivons la grammaire d'objets et les valuations d'objets en *Maple*, et nous construisons les procédures d'enumération et de génération.

```
% maple
> with(qalgo):
> convdir := {CD = cellcd + theta1(CD) + theta2(P,CD) + theta3(CDM),
              CDM = psi1(CD) + psi2(CDM) + psi3(P,CDM),
              P = cellp + phi1(P) + phi2(P) + phi3(P,P)}:
> valcdaire := [[cellcd, 1, 1], [theta1, 0, 0, [1]], [theta2, 0, 0, [0,1]],
                [theta3, 1, 1, [0]],
                [psi1, 0, 0, [0]], [psi2, 0, 0, [1]], [psi3, 0, 0, [0,1]],
                [cellp, 1, 1], [phi1, 1, 1, [0]], [phi2, 0, 0, [1]],
                [phi3, 0, 0, [1,0]]]:
> compile(convdir,valcdaire,qlinear,Identity):
```

Pour un polyomino donné, la hauteur moyenne d'une colonne est égale à la somme des hauteurs de ses colonnes divisée par sa largeur (nombre de colonnes). La somme des hauteurs de ses colonnes est en fait son aire, et la largeur est calculée par la fonction *Maple* suivante.

```
> largeur := proc(polyo)
    local f;
    if polyo=cellcd or polyo=cellp then 1
    else
        f := op(0,polyo);
        if f=phi1 or f=theta3 then
            1 + largeur(op(polyo))
        elif f=phi2 or f=theta1 or f=psi1 or f=psi2 then
            largeur(op(polyo))
        else   # f=phi3 or f=theta2 or f=psi3
            largeur(op(1,polyo)) + largeur(op(2,polyo))
        fi;
    fi;
  end:
```

Le calcul de tous les coefficients d'énumération pour les polyominos d'aire 100 est assez long (environ deux jours), ils ont donc été calculés au préalable et sauvegardés dans un fichier nommé `valcdaire100.m`. Nous chargeons ce fichier pour les utiliser.

```
> read `valcdaire100.m`:
```

Nous générons alors 1000 polyominos convexes dirigés d'aire 100 que nous stockons dans une table,

```
> tablecd :=
  table([seq(drawgo(convdir,valcdaire,qlinear,CD,100),i=1..1000)]):
```

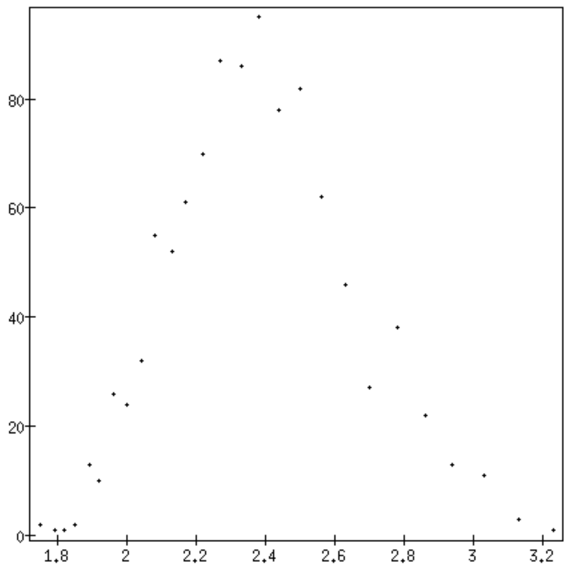
puis nous calculons la hauteur moyenne des colonnes pour chacun d'eux, et une valeur moyenne pour les 1000 tirages.

```
> for i from 1 to 1000 do
      hm[i] := evalf(100/largeur(tablecd[i])):
  od:
> sum(hm[j],j=1..1000)/1000;
                                2.356991519
```

Pour visualiser la distribution des valeurs moyennes des 1000 polyominos d'aire 100, nous fixons à 3 le nombre de chiffres significatifs (2 après la vigule), et nous construisons l'histogramme de ces valeurs : `t[j]` est le nombre de polyominos ayant j comme hauteur moyenne des colonnes.

```
> Digits :=3:
> t := `t`:
```

```
> for i from 1 to 1000 do
     j := evalf(hm[i]):
     if t[j]=evaln(t[j]) then t[j]:=1 else t[j]:=t[j]+1 fi:
  od:
> vals := sort(map(op,[indices(t)])):
> plot([seq([i,t[i]],i=vals)]);
```



Bibliographie

[1] Aho (A.), Sethi (R.) et Ullman (J.). – *Compilateurs, principes, techniques et outils.* – InterEditions, Paris, 1989.

[2] Alonso (L.). – *Structures arborescentes : algorithmes de génération, problème de l'inclusion, relations maximin.* – Thèse, Université Paris XI Orsay, Novembre 1992.

[3] Arnold (D.B.) et Sleep (M.R.). – Uniform random generation of balanced parenthesis strings. *ACM Trans. Programming Languages and Systems*, vol. 2, n° 1, 1980, pp. 122–128.

[4] Arques (D.), Eyrolles (G.), Janey (N.) et Viennot (X.). – Combinatorial analysis of ramified patterns and computer imagery of trees. *Dans : SIGGRAPH'89.* Computer Graphics. – Boston, USA, 1989.

[5] Autebert (J.M.). – *Langages algébriques.* – Masson, 1987.

[6] Barcucci (E.), Del Lungo (A.), Nivat (M.) et Pinzani (R.). – *Reconstructing convex poyominoes from their horizontal and vertical projections.* – Rapport technique n° RT 12/94, Dipartimento di Sistemi e Informatica, Université de Florence, 1994.

[7] Barcucci (E.), Del Lungo (A.), Pergola (E.) et Pinzani (R.). – *A construction for enumearting k-coloured Motzkin paths.* – Rapport technique n° RT 9/95, Dipartimento di Sistemi e Informatica, Université de Florence, 1995.

[8] Barcucci (E.), Del Lungo (A.), Pergola (E.) et Pinzani (R.). – Towards a methodology for tree enumeration. *Dans : Actes du 7ème Colloque Séries Formelles et Combinatoire Algébrique*, éd. par Leclerc (B.) et Thibon (J.Y.), pp. 53–65. – Université de Marne-la-Vallée, 1995.

[9] Barcucci (E.), Pinzani (R.) et Sprugnoli (R.). – Génération aléatoire des animaux dirigés. *Dans : Actes de l'Atelier Franco-Québécois de Combinatoire, publi LaCIM 10*, éd. par Labelle (J.) et Penaud (J. G.). – Université du Québec à Montréal, 1991.

[10] Barcucci (E.), Pinzani (R.) et Sprugnoli (R.). – Directed column-convex polyominoes by recurrence relations. *Lecture Notes in Computer Science*, vol. 668, 1993, pp. 282–298.

[11] Barcucci (E.), Pinzani (R.) et Sprugnoli (R.). – The random generation of underdiagonal walks. *Discrete Math.*, vol. 139, 1995, pp. 3–18.

[12] Bender (E.A.). – Convex n-ominoes. *Discrete Math.*, vol. 8, 1974, pp. 219–226.

[13] Berstel (J.) et Pocchiola (M.). – Random generation of finite Sturmian words. *Dans : Actes du 5ème Colloque Séries Formelles et Combinatoire Algébrique*, pp. 69–82. – Université de Florence, 1993.

[14] Berstel (J.) et Reutenauer (C.). – Recognizable formal power series on tree. *Theoretical Computer Science*, vol. 18, 1982, pp. 115–148.

[15] Berstel (J.) et Reutenauer (C.). – *Les séries rationnelles et leurs langages.* – Masson, 1984.

[16] Bertoli (F.), Del Lungo (A.) et Ulivi (F.). – Polylab : A package for the study of polyominoes. *Dans : Actes du 7ème Colloque Séries Formelles et Combinatoire Algébrique*, éd. par Leclerc (B.) et Thibon (J.Y.), pp. 77–82. – Université de Marne-la-Vallée, 1995.

[17] Bétréma (J.) et Penaud (J.G.). – Animaux et arbres guingois. *Theoretical Computer Science*, vol. 117, 1993, pp. 67–89.

[18] Bousquet-Mélou (M.). – *q-énumération de polyominos convexes.* – Thèse, Université Bordeaux I, mai 1991.

[19] Bousquet-Mélou (M.). – A method for enumeration of various classes of column-convex polygons. – 1993. Rapport LaBRI $n^o 378-93$, Université Bordeaux I. A paraître dans *Discrete Math.*

[20] Bousquet-Mélou (M.). – New enumerative results on two-dimensional directed animals. – 1995. Rapport LaBRI $n^o 1082-95$, Université Bordeaux I.

[21] Bousquet-Mélou (M.). – Percolation models and animals. – 1995. Rapport LaBRI $n^o 993-95$, Université Bordeaux I.

[22] Bousquet-Mélou (M.) et Fédou (J.M.). – The generating function of convex polyominoes : the resolution of a q-differential system. *Discrete Math.*, vol. 137, 1995, pp. 53–75.

[23] Bousquet-Mélou (M.) et Viennot (X.G.). – Empilements de segments et q-énumération de polyominos convexes dirigés. *J. Comb. Theory Ser. A*, vol. 60, n° 2, 1992, pp. 196–224.

[24] Burstall (R.M.) et Darlington (J.). – A transformation system for developing recursive programs. *J. ACM*, vol. 24, n° 1, 1977, pp. 44–67.

[25] Char (B.W.), Geddes (K.O), Gonnet (G.H.), Leong (B.L.), Monagan (M.B.) et Watt (S.M.). – *Maple V : Langage Reference Manual.* – Springer-Verlag, 1991.

[26] Cori (R.). – *Un code pour les graphes planaires et ses applications.* – Société Mathématique de France, 1975. Astérisque 27.

[27] Cori (R.) et Richard (J.). – Enumération des graphes planaires à l'aide de séries formelles en variables non commutatives. *Discrete Math.*, vol. 2, 1972, pp. 115–162.

[28] Cori (R.) et Vauquelin (B.). – Planar maps are well labeled trees. *Can. J. Math.*, vol. 33, n° 5, 1981, pp. 1023–1042.

[29] Courcelle (B.). – Basics notions of universal algebra for language theory and graph grammars. – A paraître dans *Theoretical Computer Science.*

[30] Courcelle (B.). – Equivalences and transformations of regular systems - applications to recursive program schemes and grammars. *Theoretical Computer Science*, vol. 42, 1986, pp. 1–122.

[31] Courcelle (B.). – Recursive queries and context free graph grammars. *Theoretical Computer Science*, vol. 78, 1991, pp. 217–244.

[32] Courcelle (B.). – Graph grammars, monadic second-order logic and the theory of graph minors. *Bulletin of EATCS*, vol. 46, february 1992.

[33] Delest (M.). – Algebraic languages: a bridge between combinatorics and computer science. – A paraître dans *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.*

[34] Delest (M.). – Generating functions for column-convex polyominoes. *J. Comb. Theory Ser. A*, vol. 48, 1988, pp. 217–244.

[35] Delest (M.). – Polyominoes and animals: some recent results. *J. of Maths Chemistry*, vol. 8, 1991, pp. 3–18.

[36] Delest (M.). – Langages algébriques : à la frontière entre la Combinatoire et l'Informatique. *Dans : Actes du* $6^{ème}$ *Colloque Séries Formelles et Combinatoire Algébrique*, pp. 69–78. – DIMACS, Université de Rutgers, USA, 1994.

[37] Delest (M.), Dubernard (J.P.) et Dutour (I.). – Parallelogram polyominoes and corners. – A paraître dans *Journal of Symbolic Computation.*

[38] Delest (M.) et Dulucq (S.). – Enumeration of directed column-convex animals with a given perimeter and aera. *Croatica Chemica Acta*, vol. 66, n° 1, 1993.

[39] Delest (M.) et Fédou (J.M.). – Counting polyominoes using attribute grammars. *Dans : Workshop on Attribute Grammars*, éd. par Deransart (P.) et Jourdan (M.), pp. 46–60. – Lecture Notes in Computer Science 461, 1990.

[40] Delest (M.) et Fédou (J.M.). – Attribute grammars are useful for combinatorics. *Theoretical Computer Science*, vol. 98, 1992, pp. 65–76.

[41] Delest (M.), Fédou (J.M.), Melançon (G.) et Rouillon (N.). – Faire des mathématiques avec CalICo. *Bulletin AMQ association mathématique du québec*, mai 1994, pp. 30–37.

[42] Delest (M.) et Viennot (X.G.). – Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, vol. 34, 1984, pp. 169–206.

[43] Denise (A.). – Génération aléatoire uniforme de mots de langages rationnels. – A paraître dans *Theoretical Computer Science*.

[44] Denise (A.). – Génération aléatoire et uniforme de mots. *Dans : Actes du 5ème Colloque Séries Formelles et Combinatoire Algébrique*, éd. par Barlotti (A.), Delest (M.) et Pinzani (R.), pp. 153–164. – Université de Florence, 1993. A paraître dans *Discrete Math.*

[45] Denise (A.) et Simion (R.). – Two combinatorial statistics on Dyck paths. *Discrete Math.*, vol. 137, 1995, pp. 155–176.

[46] Dhar (D.). – Equivalence of the two-dimensional directed animal problem to Baxter hard-square lattice-gas model. *Phys. Rev. Lett.*, vol. 49, 1982, pp. 959–962.

[47] Dhar (D.). – Exact solution of a directed-site animals enumeration in 3 dimensions. *Phys. Rev. Lett.*, vol. 59, 1983, pp. 853–856.

[48] Dubernard (J.P.). – *q-grammaires et polyominos parallélogrammes.* – Thèse, Université Bordeaux I, juin 1993.

[49] Dubernard (J.P.) et Dutour (I.). – Enumération de polyominos convexes dirigés. *Dans : Actes du 6ème Colloque Séries Formelles et Combinatoire Algébrique*, pp. 109–118. – DIMACS, Université de Rutgers, USA, 1994. A paraître dans *Discrete Math.*

[50] Ehrig (H.) et Mahr (B.). – *Fundamentals of algebraic specifications 1 : equations and initial semantics.* – Springer, 1985.

[51] Fédou (J.M.). – Enumeration of skew Ferrers diagrams and basic Bessel functions. *J. Stat. Planning & Inference*, vol. 34, 1993, pp. 107–123.

[52] Fédou (J.M.) et Rouillon (N.). – Polyominos et q-analogues des fonctions de Bessel, une preuve combinatoire. – 1994. Rapport LaBRI $n^o 818-94$, Université Bordeaux I.

[53] Feretić (S.) et Svrtan (D.). – On the number of column convex polyominoes with given perimeter and number of columns. *Dans : Actes du 5ème Colloque Séries Formelles et Combinatoire Algébrique*, pp. 201–214. – Université de Florence, 1993.

[54] Flajolet (P.). – Combinatorial aspects of continued fractions. *Discrete Math.*, vol. 32, 1980, pp. 125–161.

[55] Flajolet (P.). – Elements of a general theory of combinatorial structures. *Lecture Notes in Computer Science*, vol. 199, 1985, pp. 112–127.

[56] Flajolet (P.), Françon (J.) et Vuillemin (J.). – Sequences of operations analysis for dynamic data structures. *J. of Algorithms*, vol. 1, 1980, pp. 111–141.

[57] Flajolet (P.), Goldwurm (M.) et Steyaert (J.M.). – Random generation and context-free languages. – 1990. Manuscrit.

[58] Flajolet (P.), Salvy (B.) et Zimmermann (P.). – $\Lambda_\Upsilon\Omega$: an assistant algorithms analyser. *Dans : Proceedings AAECC'6.* – Rome, Italy, 1988.

[59] Flajolet (P.), Salvy (B.) et Zimmermann (P.). – Automatic average-case analysis of algorithms. *Theoretical Computer Science*, vol. 79, n° 1, 1991, pp. 37–109.

[60] Flajolet (P.), Salvy (B.) et Zimmermann (P.). – *Lambda-Upsilon-Omega, the 1989 cookbook.* – Rapport technique n° 1073, INRIA, 1992.

[61] Flajolet (P.), Zimmermann (P.) et Van Cutsem (B.). – A calculus for the random generation of combinatorial structures. *Theoretical Computer Science*, vol. 132, 1994, pp. 1–35.

[62] Goldwurm (M.). – Random generation of words in an algebraic language in linear binary space. *Information Processing Letters*, vol. 54, 1995, pp. 229–233.

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

[64] Greene (D.H.). – *Labelled formal languages and their uses.* – Thèse de PhD, Stanford University, june 1983.

[65] Guessarian (I.). – Algebraic semantics. *Lecture Notes in Computer Science*, vol. 99, 1981.

[66] Guttmann (A.J.). – Planar polygons : regular, convex, almost convex, staircase and row convex. *Dans : AIP Conference Proceedings 248, Computer-aided Statistical Physics.* – Taipei, Taiwan, 1991.

[67] Hakim (V.) et Nadal (J.P.). – Exact result for 2D directed lattice animals on a strip of finite width. *J. Phys. A: Math. Gen.*, vol. 16, 1983, pp. L213–L218.

[68] Hickey (T.) et Cohen (J.). – Uniform random generation of strings in a context-free language. *SIAM. J. Comput.*, vol. 12, n° 4, 1983, pp. 645–655.

[69] Hochstättler (W.), Loebl (M.) et Moll (C.). – Generating convex polyominoes at random. *Dans : Actes du 5ème Colloque Séries Formelles et Combinatoire Algébrique*, éd. par Barlotti (A.), Delest (M.) et Pinzani (R.), pp. 267–278. – Université de Florence, 1993.

[70] Joyal (A.). – Une théorie combinatoire des séries formelles. *Advances in Math.*, vol. 42, n° 1, 1981, pp. 1–82.

[71] Kemp (R.). – Fundamentals of the average cas analysis of particular algorithms, 1984.

[72] Lin (K.Y.) et Chang (S.J.). – Rigourous results for the number of convex polygons on the square and honeycomb lattices. *J. Phys. A: Math. Gen.*, vol. 21, 1988, pp. 2635–2642.

[73] Mairson (H.G.). – Generating words in a context free language uniformly at random. *Information Processing Letters*, vol. 49, 1994, pp. 95–99.

[74] McKay (B.D.) et Wormald (N. C.). – Uniform generation of random regular graphs of moderate degree. *Journal of Algorithms*, vol. 11, 1990, pp. 52–67.

[75] Mezei (J.) et Wright (J.B.). – Algebraic automata and context-free sets. *Information and control*, vol. 11, 1967, pp. 3–29.

[76] Nadal (J.P.), Derrida (B.) et Vannimenus (J.). – Directed lattice animals in 2 dimensions : numerical and exact results. *J. Physique*, vol. 43, 1982, p. 1561.

[77] Nijenhuis (A.) et Wilf (H.S.). – *Combinatorial algorithms.* – Academic Press Inc., 1979.

[78] Penaud (J.G.). – Matrice de ramification des arbres binaires. *Discrete Applied Math.*, vol. 31, 1991, pp. 1–21.

[79] Prellberg (T.) et Brak (R.). – Critical exponents from non-linear functional equations for partially direct cluster models. *preprint, Université de Melbourne, Australie*, mai 1994.

[80] Rémy (J. L.). – Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *R. AI. R. O Informatique Théorique*, vol. 19, n° 2, 1985, pp. 179–195.

[81] Reutenauer (C.). – A survey on noncommutative rational series. *Dans : Actes du 6ème Colloque Séries Formelles et Combinatoire Algébrique*, pp. 393–401. – DIMACS, Université de Rutgers, USA, 1994.

[82] Roblet (E.). – *Une interprétation combinatoire des approximants de Padé.* – Thèse, Université Bordeaux I, avril 1994.

[83] Rouillon (N.). – *Calcul et Image en Combinatoire.* – Thèse, Université Bordeaux I, décembre 1994.

[84] Salomaa (A.) et Soittola (M.). – *Automata - theoretic aspects of formal power series.* – Springer Verlag, 1978.

[85] Salvy (B.). – *Asymptotique automatique et fonctions génératrices.* – Thèse, Ecole Polytechnique de Palaiseau, avril 1991.

[86] Schützenberger (M.P.). – Certain elementary families of automata. *Proc. Symp. on Mathematical Theory of Automata*, 1962, pp. 139–153. – Polytechnic Institute of Brooklyn.

[87] Schützenberger (M.P.). – Context-free langages and pushdown automata. *Information and Control*, vol. 6, 1963, pp. 246–264.

[88] Tutte (W.T.). – A census of planar maps. *Can. J. Math.*, vol. 15, 1963, pp. 249–271.

[89] Viennot (X.G.). – *Une Théorie Combinatoire des Polynômes Orthogonaux Généraux.* – Montréal, Notes de conférences données à l'UQAM, 1983.

[90] Viennot (X.G.). – A combinatorial theory for general orthogonal polynomials with extensions and applications. *Dans : Polynômes orthogonaux et leurs applications*, éd. par Brezinski (C.), Draux (A.), Magnus (A.P.), Maroni (P.) et A.Ronveaux, pp. 139–157. – Lecture Notes in Math. 1171, 1985.

[91] Viennot (X.G.). – Enumerative combinatorics and algebraic laguanges. *Dans : Proceedings FCT'85*, éd. par Budach (L.), pp. 450–464. – Lecture Notes in Computer Science 199, 1985.

[92] Viennot (X.G.). – Problèmes combinatoires posés par la physique statistique. *Astérisque*, vol. 121-122, 1985, pp. 225–246. – Séminaire Bourbaki 626, 26ème année.

[93] Viennot (X.G.). – A survey of polyominoes enumeration. *Dans : Actes du 4ème Colloque Séries Formelles et Combinatoire Algébrique*, éd. par Leroux (P.) et Reutenauer (C.), pp. 399–420. – Université du Québec à Montréal, 1992. Publications du LaCIM 11.

[94] Wacrenier (L.). – *Génération aléatoire et uniforme de mots d'un langage.* – Université Paris Sud, Rapport de stage de DEA au LRI Orsay, Septembre 1992.

[95] Wechler (W.). – *Universal algebra for computer scientists.* – Springer, 1992.

[96] Wilf (H.S.). – *Generatingfunctionology.* – Londres, Academic Press, 1994, inc. harcourt brace & company édition.

[97] Wormald (N.C.). – Generating random regular graphs. *J. Algorithms*, vol. 5, 1984, pp. 247–280.

[98] Zimmermann (P.). – *Séries génératrices et analyse automatique d'algorithmes.* – Thèse, Ecole Polytechnique de Palaiseau, mars 1991.

[99] Zimmermann (P.). – Gaïa: a package for the random generation of combinatorial structures. *Maple Technical Newsletter*, vol. 1, n° 1, 1994, pp. 38–46.

Grammaires d'objets : énumération, bijections et génération aléatoire

Résumé : Cette thèse aborde divers problèmes du domaine de la combinatoire énumérative et bijective. Elle concerne la formalisation de la notion de *grammaires d'objets* - descriptions récursives d'objets combinatoires exprimées sous forme de dessins - ainsi que l'étude de différents domaines d'application : énumération, constructions de bijections, et génération aléatoire uniforme.

En appliquant des *valuations d'objets linéaires* (ou *q-linéaires*) aux grammaires d'objets, nous pouvons déduire directement le système d'équations algébriques (ou de q-équations) vérifié par la série génératrice des objets engendrés.

Dans le cadre des bijections, notre théorème central propose un élément de réponse à une constatation bien connue : "Pourquoi les mots de Dyck jouent-ils un rôle si important dans le codage des objets combinatoires ?". En effet, il exhibe la construction de bijections entre les mots de Dyck et *tout* ensemble d'objets engendré par une grammaire d'objets de dimension 1 et de degré ≥ 2.

Notre contribution dans le domaine de la génération aléatoire d'objets combinatoires est une approche systématique de la méthode récursive pour des objets engendrés par des grammaires d'objets auxquelles sont appliquées des valuations linéaires ou q-linéaires. La génération selon des paramètres non algébriques est ainsi rendue accessible. Ce travail est implémenté en *Maple*.

Mots-clés : grammaires d'objets, valuations d'objets, énumération, bijections, mots de Dyck, génération aléatoire uniforme.