Je tiens à remercier ici les personnes qui m'ont soutenue et aidée tout au long de ce mémoire.

M. Penaud responsable de mon mémoire, pour sa disponibilité et l'enthousiasme dont il a fait preuve à l'égard de mon travail.

M. Plouffe pour tout le temps qu'il m'a aimablement accordé lors de son séjour à Bordeaux en février.

M. Cohen pour avoir bien voulu me fournir quelques explications sur l'utilisation du logiciel PARI.

M. Mélançon pour ses précieux conseils lors de mes premiers pas avec MAPLE.

Emmanuel Roblet pour toutes les claires explications qu'il m'a apportées et qui m'ont bien aidé pour l'implémentation de sa méthode.

Sommaire

Introduction

Les problèmes abordés dans ce mémoire sont du domaine de la **"combinatoire énumérative"**.

Les objets manipulés en combinatoire sont des objets discrets tels que les arbres, les graphes ou les mots d'un langage. Une façon de les connaître est de les énumérer, soit directement, soit par le biais de bijections avec des familles d'objets dont l'énumération est déjà connue, ou alors plus aisée.
La deuxième voie est de plus en plus prisée par les chercheurs, car elle permet d'éviter de fastidieux calculs techniques.

On retrouve de nombreuses applications de la combinatoire en physique théorique. A ce propos, Viennot a donné dans [11] un large éventail des rapports entre la *combinatoire* et la *physique statistique*, notamment en ce qui concerne les animaux dirigés.

Les animaux dirigés dans le plan combinatoire à deux dimensions sont des configurations de points deux à deux voisins sur une grille régulière à maille carrée ou triangulaire. La direction privilégiée donne la direction vers laquelle peut s'étendre l'animal par ajout de nouveaux points.

On étend la notion d'animal dirigé dans l'espace à trois dimensions en plaçant les configurations de points sur des réseaux à maille cubique.

Ce modèle des animaux dirigés dans le plan a été introduit en 1982. Il présente un grand intérêt pour les physiciens car il permet de rendre compte des phénomènes critiques et de changements de phase, tels que l'ébullition de l'eau ou l'aimantation.
En ce qui concerne la percolation dirigée, les paramètres étudiés sur les animaux dirigés sont l'aire (nombre de points), le périmètre, la longueur et la largeur.

Nous nous sommes ici attachés à l'étude des **animaux dirigés dans l'espace 3D**.

Si déjà beaucoup de résultats sont connus pour les animaux 2D (cf [12] et [8]), très peu le sont pour les animaux 3D.
Seul le problème d'énumération d'un certain type d'animal 3D (associé au modèle des hexagones durs) a été récemment résolu par Baxter [1], mais par des méthodes analy-

tiques et physiques.
De plus, les deux autres modèles, les carrés durs et les triangles durs, échappent à ces méthodes.

Le but des combinatoriciens est alors de trouver, pour les différents types d'animaux 3D, une solution commune au problème de leur énumération, c'est à dire calculer leur série génératrice, ou tout au moins une approximation asymptotique de celle-ci.

Le mémoire présenté ici se compose donc de deux parties :

1. *L'énumération des animaux dirigés dans l'espace 3D.*

   Un algorithme est mis en œuvre pour calculer les premiers coefficients des séries génératrices des différents types d'animaux 3D.

2. *Les approximations de séries génératrices.*

   Différentes méthodes numériques sont étudiées dans le but de les appliquer aux séries de nombres que l'on a calculées grâce à notre algorithme.

Partie I

L'énumération des animaux dirigés dans l'espace 3D

Chapitre 1

Définitions sur les animaux dirigés 2D

Un animal dirigé est une configuration de points du plan combinatoire $\prod = \mathrm{Z} \times \mathrm{Z}$ se développant dans la direction de la première bissectrice.

Définition 1.1 *Un animal dirigé $\mathcal{A}$ à une source est un ensemble de points de $\prod$ tel que:*

- *l'origine $O = (0,0)$ appartient à $\mathcal{A}$; c'est la source de $\mathcal{A}$,*
- *tout point (x,y) de $\mathcal{A}$ peut être atteint depuis la source par un chemin dont tous les sommets sont dans $\mathcal{A}$ et dont les pas élémentaires sont soit Est, soit Nord, soit Nord-Est.*

Si les pas autorisés sont réduits aux pas Nord et Est, on dit alors que l'on a un animal sur réseau carré; si l'on autorise de plus le pas Nord-Est (diagonal), le réseau est dit triangulaire. La direction Nord-Est est dite *direction privilégiée* de l'animal.

On définit les notions de *longueur* et *largeur* de l'animal : ce sont les dimensions du plus petit rectangle qui le contienne, dont les côtés sont parallèles à la direction privilégiée (longueur) et à sa perpendiculaire (largeur).

La première bissectrice partage ce rectangle en deux parties, définissant ainsi les *demi-largeurs supérieure* et *inférieure.*

L'unité de mesure est le demi-pas diagonal.

On peut généraliser au cas des animaux dirigés à plusieurs sources, et en particulier à sources compactes, qui sont placées sur une perpendiculaire à la direction privilégiée.

Définition 1.2 *Un animal dirigé à k sources compactes est un ensemble de points que l'on peut atteindre par des chemins ayant comme origines les point O_s de coordonnées $(-s,s)$ pour $0 \leq s \leq k-1$.*

Un animal dirigé à 4 sources compactes sur réseau carré est présenté dans la figure 1.1 ci-dessous. Il contient 35 points, sa largeur est 10 et sa longueur est 11.

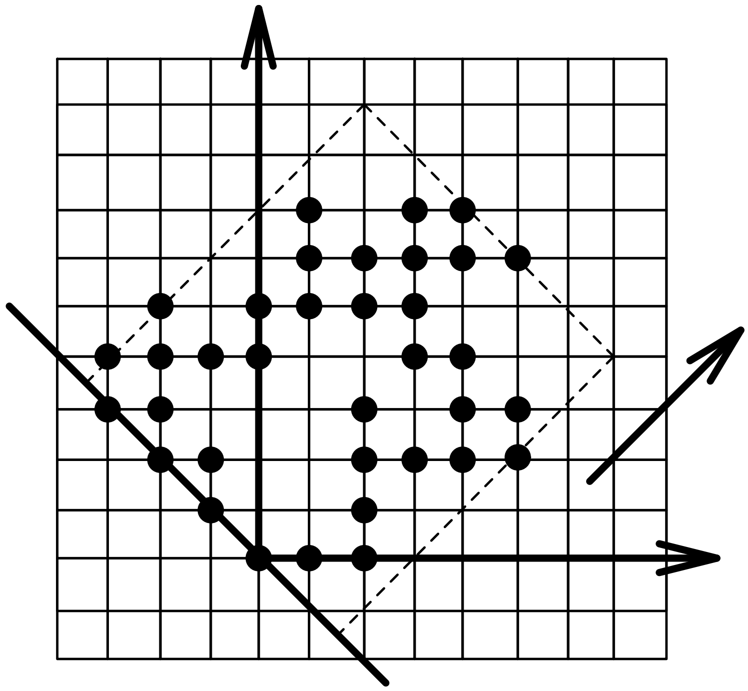


Figure 1.1: Un animal dirigé sur réseau carré.

Convention :

Afin de rendre plus agréable la manipulation des animaux dirigés, on effectue un changement de repère.

On considère que la direction privilégiée devient la direction Sud et que les pas alors autorisés sont Sud-Ouest, Sud-est et Sud.

Les points de l'animal ont alors pour coordonnées les couples $(i + 1/2, j)$.

Ainsi, la figure 1.2 ci-dessous représente l'animal de la figure 1.1 dans le nouveau repère.

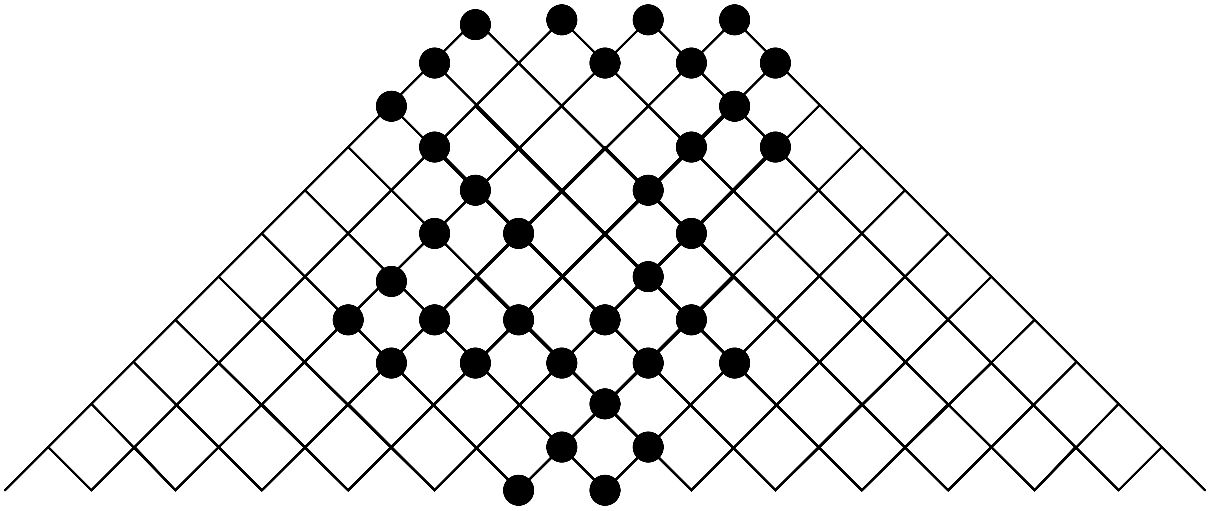


Figure 1.2: Le même animal dans le nouveau repère.

Chapitre 2

Notions sur les empilements de dominos

La notion d'empilement de dominos a été introduite par X.G.Viennot dans le cadre plus général des empilements de pièces.

On donne ici quelques définitions permettant de comprendre les résultats liant les empilements de dominos aux animaux dirigés.

2.1 Les systèmes de p-dominos

On considère le plan combinatoire $\prod = \mathrm{Z} \times \mathrm{Z}$.

Définition 2.1 *Un domino placé (ou p-domino) est un couple de points (i, j) et $(i+1, j)$ du plan $\prod$.*

Soit δ un p-domino ; δ est noté $< i, j >$.
L'entier i est son *abscisse*, $abs(\delta)$, et l'entier j est son *niveau*, $h(\delta)$.
Le couple $(i, i+1)$ est la *projection horizontale* de δ, notée $\pi(\delta)$.

Définition 2.2 *Un système de p-dominos est un ensemble fini de p-dominos deux à deux disjoints.*

Un exemple d'un tel système est donné figure 2.1.

L'abscisse d'un système est l'abscisse minimale des p-dominos, et sa hauteur, la hauteur minimale des p-dominos.

Définition 2.3 *Soient δ et δ' deux p-dominos d'un système γ.*
On dit que δ domine δ' si

$$h(\delta) > h(\delta') \ et \ \pi(\delta) \cap \pi(\delta') \neq \emptyset \ .$$

La fermeture réflexive et transitive de cette relation est appelée "*la relation de dominance*" et est notée $\geq$.
L'ensemble des p-dominos de γ que domine δ est noté $dom(\delta)$.

Définition 2.4 *Un p-domino δ de γ est dit <u>maximal</u> si γ ne contient aucun autre p-domino qui domine δ.*

De la même façon, un p-domino δ de γ est dit <u>minimal</u> si γ ne contient aucun autre p-domino qui n'est dominé par δ.

Sur la figure 2.1, les p-dominos maximaux sont noirs et les minimaux sont blancs.
Un système n'ayant qu'un élément maximal est appelé "*pyramide*".

Définition 2.5 *On définit la <u>demi-largeur gauche</u> de γ comme la différence entre l'abscisse d'un élément maximal de γ d'abscisse minimale (la plus à gauche) et celle de γ.*

La <u>demi-largeur droite</u> est la différence entre l'abscisse maximale des p-dominos de γ et l'abscisse d'un élément maximal de γ d'abscisse maximale (la plus à droite).

Le système de la figure 2.1 a pour demi-largeur gauche x et demi-largeur droite x.

La numérotation canonique :

L'algorithme : On considère un système γ de n p-dominos.
Soit δ le p-domino maximal d'abscisse minimale de γ ; on le numérote 1.
On retire δ de γ, on a donc le système $\gamma_1 = \gamma \backslash \{\delta\}$.
On numérote alors 2 le p-domino maximal d'abscisse minimale de γ_1.
Et ainsi de suite, en réitérant ce procédé, on obtient une numérotation de tous les p-dominos de γ par les entiers de 1 à n.

La propriété : On note $num(\delta)$ le numéro de δ.

$$num(\delta) < num(\delta') \Longleftrightarrow \begin{cases} \delta \text{ domine } \delta' \\ ou \\ \delta \text{ et } \delta' \text{ sont incomparables et } abs(\delta) < abs(\delta') \end{cases}$$

Un exemple d'une telle numérotation est exhibé sur la figure 2.1 ci-dessous.

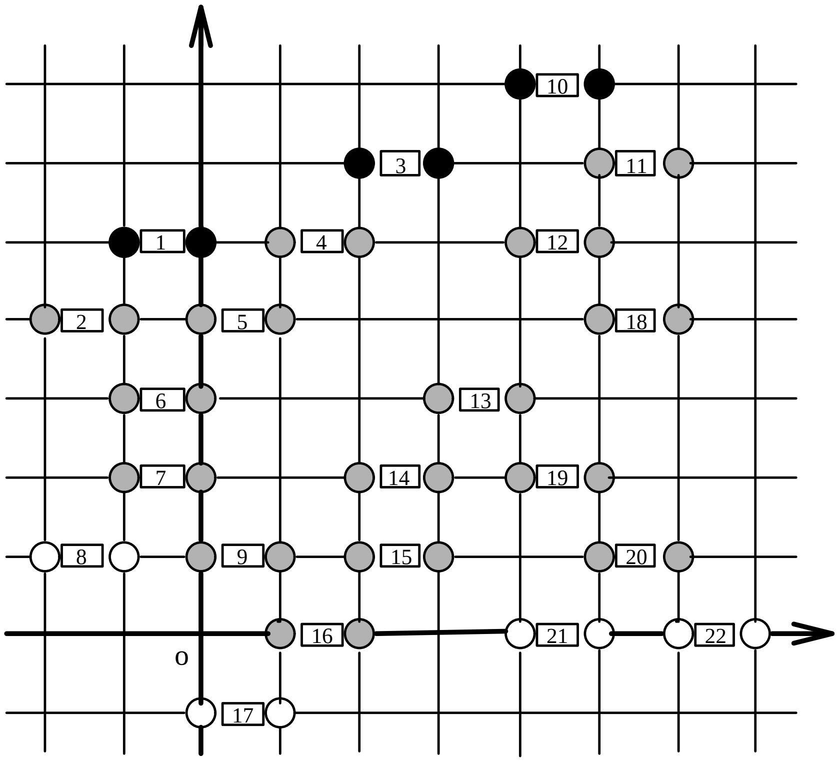


Figure 2.1: Un système de p-dominos.

2.2 Systèmes équivalents et empilements

Soit γ un système de p-dominos.

On dispose de trois transformations qui appliquées à γ donnent un système équivalent.

La translation $\mathrm{T}_{l,m}(\gamma)$ qui opère sur chaque p-domino de γ la translation de vecteur (l, m).

La déformation $\mathrm{D}(\gamma, \delta, k)$ qui opère sur un seul p-domino δ une translation verticale de vecteur $(0, k)$: k est tel que δ n'intersecte aucun autre p-domino lors de son déplacement.

La réduction qui est une déformation particulière $\mathrm{D}(\gamma, \delta, k)$: elle est telle que δ n'est pas de niveau minimal, et k est strictement négatif.

Ces trois opérations induisent des relations d'équivalence.

Remarque : on ne donne ici qu'une définition succincte des transformations. Le lecteur pourra trouver les définitions complètes dans [9].

Proposition 2.1 *Chaque classe d'équivalence $[\gamma]_r$ contient un élément irréductible unique pour la relation de réduction r. On l'appellera* <u>*empilement de p-dominos.*</u>

La proposition suivante donne une condition nécessaire et suffisante pour qu'un système γ soit un empilement.

Proposition 2.2 *Un système de p-dominos de $\prod$ de niveau minimal k_0 est un empilement de p-dominos si et seulement si*

1. *les p-dominos sont 2 à 2 disjoints,*

2. *si un p-domino δ est au niveau $k > k_0$, il repose sur un p-domino de niveau $k - 1$.*

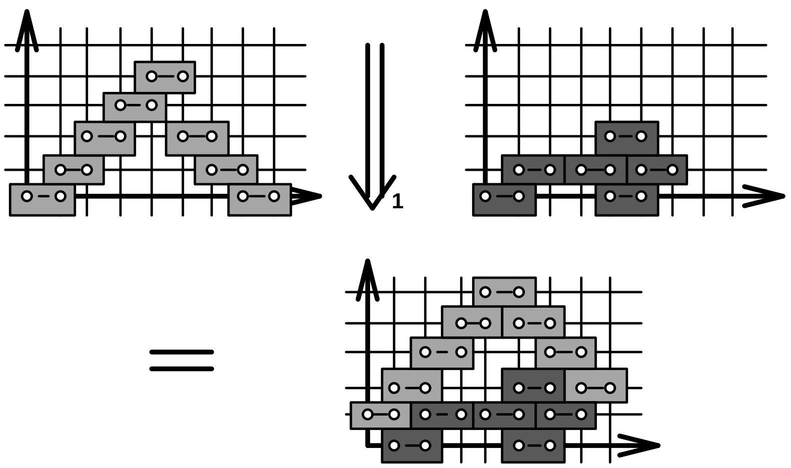
Parmi les empilements de p-dominos, on distingue ceux d'abscisse et de niveau nuls (systèmes équivalents qui peuvent être obtenus par translation).
On les appelle **empilements de dominos**.

Remarque : toutes les défintions données en 2.1 sur les systèmes de p-dominos s'étendent de manière naturelle aux empilements de dominos.

Deux opérations élémentaires sont définies sur les empilements. (on donne ici l'idée intuitive qui servira au chapitre 3 pour le codage des animaux)

Le glissement d'une sous-pyramide qui consiste à "baisser" la sous-pyramide contenant un p-domino δ et $dom(\delta)$.

La m-superposition qui consiste à "laisser choir" un système γ_1 sur un autre γ_2 qui a éventuellement subi une translation horizontale.
On note $\gamma_1 \Downarrow_m \gamma_2$ où m signifie que γ_2 a été translaté horizontalement de m.



2.3 Bijection entre les empilements et les animaux dirigés

Théorème 2.1 (X.G. Viennot) *Les animaux à source compacte ayant n points sont en bijection avec les empilements dont les dominos maximaux sont à projection consécutive et ayant n dominos.*

Remarque : la démonstration de ce théorème se fait par récurrence sur le nombre de points de l'animal.

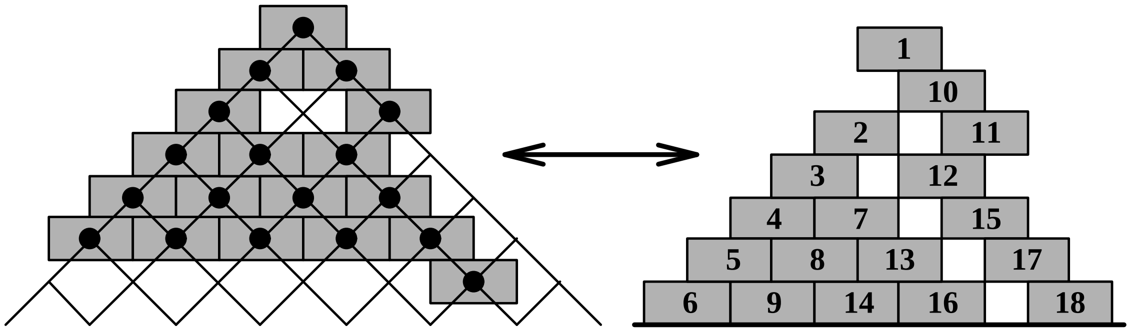


Figure 2.2: Un animal et l'empilement correspondant.

A chaque point de l'animal, on associe un domino; l'empilement correspondant est obtenu en laissant "choir" les dominos.

La numérotation canonique de l'animal est donc immédiate :

le numéro d'un point de l'animal est le numéro du domino correspondant dans l'empilement associé à l'animal.

Une conséquence importante :

cette numérotation canonique de l'animal permet la définition d'une nouvelle famille d'arbres asymétriques, "*les arbres guingois*", en bijection avec les animaux dirigés à 1 source.

Chapitre 3

Le codage des animaux

On rappelle ici le codage des animaux dirigés à 1 source établi par Penaud dans [9].
Ce codage s'inspire de la bijection avec les empilements ; et on le verra au chapitre 5, la même idée de codage est utilisée pour les animaux 3D.

Le codage :

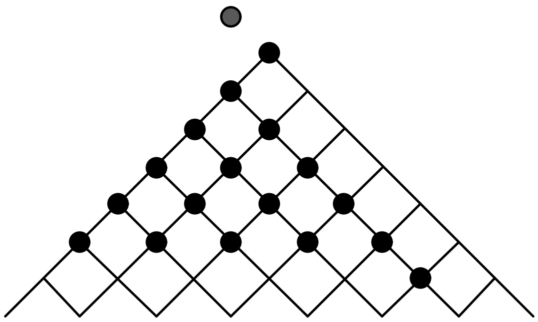
Les animaux dirigés sur réseau carré à 1 source sont codés à l'aide de 3 opérateurs, nommés a, b, et c.
Les animaux sur réseau triangulaire utilisent un quatrième opérateur d.

Le codage s'effectue par construction de l'animal à l'aide des quatre opérateurs qui ajoutent chacun un point à un animal à 1 source.
On donne ici un brève description de ces opérateurs :

Soit $\mathcal{A}$ un animal à 1 source et n points ($n \geq 1$).

L'opérateur a

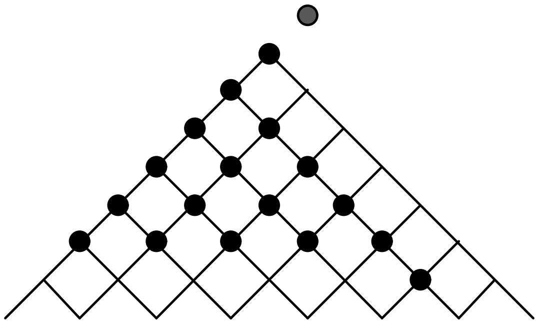
L'animal $\mathcal{A}' = a(\mathcal{A})$ est obtenu en ajoutant 1 point au Nord-Ouest de la source de $\mathcal{A}$. Ce point devient alors la nouvelle source de $\mathcal{A}'$.



L'opérateur b

Pour l'unicité du codage, cet opérateur n'est défini que si $\mathcal{A}$ a une demi-largeur droite non nulle.

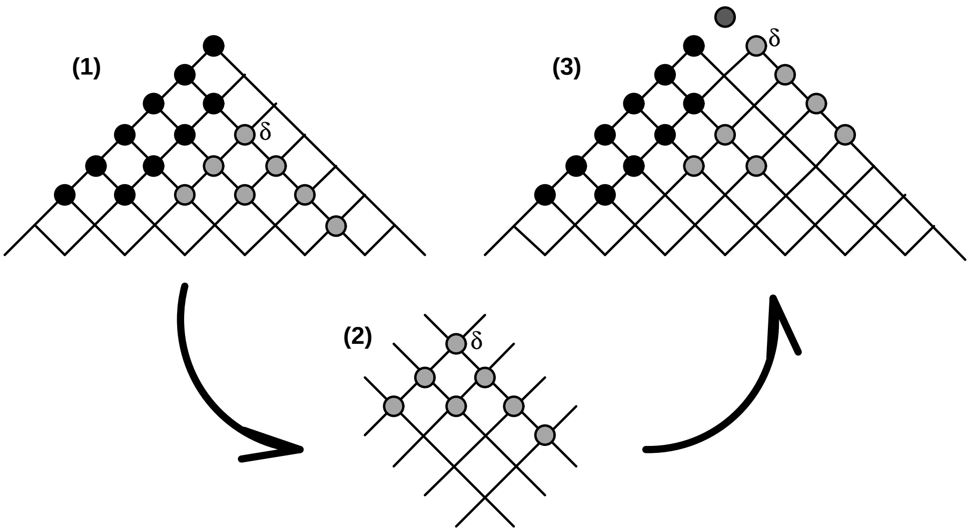
L'animal $\mathcal{A}' = b(\mathcal{A})$ est obtenu en ajoutant 1 point au Nord-Est de la source de $\mathcal{A}$. De la même façon que précédemment, il devient la nouvelle source de $\mathcal{A}'$.



L'opérateur c

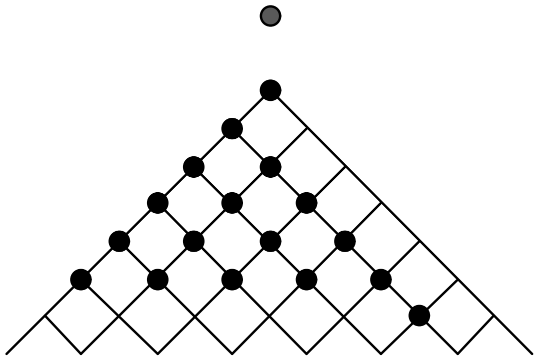
L'animal $\mathcal{A}' = c(\mathcal{A})$ est obtenu de la manière suivante :

- (1) on choisit le point d'abscisse $abs(source) + 1$ de hauteur maximale. Soit δ ce point.
- (2) on "pousse" sur δ pour obtenir $dom(\delta)$ et on ajoute 1 à l'abscisse de tous les points de $dom(\delta)$.
- (3) on ajoute un point au Nord-Est de la source et on laisse "remonter" $dom(\delta)$.



L'opérateur d

L'animal $\mathcal{A}' = d\,(\mathcal{A})$ est obtenu en ajoutant 1 point au Nord de la source de $\mathcal{A}$, qui devient la nouvelle source.



Donc pour tout animal de taille n à 1 source, l'application d'un des quatre opérateurs lui associe un animal unique de taille $n+1$.

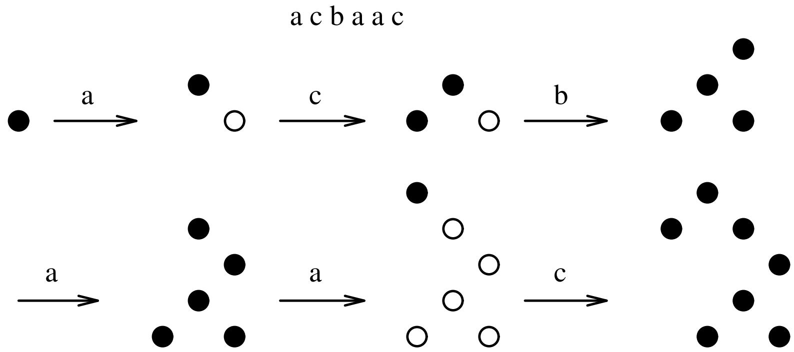
D'où le théorème fondamental :

Théorème 3.1 *Tout animal de taille $n+1$ ($n \geq 0$) à 1 source sur réseau carré (resp. triangulaire) s'obtient de façon unique comme résultat de l'application d'un produit de composition de n opérateurs pris parmi a, b, c, (resp. d).*

Et on a alors la proposition suivante :

Proposition 3.1 *A tout animal de taille $n+1$ à 1 source sur réseau carré (resp. triangulaire), on peut associer un mot unique de longueur n sur $\{a, b, c\}$ (resp. $\{a, b, c, d\}$). Ce mot est un facteur gauche de Motzkin (resp. bicoloré).*

Un exemple de construction de l'animal à partir du mot de Motzkin :



Chapitre 4

Les arbres guingois et les animaux dirigés

Les **arbres guingois** sont une famille d'arbres binaires définie par Bétréma et Penaud dans [4]. Les auteurs ont montré que ces arbres sont en bijection avec les animaux dirigés à une source sur réseau carré.

4.1 Définition des arbres guingois

On définit tout d'abord un paramètre qui mesure l'écart entre les projections horizontales d'un sommet et celle de la racine.

A tout sommet x d'un arbre binaire de racine z, on associe un entier relatif appelé *déport de x par rapport à z* et noté $dep_z(x)$.

Définition 4.1 *$dep_z(x)$ est la différence entre le nombre de pas à droite et le nombre de pas à gauche dans l'unique chemin de z à x dans l'arbre.*

Le déport d'un arbre binaire T est alors défini par

$$dep(T) = \max_{x \in T} dep_z(x) \ .$$

Remarque : $dep_z(x)$ est en fait la projection horizontale de x, si celle de z est 0.

<u>Exemple :</u>

Sur la figure 4.1 est représenté un arbre binaire, avec l'axe horizontal des projections où on lit directement les déports des sommets.
Ainsi, $dep_z(x) = -3$ et le déport de l'arbre est 2.

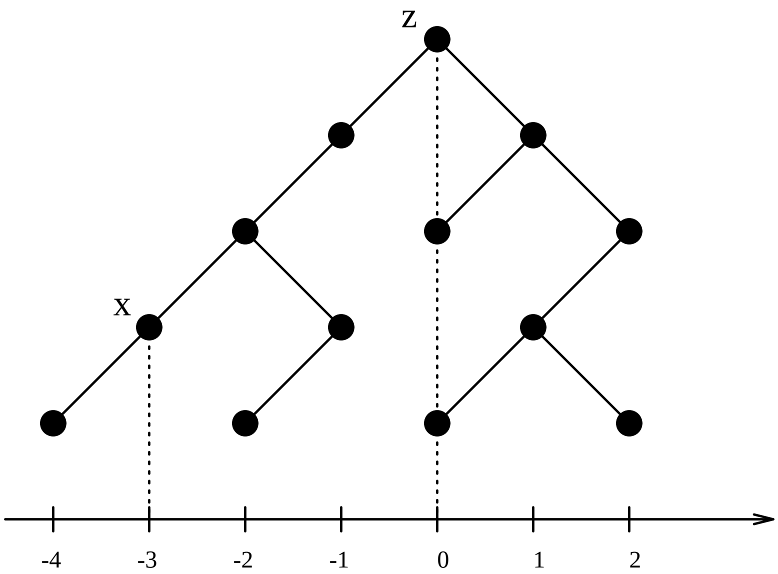


Figure 4.1: Un arbre guingois.

On appelle *équerre* un arbre binaire de déport négatif ou nul.
Une équerre est dite *stricte* si le dernier sommet dans l'ordre préfixe est de déport nul.

Définition 4.2 *Un arbre guingois est un arbre binaire tel que pour tout point double, le sous-arbre gauche est une équerre.*

Un tel arbre est présenté figure 4.1.

4.2 Grammaire des arbres guingois

On présente figure 4.2 une illustration de la grammaire permettant d'engendrer les arbres guingois.

Toutes les précisions sur l'établissement de cette grammaire et ses propriétés sont données dans [4].

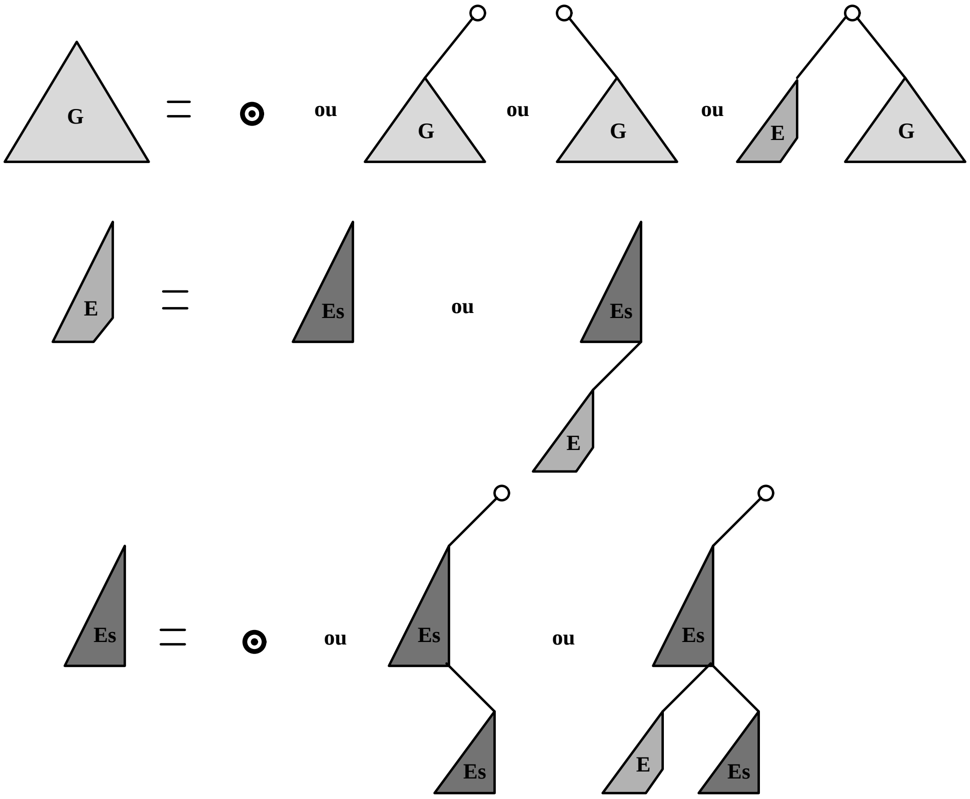


Figure 4.2: La génération des arbres guingois.

De cette décomposition des arbres guingois, on déduit les équations du langage qui les code :

$$\begin{cases} G = \varepsilon + a\,G + b\,G + x\,E\,\overline{x}\,G \\ E = E_s + E_s\,a\,E \\ E_s = \varepsilon + a\,E_s\,b\,E_s + a\,E_s\,x\,E\,\overline{x}\,E_s \end{cases}$$

Penaud et Bétréma montrent alors :

Théorème 4.1 *Il y a bijection entre le langage codant les arbres guingois G et celui codant les facteurs gauches de Motzkin avec conservation des longueurs.*

4.3 La bijection avec les animaux dirigés (Bétréma/Penaud)

A tout animal dirigé $\mathcal{A}$, on peut associer un graphe orienté binaire $T_{\mathcal{A}}$.

Si on considère le graphe de la relation de dominance directe, on peut définir $T_{\mathcal{A}}$ comme son arbre recouvrant en privilégiant le fils gauche.

Définition 4.3 *Le* <u>*graphe* $T_{\mathcal{A}}$</u> *est défini comme suit :*

1. *les sommets de* $T_{\mathcal{A}}$ *sont les points de l'animal* $\mathcal{A}$,
2. y *est fils gauche de* x *dans* $T_{\mathcal{A}}$ *si*

$$\begin{gathered} abs(y) = abs(x) - 1 \\ et \\ il\ n'existe\ pas\ z \in \mathcal{A}\ tel\ que \\ abs(z) = abs(y)\ ou\ abs(z) = abs(x)\ et\ h(y) < h(z) < h(x), \end{gathered}$$

3. y *est fils droit de* x *dans* $T_{\mathcal{A}}$ *si*

$$\begin{gathered} abs(y) = abs(x) + 1\ ,\ h(y) = h(x) - 1 \\ et \\ y\ n'est\ fils\ gauche\ d'aucun\ point\ de\ \mathcal{A}. \end{gathered}$$

Une propriété importante est que tout point de $\mathcal{A}$ possède un prédécesseur unique dans $T_{\mathcal{A}}$.

Dans [4], Penaud et Bétréma ont alors montré la proposition suivante :

Proposition 4.1 *Pour tout animal* $\mathcal{A}$, *le graphe* $T_{\mathcal{A}}$ *est un arbre guingois ayant pour racine la source de* $\mathcal{A}$.

La figure 4.3 ci-dessous illustre l'association animal dirigé et arbre guingois.

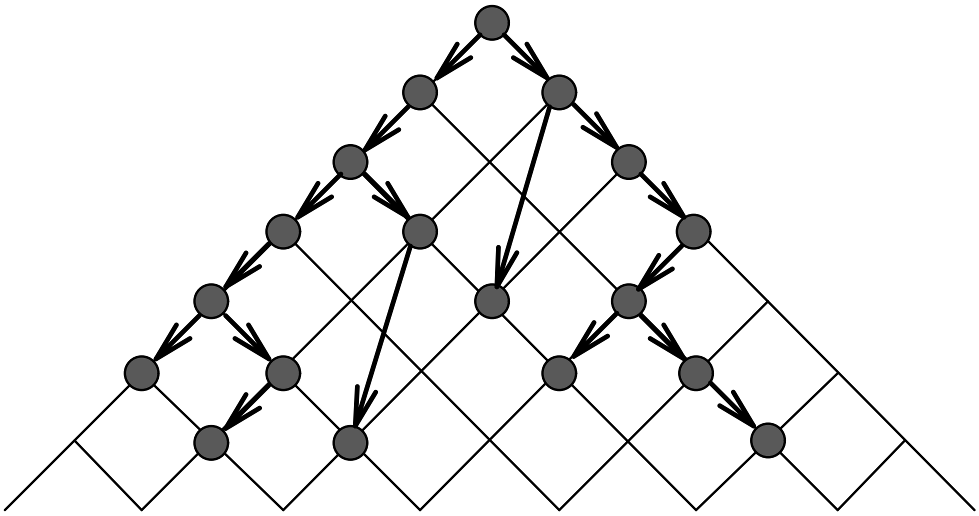


Figure 4.3: Un animal avec l'arbre guingois associé.

Réciproquement : on montre qu'à tout arbre guingois, on peut associer un animal dirigé.

On a alors le théorème suivant :

Théorème 4.2 *Les animaux dirigés à* 1 *source et* n *points sont en bijection avec les arbres guingois à* n *sommets.*

Corollaire 4.1 *Les animaux dirigés à* 1 *source et* n *points sont codés avec les facteurs gauches de Motzkin de longueur* $n-1$.

Le mot codant l'animal de la figure 4.3 est donc le suivant :

$$x\,a\,x\,a\,x\,\overline{x}\,a\,\overline{x}\,a\,\overline{x}\,x\,\overline{x}\,b\,a\,x\,\overline{x}\,b\ .$$

Remarque : ce codage est le codage de l'arbre binaire.

- x pour un sommet double,
- a pour un fils gauche,
- b pour un fils droit,
- $\overline{x}$ pour une feuille.

Chapitre 5

Les animaux dirigés 3D

On se place dans l'espace combinatoire à 3 dimensions $E = Z \times Z \times Z$, pavé par des cubes unitaires.
On étend alors aisément les notions d'animal et d'empilements de pièces.

On considère 2 types de réseaux : cubique et cubique centré.

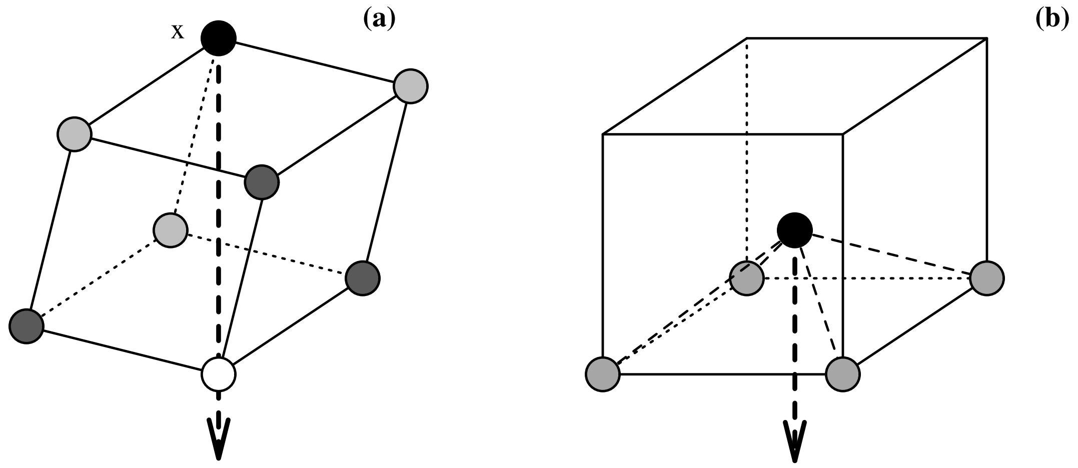


Figure 5.1: Les 2 types de réseaux en 3 dimensions.

Réseau cubique (a): les points sont placés sur les sommets des cubes du pavage. La direction privilégiée est représentée sur la figure 5.1 vers le sud.

Réseau cubique centré (b): en plus des sommets des cubes, on autorise un point au centre des cubes.

5.1 Définition des animaux 3D

On peut alors définir 3 types d'animaux suivant la façon de placer les points et de définir les voisinages.

- Les animaux sur réseau cubique : figure 5.1 (a)
Les pas autorisés d'un point x de l'animal à ses voisins sont

- les pas selon les arêtes incidentes en x.

(3 voisins en gris clair)

- les pas selon la diagonale d'une face, lorsque x n'a pas de voisin selon les arêtes de cette face.

(3 voisins en gris foncé)

- éventuellement (cas large), un pas selon la diagonale principale du cube.

(1 voisin en blanc)

- Les animaux sur réseau cubique alterné : figure 5.1 (a)
Les mêmes pas que précédemment sont autorisés, mais les pas selon les arêtes issues de x et les pas selon les diagonales des faces incidentes à x ne sont utilisés qu'alternativement, selon la parité de la distance à la source.

- Les animaux sur réseau cubique centré : figure 5.1 (b)
Les pas autorisés sont

- les pas selon les 4 demi-diagonales.

(4 voisins en gris clair)

- éventuellement (cas large), un pas selon la direction privilégiée vers le centre du cube contigu, seulement si les pas précédents ne sont pas utilisés.

Il est clair géométriquement que si l'on projette les points d'un animal sur un plan perpendiculaire à la direction privilégiée, on obtient des points

– sur réseau triangulaire pour un animal sur réseau cubique,
– sur réseau hexagonal pour un animal sur réseau cubique alterné,
– sur réseau carré pour un animal sur réseau cubique centré,

les points voisins de l'animal étant reliés par une arête du réseau du plan de projection (ou superposés dans le cas large).

5.2 Les empilements de "carreaux"

On considère des empilements de pièces identiques; ces pièces sont des "carreaux" en forme d'**hexagones**, de **triangles** ou de **carrés**.
Leurs centres se projettent en formant respectivement un réseau régulier triangulaire, hexagonal, ou carré.

On ne s'intéresse ici qu'aux pyramides de pièces.
Voici donc quelques propriétés de ces pyramides :

- Une pyramide est dite **stricte** si aucune de ses pièces n'a la même projection sur le réseau considéré qu'une pièce qu'elle domine directement, c'est à dire qu'aucune pièce n'est exactement superposée sur une autre.
- Une pyramide est dite **primitive** si aucune de ses pièces n'a la même projection que sa pièce maximale.
- Si l'on n'exige pas les contraintes précédentes, la pyramide est dite **large**.

5.3 Codage

Proposition 5.1 *Il y a bijection entre les animaux à* 1 *source et* n *points sur réseau cubique (resp. alterné, centré) et les pyramides de* n *hexagones (resp. triangles, carrés).*

On démontre cette proposition en plaçant les pièces correspondantes centrées sur les points de l'animal.
C'est une généralisation de la bijection entre les animaux dirigés 2D et les empilements de dominos.

Cette bijection entre les animaux dirigés 3D et les empilements de pièces permet de déduire un codage.

Théorème 5.1 *Les animaux à* 1 *source et de taille* n *sont codés par des mots de longueur* $n-1$ *sur un alphabet à* 18 *lettres pour le réseau cubique,* 16 *lettres pour le réseau cubique centré et* 8 *lettres pour le réseau cubique alterné.*

La démonstration de ce théorème est une preuve constructive.

Si on a un animal à 1 source et n points sur un des 3 réseaux, pour déterminer la $n-1^{ième}$ lettre du mot associé, le procédé est le suivant :

1. suppression de la pièce maximale,
2. codage de la configuration définie par les nouvelles pièces maximales,
3. reconstitution si nécessaire d'une pyramide à $n-1$ points par une opération standard, dépendant de la configuration.

La figure 5.2 ci-dessous illustrent toutes les configurations possibles, pour chaque type d'animal, qui correspondent aux lettres du code associé.

Si dans le cas des animaux dirigés 2D, les mots de code ont pu être caractérisés (facteurs gauches de Motzkin pour le réseau carré et bicolorés pour le réseau triangulaire), pour les animaux dirigés 3D, cette caractérisation reste un problème ouvert.

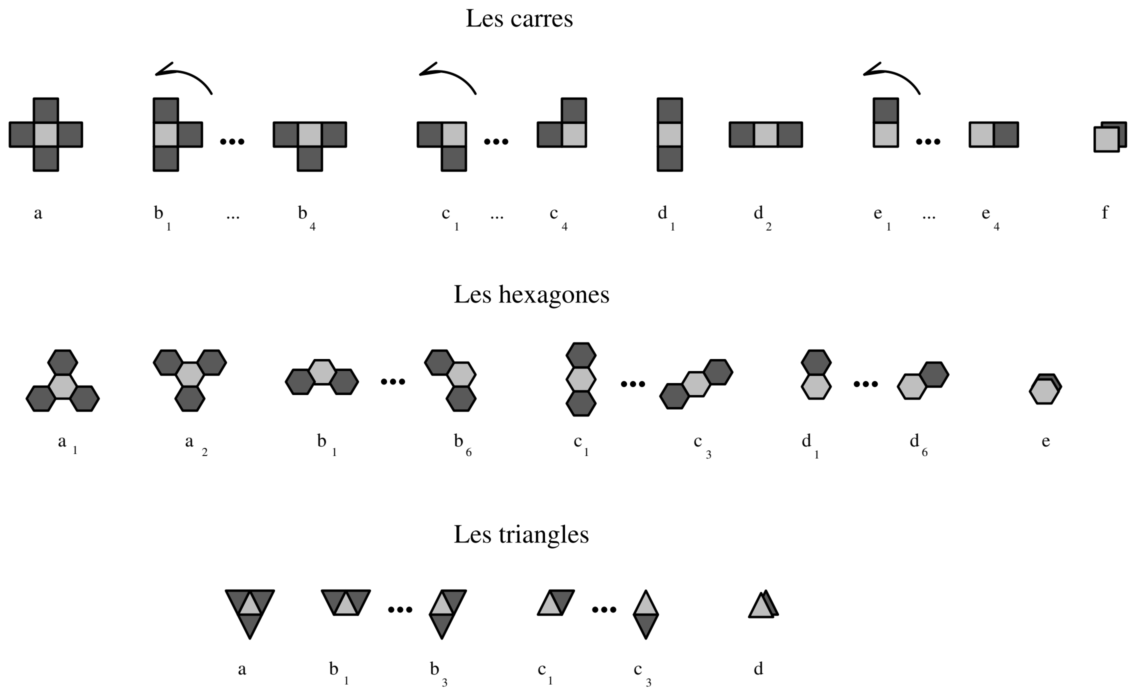


Figure 5.2: Les différentes configurations de pièces.

5.4 Une autre approche : extension des arbres guingois

L'idée est d'étendre la propriété des arbres guingois ("tout point double a une équerre comme sous-arbre gauche") aux arbres p-aires dans l'espace, dont les sommets peuvent avoir jusqu'à p fils. (on se restreint ici au cas où $p = 3$, 4 ou 6)

De la même façon que l'on projette les points d'un arbre binaire sur un axe pour vérifier s'il est guingois, on définit la projection sur un plan des arbres p-aires.
A tout sommet de l'arbre est associé un point du réseau régulier hexagonal ($p = 3$), triangulaire ($p = 6$) ou carré ($p = 4$), et à chaque arc une arête du réseau.

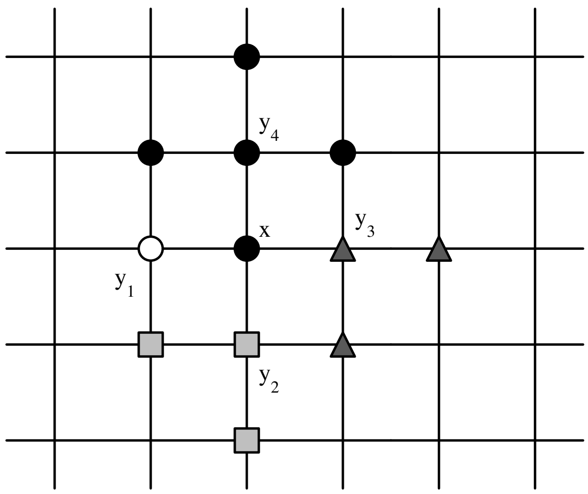
On rappelle que les fils d'un sommet d'un arbre p-aire ont un rang, un entier entre 1 et p.

L'extension de l'arbre guingois en 3 dimensions est donc définie comme suit :

Définition 5.1 *On appelle g-arbre un arbre p-aire qui possède la propriété suivante pour tout sommet x ayant plusieurs fils :*

si $i_1, \ldots, i_k$ sont les rangs des fils de x, alors aucune des projections des sommets du sous-arbre de rang i_r ($1 \leq r \leq k-1$) n'est relié par une arête à la projection des fils de x de rang supérieur.

Exemple :



Cette figure illustre la projection sur réseau carré d'un sommet x d'un g-arbre 4-aire ayant 4 fils : dans l'ordre y_1, y_2, y_3, y_4.

- Il n'y a pas de contrainte sur les projections des sommets du sous-arbre de racine y_4.
- Projections interdites pour les sommets du sous-arbre de racine y_3 : les points noirs.
- Projections interdites pour les sommets du sous-arbre de racine y_2 : les points noirs et les triangles gris foncé.
- Projections interdites pour les sommets du sous-arbre de racine y_1 : les points noirs, les triangles gris foncé et les carrés gris clair.

On déduit alors par récurrence :

Proposition 5.2 *Les animaux dirigés à 1 source et n points sur réseau cubique alterné (resp. centré) sont en bijection avec les g-arbres 3-aires (resp. 4-aires) ayant n sommets.*

Les animaux dirigés à 1 source et n points sur réseau cubique sont en bijection avec les g-arbres 6-aires ayant n sommets et tels que 2 fils d'un même sommet ne sont pas de rangs consécutifs (modulo 6).

Contrairement aux arbres guingois, on ne connaît pas de langage codant les g-arbres p-aires; on ne sait pas non plus les énumérer, sauf pour $p=6$, ce qui correspond au problème des hexagones durs résolus par Baxter [1].

Par contre, la représentation des animaux 3D par les g-arbres a permis de trouver un algorithme qui les énumère selon leur taille pour les premières valeurs.

Chapitre 6

L'algorithme d'énumération des animaux 3D

On énumère ici les animaux dirigés 3D sur réseau cubique, cubique alterné et cubique centré suivant leur taille, c'est à dire leur nombre de points.

6.1 Le principe général de l'algorithme

Notre algorithme est basé sur la bijection entre les animaux dirigés 3D et les g-arbres p-aires ($p=3$, 4 ou 6).

Il consiste à générer récursivement tous les arbres p-aires de la taille voulue et éliminer au fur et à mesure ceux qui violent la condition de projection des g-arbres sur le réseau correspondant (hexagonal, carré ou triangulaire).

Remarque : un non g-arbre n'est pas obligatoirement généré entièrement. Dès qu'un sommet de l'arbre ne remplit pas la condition, cet arbre est abandonné.

6.2 L'énumération des arbres p-aires

6.2.1 Une définition des arbres p-aires

Définition 6.1 *Une* <u>*p-répartition*</u> *d'un entier n est un p-uplet R de p entiers positifs ou nuls $(n_1, \ldots, n_p)$ tels que $n_1 + \cdots + n_p = n$.*

Notation :

- $|T|$ est la taille de l'arbre T, c'est à dire son nombre de sommets.
- N_{t+1} est le nombre d'arbres p-aires de taille $t+1$.

On peut définir un arbre p-aire T de taille $t+1$ de la manière suivante :

> T est formé d'une racine r et d'une forêt d'arbres $(T_1, \ldots, T_p)$ telle que $|T_1| + \cdots + |T_p| = t$. (Si le sous-arbre T_i est vide alors $|T_i| = 0$)
> A la racine r est alors associée la p-répartition $(|T_1|, \ldots, |T_p|)$ de t.
>
> En appliquant récursivement ce procédé aux sous-arbres issus de r, on associe ainsi une p-répartition à chaque sommet interne de l'arbre de T.
>
> L'ensemble des p-répartitions obtenues définit entièrement l'arbre T.

Un exemple :

Un arbre 3-aire de taille 11 est représenté ci-dessous; les rangs des sous-arbres de chaque sommet sont croissants de gauche à droite.

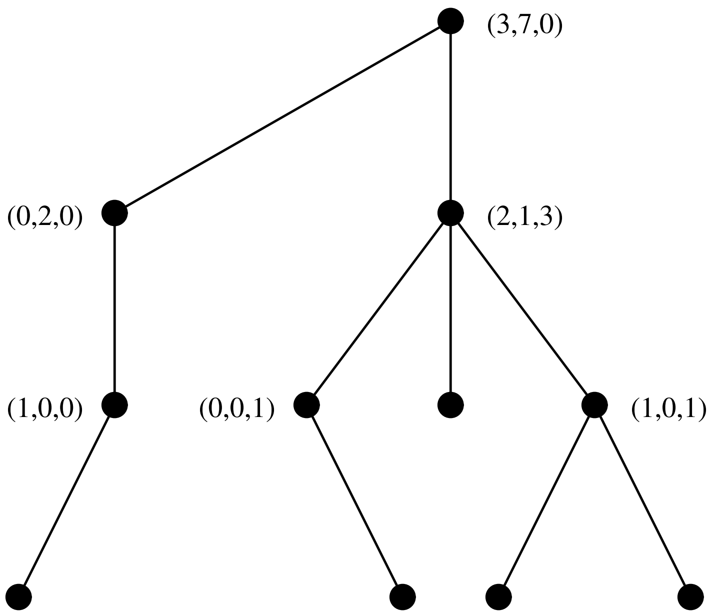


Figure 6.1: Un arbre 3-aire et les 3-répartitions associées.

De cette représentation des arbres p-aires, on déduit leur formule de dénombrement pour une taille donnée $t+1$:

$$N_{t+1} = \sum_{R=(t_1,\ldots,t_p)} N_{t_1} \times \cdots \times N_{t_p} \tag{6.1}$$

où R est une p-répartition de t.

Remarque : on verra en 6.4 que l'algorithme d'énumération est basé sur cette formule.

6.2.2 Un algorithme générant les p-répartitions

D'après la formule 6.1, on voit que le calcul du nombre d'arbres p-aires de taille n utilise toutes les p-répartitions de tous les entiers $k \in [1, n-1]$.

On a donc eu besoin d'établir un algorithme qui génère toutes les p-répartitions d'un entier k.

On a à notre disposition une variable globale **pp** qui est un tableau indicé de 0 à $p-1$ où est stockée la p-répartition en cours.

```
p-rep(p,k,j)
     int p;
     int k;  /* entier dont on calcule les p-répartitions */
     int j;  /* indice dans la p-répartition en cours */
{
     int i;  /* compteur */

     Si (k = 0)
     {
          Compléter pp par des 0;
          Sauver pp dans une liste de p-répartitions;
          return;
     }
     Si (p != 0)
     {
          Pour i de k à 0 par pas de -1
          {
               pp[j] = i;
               p-rep (p-1,k-i,j+1);
          }
     }
}
```

Figure 6.2: L'algorithme p-rep.

La complexité de cet algorithme est en $O(k^{p-1})$.

L'appel initial est fait par p-rep(p,k,0) où k est l'entier dont on veut toutes les p-répartitions.

L'appel de l'algorithme p-rep(p,k,0) est inséré dans une boucle pour k allant de 1 à $n-1$. Est disponible ainsi en mémoire un tableau indexé par k contenant la liste des p-répartitions pour chaque $k \in [1, n-1]$. On appelle ce tableau **tab-repart**.

6.3 L'élimination des arbres p-aires non g-arbres

On rappelle que la condition vérifiée par un g-arbre porte sur la façon dont se projettent sur un plan de référence les sous-arbres associés à un même sommet (cf chapitre 5).

Dans notre algorithme, on considère donc la projection des arbres p-aires sur un plan modélisant suivant le cas le réseau carré, triangulaire ou hexagonal.

Cas d'un arbre :

Chaque nœud de l'arbre a ses fils numérotés : rang entre 1 et p.
On génère alors récursivement les sous-arbres par ordre croissant des rangs des racines.

Lorsqu'on traite le fils de rang i d'un nœud x, les projections de ses frères de rang $j > i$ et celles de leurs voisins par une arête du réseau sont marquées sur le plan de référence (cf figure 6.3).

On ne garde alors que les sous-arbres issus du fils de rang i dont tous les nœuds se projettent sur une position non marquée du réseau.

Par exemple, lorsqu'on traite le fils n^o 1 d'un sommet x d'un arbre 4-aire ayant 3 fils : de rang 1, 2 et 4, on a la configuration suivante sur le plan de référence :

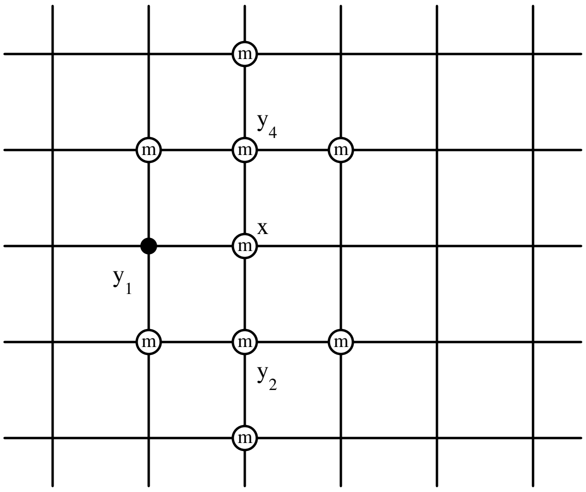


Figure 6.3: Configuration des positions marquées.

6.4 L'algorithme d'énumération des animaux 3D

Un tel algorithme revient à énumérer les g-arbres, du fait de leur bijection avec les animaux 3D.

Et ceci est fait en incluant dans un algorithme générant récursivement les arbres p-aires, le test sur la projection des sommets que l'on vient de voir en 6.3.

La constante globale **p** est le nombre maximum de fils d'un sommet de l'arbre : elle détermine le type d'animal 3D que l'on énumère (sur réseau cubique, cubique alterné, ou cubique centré).

Au départ, toutes les positions possibles du réseau considéré, représenté par une matrice, sont initialisées à "*non marquées*".

Toutes les p-répartitions utiles sont disponibles, on l'a vu en 6.2, dans le tableau *tab-repart*. Ce tableau est utilisé par la fonction **p-repsuiv** pour retourner la p-répartition suivante à traiter.

L'algorithme récursif dénombrant les g-arbres de taille "*taille*" est présenté figure 6.4.

Remarque : cet algorithme est très coûteux en temps. En effet, on ne peut se servir du nombre de g-arbres calculé pour des tailles plus petites à cause du test sur les projections que l'on doit effectuer pour chaque sous-arbre.

Par ce fait, pour calculer le nombre de g-arbres d'une taille donnée, on génère récursivement plusieurs fois les mêmes sous-arbres.

Cependant, une amélioration est possible dans le cas des animaux stricts et des animaux larges.

```
int enumeration3D(taille,pos-x,pos-y)
     int taille;  /* nbre de sommets de l'arbre */
     int pos-x,pos-y;  /* coordonnées de la projection de la racine */
{
     int prep[p];  /* p-répartition courante */
     int ss-arbres,nb-arbres = 0;  /* variables intermédiaires */
     int i;  /* compteur */
     int x-fils,y-fils;  /* coordonnées du fils traité */

     Si (taille = 1)
          return(1);
     Tant que (p-repsuiv(prep,taille−1,p)=OK)
          /* pour toutes les p-répartitions de taille−1 */
     {
          Si (marquer-fils(prep,pos-x,pos-y)=OK)
               /* vérification du fait que les fils ne se projettent pas
                  sur une position marquée, puis on marque leur position */
          {
               ss-arbres = 1;
               Pour i de 0 à p−1  /* pour chaque fils */
                    Si (prep[i] > 0)
                    {
                         calcul-coord-fils(i,pos-x,pos-y,x-fils,y-fils);
                         demarquer(x-fils,y-fils);
                         ss-arbres *= enumeration3D(prep[i],x-fils,y-fils);
                    }
               nb-arbres += ss-arbres;
          }
     }
     return(nb-arbres);
}
```

Figure 6.4: L'algorithme dénombrant les g-arbres de taille "*taille*".

6.5 Une amélioration de l'algorithme

L'amélioration apportée ici est valable pour les 3 types d'animaux : sur réseau cubique, cubique alterné et cubique centré.

Seuls les animaux stricts et larges de chaque type sont concernés; on verra plus tard pourquoi les primitifs ne remplissent pas les conditions requises.

On souhaite calculer le nombre de g-arbres p-aires de taille n.
Toutes les p-répartitions de $n-1$ donnent donc toutes les tailles et configurations possibles de sous-arbres issus de la racine r.

L'algorithme traite une à une les p-répartitions.

Deux cas sont privilégiés :

1. le cas où un seul élément de la p-répartition en cours est non nul.
   Cela veut dire qu'un seul sous-arbre est issu de r, et dans ce cas il n'y a aucun problème pour les projections des sommets.

2. le cas où, dans la p-répartition en cours, on traite le dernier élément non nul.
   Cela signifie que c'est le dernier sous-arbre issu de r dans l'ordre des rangs des fils de r.
   Et donc, il n'y a pas, non plus dans ce cas, de problème de projections.

Dans les deux situations que l'on vient de décrire, on peut donc, au lieu générer récursivement les sous-arbres possibles pour les compter, utiliser directement le nombre de g-arbres de la taille voulue, $n-1$ dans le cas 1 et la valeur de l'élément traité dans le cas 2.

Remarque : Ceci implique que l'on puisse fournir au programme la connaissance du nombre de g-arbres de toutes les tailles inférieures à celle que l'on veut calculer n.

Cependant, ce n'est pas une trop grande restriction dans le sens où ce programme a été mis en œuvre pour déterminer les premiers termes des séries génératrices des animaux 3D et qu'il n'est pas intéressant d'avoir le nombre d'animaux de taille k si on n'a pas celui pour la taille $k-1$ ou $k-2$.

Ainsi, avant d'exécuter la fonction *enumeration3D*, le programme lit dans un fichier les nombres déjà calculés et les met dans un tableau indexé par la taille. On nomme ce tableau "*ancien*".

La figure 6.5 donne l'algorithme d'énumération qui tient compte des 2 situations décrites précédemment.

```
int enumeration3D(taille,pos-x,pos-y)
     int taille;  /* nbre de sommets de l'arbre */
     int pos-x,pos-y;  /* coordonnées de la projection de la racine */
{
   int prep[p];  /* p-répartition courante */
   int ss-arbres,nb-arbres = 0;  /* variables intermédiaires */
   int i;  /* compteur */
   int x-fils,y-fils;  /* coordonnées du fils traité */

   Si (taille = 1)
        return(1);
   Tant que (p-repsuiv(prep,taille-1,p)=OK)
        /* pour toutes les p-répartitions de taille-1 */
   {
        Si ((taille = n) et (un-seul-non-nul(prep)=OK))  /* cas 1 */
             nb-arbres += ancien[taille-1];
        Sinon
             Si (marquer-fils(prep,pos-x,pos-y)=OK)
                  /* vérification du fait que les fils ne se projettent pas
                     sur une position marquée, puis on marque leur position */
             {
                  ss-arbres = 1;
                  Pour i de 0 à p-1  /* pour chaque fils */
                       Si (prep[i] > 0)
                       {
                            calcul-coord-fils(i,pos-x,pos-y,x-fils,y-fils);
                            demarquer(x-fils,y-fils);
                            Si ((taille = n) et (dernier-non-nul(prep) = i))  /* cas 2 */
                                 ss-arbres *= ancien[prep[i]];
                            Sinon
                                 ss-arbres *= enumeration3D(prep[i],x-fils,y-fils);
                       }
                  nb-arbres += ss-arbres;
             }
   }
   return(nb-arbres);
}
```

Figure 6.5: L'algorithme amélioré dénombrant les g-arbres de taille "*taille*".

Le cas des animaux primitifs :

La particularité d'un animal primitif est qu'aucun sommet du g-arbre associé n'a la même projection que la racine.

C'est une condition **globale** sur l'arbre qui ne peut être vérifiée qu 'en projetant systématiquement tous les sous-arbres possibles sur le plan de référence.
L'amélioration que l'on vient de voir ne peut donc pas s'appliquer pour les animaux primitifs.

6.6 Quelques précisions sur l'implémentation

6.6.1 Cas des animaux non stricts

Pour les animaux non stricts, une position supplémentaire est possible : un fils peut se trouver exactement sous son père (c'est à dire qu'il a la même projection que son père), mais seulement s'il est le seul fils.

On est alors amené à générer des 4, 5 ou 7-répartitions particulières d'un entier k : le dernier élément de la p-répartition est tout le temps nul sauf pour une qui est $(0, \ldots, 0, k)$.

L'astuce que l'on a choisi est la suivante :

on génère les 3, 4 ou 6-répartitions, en rajoutant un 0 à la fin de chacune d'elles, puis on crée celle pour le cas large : $(0, \ldots, 0, k)$.

6.6.2 Cas des animaux sur réseau cubique

On l'a vu au chapitre 5, les animaux sur réseau cubique sont en bijection avec les g-arbres 6-aires qui ont au maximum 3 fils de rangs non consécutifs par sommet.

Ainsi, au lieu de générer des 6-répartitions, il suffit de générer des 3-répartitions (t_1, t_2, t_3), puis pour chacune d'elles, intercaler des 0 de façon à avoir les configurations de la figure 6.6.

- Les configurations (1) et (2) sont créées pour chaque 3-répartition (t_1, t_2, t_3).
- Les configurations (3), (4) ou (5) sont créées pour les 3-répartitions ayant exactement 2 éléments non nuls : ceux spécifiés sur la figure.

Il est facile de voir que par ce procédé, on obtient bien toutes les 6-répartitions voulues.

En effet, pour la 3-répartition (t_1, t_2, t_3), on a également toutes les autres combinaisons (triées) possibles des éléments t_1, t_2 et t_3.
A toutes ces 3-répartitions (6 en tout), on applique le procédé décrit plus haut; ainsi on récupère toutes les configurations possibles avec t_1, t_2, t_3.

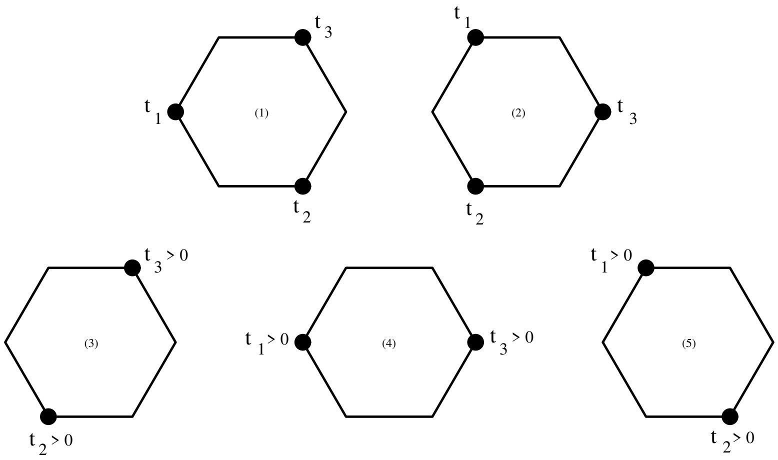


Figure 6.6: Positions des sommets à partir d'une 3-répartition.

6.6.3 Utilisation de PARI

Le nombre d'animaux 3D augmente très vite en fonction de la taille et, de ce fait, les nombres entiers que nous devions manipuler ont très rapidement dépassé les capacités du langage *C* : nous avons alors choisi d'utiliser PARI.

Le système PARI a été développé à l'UFR Mathématiques de l'université Bordeaux I par C. Batut, D. Bernardi, H. Cohen et M. Olivier [14].
C'est un logiciel de calcul formel, dont la puissance ne peut cependant pas être comparée à celle de MAPLE ou Mathematica, qui sont beaucoup plus sophistiqués.
Les deux principaux avantages de PARI sont d'une part sa rapidité, et d'autre part d'offrir la possibilité de manipuler directement des types bien familiers aux mathématiciens : nombres entiers et réels (de toute taille), nombres rationnels, nombres complexes, polynômes, séries, vecteurs et matrices, ...

On peut utiliser PARI sous deux formes :

- comme un calculateur sophistiqué nommé GP,
- comme une librairie pour un programme *C*, *C++*, *Pascal* ou *Fortran.*

Nous avons donc inclus dans notre programme la librairie PARI, ce qui a permis une manipulation simple de nos grands nombres entiers.

Toutefois, une précaution était à prendre en ce qui concerne la mémoire : en effet, tout élément PARI se trouve dans une pile spéciale pour PARI, et le soin de la gestion de cette pile est laissé à l'utilisateur.
Un effort a donc été fait pour optimiser cette gestion et la mémoire requise.

Chapitre 7

Les résultats

On présente ici les trois tableaux de résultats que l'on a obtenus avec l'algorithme d'énumération décrit au chapitre précédent.

Par abus de langage, on identifie un type d'animal 3D avec le modèle de physique statistique correspondant :

- "les hexagones durs" pour les animaux sur réseau cubique,
- "les triangles durs" pour les animaux sur réseau cubique alterné,
- "les carrés durs" pour les animaux sur réseau cubique centré.

Bien que Baxter ait trouvé des équations permettant de résoudre le modèle des hexagones durs [1], on a implémenté l'algorithme pour ce modèle pour 2 raisons :

1. pouvoir vérifier l'exactitude de l'algorithme,
2. pouvoir calculer la série des stricts qui n'est pas donnée par les équations de Baxter.

Les hexagones durs :

Taille	Stricts	Primitifs	Larges
1	1	1	1
2	6	6	7
3	45	45	58
4	365	374	519
5	3 101	3 300	4 856
6	27 144	30 282	46 780
7	242 636	285 682	460 027
8	2 202 873	2 751 258	4 593 647
9	20 241 055	26 921 589	46 416 730
10	187 766 940	266 797 836	473 464 492
11	1 755 409 652	2 671 518 873	4 866 762 231
12	16 517 284 570	26 981 681 412	50 346 419 064
13		274 493 963 898	523 649 493 732
14		2 809 920 769 440	5 471 647 249 551

Les carrés durs :

Taille	Stricts	Primitifs	Larges
1	1	1	1
2	4	4	5
3	22	22	31
4	130	136	209
5	807	897	1476
6	5 163	6 168	10 739
7	33 742	43 670	79 780
8	224 002	315 956	601 905
9	1 505 146	2 324 479	4 595 485
10	10 211 027	17 329 828	35 419 710
11	69 814 781	130 605 478	275 109 858
12	480 435 484	993 182 984	2 150 537 435
13	3 324 233 772	7 610 051 579	16 901 814 190
14	23 108 532 996	58 689 316 888	133 452 123 796
15	161 288 459 289		

Les triangles durs :

Taille	Stricts	Primitifs	Larges
1	1	1	1
2	3	3	4
3	12	12	19
4	52	55	98
5	238	273	531
6	1 125	1 425	2 971
7	5 438	7 695	16 997
8	26 715	42 576	98 830
9	132 871	239 925	581 788
10	667 312	1 371 555	3 458 249
11	3 377 906	7 931 817	20 718 292
12	17 210 522	46 310 127	124 929 233
13	88 169 685	272 559 558	757 421 601
14	453 810 095	1 615 163 592	4 613 459 330
15	2 345 209 383	9 627 985 773	28 213 402 944
16	12 162 367 228	57 688 721 354	
17	63 270 384 303		

Partie II

Les approximations de séries génératrices

Chapitre 1

La méthode de S. Plouffe

1.1 L'idée

D'une série α, on ne connait que les premiers termes :

$$\alpha = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n + O(x^{n+1}) \ .$$

On veut alors calculer une fonction génératrice $f(x)$ pour α, sans avoir d'information particulière sur la forme de $f(x)$.
Un cas pour lequel on connait une solution à ce problème est celui où la fonction génératrice est une fonction rationnelle :

$$f(x) = \frac{p_0 + p_1 x + p_2 x^2 + \cdots + p_j x^j}{q_0 + q_1 x + q_2 x^2 + \cdots + q_k x^k} \ . \tag{1.1}$$

Plusieurs algorithmes résolvant ce problème ont été implémentés :

- certains utilisent une méthode à coefficients indéterminés dans l'équation 1.1 avec $j + k = n$, n étant le degré de précision de la série α considérée.
- d'autres algorithmes utilisent la théorie des approximants de Padé.
  L'approximant de Padé (p, q) de $g(x) = \sum u_n x^n$ est la fraction P/Q où $d^\circ(P) = p$ et $d^\circ(Q) = q$, qui lorsqu'on la développe en série de Taylor, correspond avec $g(x)$ le plus loin possible.
  remarque : un tel algorithme est implémenté dans MAPLE (convert/ratpoly).

Plouffe et Bergeron ont montré que l'on peut facilement étendre la classe des séries dont on est en mesure de fournir une fonction génératrice explicite, en ne connaissant que les premiers termes.

L'idée est d'appliquer à une série α donnée des opérations qui pourraient la transformer en une série admettant une fonction génératrice rationnelle.

Ainsi, si cela réussit, pour trouver la fonction génératrice explicite de la série α de départ, il suffit d'appliquer l'inverse de ces opérations sur la fonction rationnelle obtenue $P(x)/Q(x)$.

Les opérations possibles qui viennent à l'esprit sont :

- $\ln(\alpha)$
- $\exp(\alpha)$
- dérivée(α)
- dérivée logarithmique(α)
- ...

Et alors, on récupèrera des formules du type :

$\exp(\frac{P(x)}{Q(x)})$, $\ln(\frac{P(x)}{Q(x)})$, $\int(\frac{P(x)}{Q(x)})$, $\sqrt{(\frac{P(x)}{Q(x)})}$, et des combinaisons de celles-ci.

Cependant, il n'est pas utile de combiner arbitrairement toutes ces transformations.
Il est en fait apparu, au fil des applications, que seules 3 opérations sont suffisantes :

- la dérivée,
- la dérivée logarithmique,
- l'inverse pour la substitution.

1.2 Le programme

Plouffe et Bergeron ont écrit un programme MAPLE qui implémente leur méthode.

La base de ce programme est bien sûr un test sur l'existence d'une "bonne" fonction génératrice rationnelle pour une série.
La fonction "convert/ratpoly" de MAPLE, qui implémente l'un des meilleurs algorithmes connus (basé sur la théorie des approximants de Padé), convertit une série en une fonction rationnelle, si c'est possible.
La question est : "*Comment décider si la fraction rationnelle obtenue est la bonne ?*".
En effet, MAPLE donne toujours un résultat, même si ce résultat est énorme.
La longueur de l'expression est donc le premier critère de validité de la fonction.
Le deuxième critère est un test sur la somme des degrés du numérateur et du dénominateur : cette somme doit être inférieure au degré de précision de la série donnée.

L'association de ce test de rationalité avec les opérations décrites ci-dessus (dérivée, dérivée logarithmique, inversion) fait toute l'efficacité de ce programme.
Plusieurs compositions de ces opérations sont appliquées à la série :

si l'une d'elles Γ aboutit à une fonction rationnelle $f(x)$, alors $\Gamma^{-1}(f(x))$ est retournée comme fonction génératrice pour la série donnée.

sinon le programme s'arrête en avouant son échec.

Remarque : Ce programme a permis de trouver une fonction génératrice, ordinaire ou exponentielle, pour plus de 1000 séries apparaissant dans le Handbook de Sloane [10] qui en contient 4568.

Des exemples d'application :

Exemple 1:

"Comment couper un gâteau pour faire un maximum de parts?"

Les premiers termes de la suite sont :

$$1, 2, 4, 7, 11, 16, 22, 29, 37, 46, 56, ...$$

Le programme donne le résultat suivant :

$$-\frac{1 - x + x^2}{(x-1)^3}$$

Exemple 2:

"La suite de Fibonacci"

Les premiers termes de la suite sont :

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...$$

Le programme donne le résultat suivant :

$$\frac{1}{1 - x - x^2}$$

Exemple 3:

"Les nombres merveilleux de Delmo"

Les premiers termes de la suite sont :

$$1, 121, 12321, 1234321, 123454321, 12345654321, 1234567654321, ...$$

Ces nombres sont générés par l'astuce suivante :

$$\begin{array}{c} 1 * 1 \\ 11 * 11 \\ 111 * 111 \\ 1111 * 1111 \\ \cdots \end{array}$$

Le programme donne le résultat suivant :

$$\frac{1 + 10x}{(1-x)(1-10x)(1-100x)}$$

Exemple 4:

Une fonction qui n'est pas une fraction rationnelle.

Les premiers termes de la suite sont :

$$1, 3, 13, 63, 321, 1683, 8989, ...$$

Le programme donne le résultat suivant en indiquant que c'est la dérivée logarithmique de la série qui est rationnelle :

$$\frac{1}{\sqrt{1-6x+x^2}}$$

1.3 Quelques remarques

C'est une méthode numérique n'agit que par approximations. Par conséquent, les résultats auxquels on peut aboutir restent dans le domaine des conjectures et ne peuvent être considérés comme des démonstrations.

Cependant, on obtient, pour la fonction génératrice d'une série, une expression qui a de "fortes chances" d'être la bonne; et ceci ne peut que faciliter la démonstration rigoureuse de sa validité (et pourquoi pas donner des idées de bijections nouvelles?).

1.4 Essais d'application aux animaux 3D

On a donc essayé d'appliquer cette méthode d'approximations de séries génératrices aux suites de nombres calculés avec notre algorithme d'énumération des animaux 3D, présenté partie I.

Dans un premier temps, on a tenté d'exécuter le programme de Plouffe et Bergeron pour chaque série de nombres, mais sans succès; la réponse était toujours la même : "Je ne peux calculer la fonction génératrice, essayez de donner plus de termes".

Pour avoir plus de précisions sur la méthode et sur ses applications, nous avons pris contact avec Simon Plouffe, lors de sa venue au LaBRI au mois de février. Celui-ci nous a cordialement aidé dans notre travail.
Nous avons essayé ensembles d'exploiter une autre idée : celle d'étudier plus précisément les équations déterminées par Baxter dans le cas des hexagones durs, de façon numérique et par le biais de MAPLE, pour tenter de trouver quelques particularités qui auraient peut-être pu être ensuite appliquées aux modèles des carrés durs et des triangles durs.
L'ensemble de ces équations est un système de 5 équations dont les inconnues sont des séries.
Remarque : on ne peut les spécifier ici car, à notre connaissance, ces résultats ne sont pas encore publiés.

Malheureusement, cette étude n'a pu être menée à bien compte tenu de la complexité de l'équation obtenue après avoir résolu les deux premières et inclus leurs résultats dans la troisième.

Une conclusion est qu'il faut calculer le plus de termes possibles de chaque série pour espérer exploiter au mieux cette méthode.
Cependant, une question se pose : est-ce réellement le manque de coefficients qui fait échouer le calcul de la fonction génératrice ?

En effet, la complexité rencontrée avec les équations de Baxter ne reflète pas une impression positive quant à l'existence d'une formule close simple pour nos séries, tout au moins pour celle des hexagones durs !...

Chapitre 2

La méthode de A.J. Guttmann

En 1970, Guttmann et Joyce ont développé une méthode d'analyse de séries, qui par la suite a été étendue par d'autres, comme Fisher et Au-Yang; elle est connue sous le nom de "*méthode d'approximants différentiels*".

Dans cette méthode, les coefficients de la série sont utilisés pour définir une équation différentielle linéaire avec des coefficients polynômiaux.

Un grand nombre de modèles de physique statistique ont pu être résolus par cette méthode; mais beaucoup ne satisfont pas d'équations différentielles sous-jacentes à la méthode d'approximants différentiels.

L'analyse initiale des auteurs indique que dans la plupart des cas, ils vérifient une équation algébrique de la forme :

$$\sum_{m=0}^{k} P_m(x) f(x)^m = 0 \tag{2.1}$$

où $f(x)$ est la solution du modèle considéré et les $P_m(x)$ sont des polynômes.
Si on note α_m le degré du polynôme $P_m(x)$, alors l'équation algébrique 2.1 est dénotée par $[\alpha_k, \alpha_{k-1}, \ldots, \alpha_1, \alpha_0]$.

Guttmann et Brak ont donc développé une nouvelle méthode d'analyse de séries, basée sur l'équation algébrique correspondante [5].

2.1 L'idée

On ne connait que les premiers termes d'une série $f(x)$.

Une première étape consiste à chercher si $f(x)$ vérifie une équation algébrique de la forme 2.1.

Pour chaque ordre k de l'équation algébrique, on essaie de trouver un ensemble de polynômes vérifiant l'équation 2.1 en augmentant méthodiquement les degrés des polynômes $P_m(x)$.
On décide que l'ordre k n'est pas approprié lorsque les degrés des $P_m(x)$ sont tels que l'on dépasse la précision de la série initiale, c'est à dire le nombre de coefficients disponibles.

Si la première étape aboutit, une seconde étape consiste à analyser l'équation algébrique obtenue afin de déterminer le point critique et les exposants critiques, paramètres fondamentaux d'un modèle de physique statistique.

Si on considère l'équation 2.1 pour une valeur de x fixée, elle devient alors un polynôme de degré k en f, qui peut avoir au maximum k racines distinctes.
Le *point critique* est alors un des points pour lesquels les k racines ne sont pas toutes distinctes.

En ce qui concerne les *exposants critiques*, ils sont calculés grâce aux développements de Puiseux (cf [5] et [7]).

2.2 Quelques remarques et exemples

A première vue, on est tenté de penser que cette nouvelle méthode n'apporte rien de plus que la méthode déjà existante d'approximants différentiels.

Elle a toutefois un énorme avantage, étant donné qu'on ne dispose souvent que d'un nombre fini (et réduit) de termes de la série cherchée.
En effet, la détermination de l'équation algébrique nécessite beaucoup moins de coefficients que celle de l'équation différentielle correspondante.

Guttmann donne l'exemple de la fonction densité moyenne ρ du modèle des hexagones durs, qui satisfait l'équation algébrique :

$$3(1-11z-z^2)\rho^4-(1-66z-11z^2)\rho^3-15z(3+z)\rho^2+3z(4+3z)\rho-z(1+2z)=0$$

Les coefficients polynômiaux de l'équation différentielle correspondante sont les suivants :

$$\begin{aligned}
P_0(z) &= 40(-50-204z-3462z^2+21999z^3+3462z^4-204z^5+50z^6)\\
P_1(z) &= 40(-68+926z+9768z^2-52419z^3-149952z^4+32394z^5\\
&\quad -95291z^6-17954z^7-558z^8+11z^9)\\
P_2(z) &= 4R(z)(-30+2194z+26526z^2-493774z^3+52405z^4-415340z^5\\
&\quad -271417z^6-16473z^7+352z^8)\\
P_3(z) &= 36zR(z)^2(6+6z-3620z^2+1600z^3-6630z^4-1013z^5+22z^6)\\
P_4(z) &= 9z^2R(z)^3(-6-184z+120z^2-504z^3+11z^4)
\end{aligned}$$

où $R(z)=-1+11z+z^2$.

L'équation algébrique est une équation $[2, 2, 2, 2, 2]$ nécessitant 14 coefficients pour sa détermination, tandis que l'équation différentielle est une équation $[12, 11, 10, 9, 6]$ en nécessitant 52.

On remarque ici que la différence entre les deux méthodes n'est pas négligeable, et ceci à deux niveaux :

- le nombres de termes nécessaires,
- la complexité des polynômes et donc de l'équation.

D'autres problèmes pour lesquels Guttmann et Brak ont trouvé une équation algébrique :

Un problème déjà résolu les polygones convexes.
L'équation algébrique du $2^{ème}$ ordre associée est du type $[4, 7, 10]$.

Un problème jusqu'alors non résolu les polygones linéairement convexes.
L'équation algébrique associée est d'ordre 4 et de type $[3, 4, 5, 6, 6]$.

Si la nouvelle méthode de Guttmann et Brak peut permettre d'obtenir une équation algébrique vérifiée par la fonction génératrice d'un modèle, elle ne donne pas de manière explicite une façon de calculer une formule close pour cette fonction, comme c'est le cas dans la méthode de Plouffe.

Si l'ordre de l'équation algébrique que l'on trouve est inférieur ou égal à 4, alors par des techniques de résolution classiques, on peut espérer trouver la fonction génératrice, si l'équation n'est pas trop compliquée.

Par contre, si l'ordre de l'équation algébrique est plus grand que 4, sa résolution échappe aux techniques connues, et en général, on a peu de chance de déterminer la fonction génératrice, sauf peut-être dans des cas très particuliers.

Les méthodes de Guttmann et de Plouffe se distinguent donc par le fait que celle de Guttmann est beaucoup plus proche des problèmes de physique statistique (point et exposants critiques), alors que celle de Plouffe se rapproche plus de l'optique des combinatoriciens (fonction génératrice).

Par contre, une même remarque que pour la méthode de Plouffe peut être faite ici.
L'approximant que l'on trouve ne constitue pas un résultat indiscutable, il n'est qu'une conjecture. Cependant, si l'on retrouve nos coefficients de départ grâce à l'équation algébrique, il est peu probable que la conjecture soit fausse.
Et il est bien connu qu'un résultat conjecturé est beaucoup plus simple à prouver ...

2.3 Le programme de Andrew Conway

Andrew Conway, un étudiant travaillant avec A.J. Guttmann, a mis en œuvre un programme C++ implémentant la méthode des approximants algébriques, ceci afin d'étudier l'utilité et les performances de la méthode dans le cadre de la mécanique statistique [7].

Une petite description

Conway a gracieusement mis à notre disposition les sources de la première étape de son programme. Celle-ci consiste à calculer l'approximant algébrique d'une série en analysant ses premiers termes.
Remarque : on rappelle que l'approximant algébrique de f est une équation de la forme 2.1.

Ce programme, qu'il a nommé **alg**, est exécuté avec un fichier d'entrée, qui contient les données et les différentes options. Ce fichier décrit au minimum les premiers termes de la série, et le (ou les) type d'approximant algébrique $[\alpha_k, \alpha_{k-1}, \ldots, \alpha_1, \alpha_0]$ que l'algorithme doit tenter de calculer.

En sortie, **alg** donne l'équation algébrique de la forme 2.1 pour chaque type d'approximant n'ayant provoqué aucune erreur.

Essai d'application aux animaux 3D

L'utilisation de ce programme pourrait être très utile pour essayer de découvrir d'éventuels approximants algébriques pour les séries des animaux 3D.

En effet, on a vu en 2.2 que Guttmann a exhibé un approximant algébrique de type $[2, 2, 2, 2, 2]$ pour la fonction densité moyenne des hexagones durs.
On pourrait être alors tenté de penser qu'il en existe un aussi pour les carrés durs et les triangles durs.

On a appris malheureusement trop tard l'existence de ce programme, et on n'a donc pas pu l'exploiter pleinement.

Cependant, on a effectué quelques tests pour chaque série. Et si ces premiers essais n'ont pas abouti à un approximant algébrique, cela nous a permis de constater à nouveau que le nombre de termes dont on dispose actuellement est insuffisant. On a trop souvent rencontré le message : “not enough information”.

A.J. Guttmann et Andrew Conway viennent à Bordeaux en septembre prochain. Et il pourrait être particulièrement intéressant d'avoir ensemble une discussion sur ce vaste sujet que sont les animaux 3D.

Chapitre 3

Comment deviner qu'une série est rationnelle ?

Le titre de ce chapitre est le titre de l'article de Viennot et Roblet [13].

Ils présentent une démonstration du fait que toute série formelle de terme constant non nul admet un développement en fraction continuée, du type C-fraction, en exhibant un nouvel algorithme de calcul des coefficients de cette C-fraction à partir de ceux de la série formelle.

Pour cela, ils utilisent une représentation des coefficients de la série comme somme de valuations de chemins "avec sauts", qui sont une généralisation des chemins de Dyck.

Cette propriété des séries formelles est intéressante dans le cadre de notre travail, car elle fournit un moyen heuristique pour déterminer si une série dont on connait les premiers termes est une fraction rationnelle, ainsi qu'un moyen de calcul de cette fraction.

Nous avons eu alors l'idée d'implémenter et d'utiliser ce nouveau test de rationalité dans le même contexte qui celui de la méthode de Plouffe.

3.1 Les chemins avec sauts et les fractions continues

Soient $\alpha = (\alpha_i)_{i\geq 1}$ une suite infinie (resp. finie) de nombres entiers strictement positifs et $\lambda = (\lambda_i)_{i\geq 1}$ une suite infinie (resp. finie de même longueur que α) de nombres réels non nuls.

On note $S = (s_k)_{k\geq 0}$ la suite des sommes partielles de α :

$$\begin{cases} s_0 = 0 \\ s_k = \alpha_1 + \cdots + \alpha_k \quad \forall k \geq 1 \end{cases}$$

Définition 3.1 *Soit $w = (t_0, \ldots, t_n)$ un chemin sur $\mathrm{Z} \times \mathrm{N}$, où $t_i = (x_i, y_i) \in \mathrm{Z} \times \mathrm{N}$.*
On dit que w est un chemin de type α valué par λ s'il vérifie :

1. *l'origine est* $t_0 = (0,0)$ *et l'extrémité* $t_n = (x_n, 0)$,

2. *pour tout* i *vérifiant* $0 \leq i \leq n-1$ *:*

   - *ou bien il existe un entier* $k \geq 0$ *tel que (cf figure 3.1 (a))*

   $$y_i = s_k \ , \ y_{i+1} = s_{k+1} \quad et \quad x_{i+1} = x_i + \alpha_{k+1} \ ,$$

   (t_i, t_{i+1}) *est un pas montant, sa valuation est* 1 *;*

   - *ou bien il existe un entier* $k \geq 1$ *tel que (cf figure 3.1 (b))*

   $$y_i = s_k \ , \ y_{i+1} = s_{k-1} \quad et \quad x_{i+1} = x_i + \alpha_k \ ,$$

   (t_i, t_{i+1}) *est un pas descendant, sa valuation est* λ_k *.*

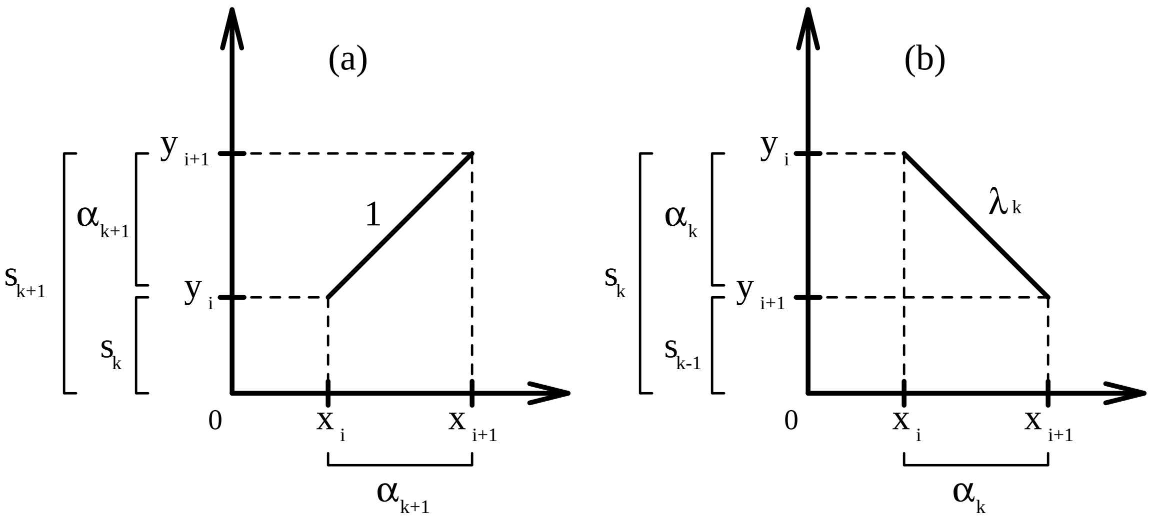


Figure 3.1: Illustration de la définition 3.1.

D'autres définitions :

La valuation d'un chemin w, notée $v(w)$, est définie comme le produit des valuations de ses pas.
Par convention, le chemin vide est valué 1 ($v((0,0)) = 1$).

On note $|w|$ la longueur de w.

On appelle les nombres s_k les niveaux des chemins de type α.

Un exemple :

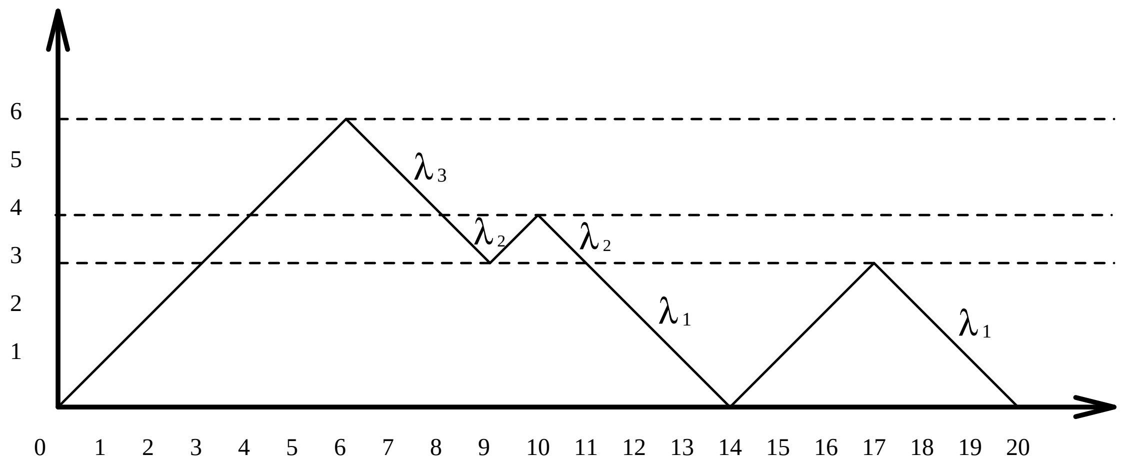


Figure 3.2: Un chemin de type $(3, 1, 2)$ de longueur 20.

On note $w(\alpha, \lambda)$ l'ensemble des chemins de type α valués par λ.

Définition 3.2 *On définit* $f(t; \alpha; \lambda) = f(t; \alpha_1, \ldots, \alpha_i, \ldots; \lambda_1, \ldots, \lambda_i, \ldots)$ *par*

$$f(t; \alpha; \lambda) = \sum_{w(\alpha,\lambda)} v(w) t^{\frac{|w|}{2}} \ .$$

Viennot et Roblet ont alors montré que $f(t; \alpha; \lambda)$ se développe en C-fraction :

$$f(t; \alpha; \lambda) = \cfrac{1}{1 - \cfrac{\lambda_1 t^{\alpha_1}}{1 - \cfrac{\lambda_2 t^{\alpha_2}}{\cdots}}} \ .$$

Le théorème fondamental de cet article est le suivant :

Théorème 3.1

1. *Soit f une série formelle $f = 1 + \sum_{n \geq 1} \mu_n t^n$.*
   Il existe alors un unique couple (α, λ), tel que $f = f(t; \alpha; \lambda)$.

2. *De plus, f est une fraction rationnelle si et seulement si les suites α et λ sont finies.*

La démonstration de ce théorème est une preuve constructive des suites α et λ.

Elle induit donc un algorithme de calcul du développement en C-fraction de la série, ce qui nous donne une fraction rationnelle si les suites α et λ sont finies.

3.2 L'algorithme de calcul de α et λ

On suppose connus les $(n+1)$ premiers termes de la série $f = 1 + \sum_{i=1}^{n} \mu_i t^i$.

Etape 1 **:** <u>détermination de α_1 et λ_1.</u>

Il existe un entier $p \geq 1$ tel que $\mu_1 = \ldots = \mu_{p-1} = 0$ et $\mu_p \neq 0$.
On a $f = 1 + \mu_p t^p + o(t^p)$.
Si on remarque que $f(t;\alpha;\lambda) = 1 + \lambda_1 t^{\alpha_1} + o(t^{\alpha_1})$, on en déduit que

$$\alpha_1 = p \quad \text{et} \quad \lambda_1 = \mu_p \ .$$

Notation : On note h^k_{m,s_i} la somme des valuations des chemins de type α, bornés au niveau s_k et allant du point $(0,0)$ au point de la droite D_m d'équation $x + y = 2m$ situé au niveau s_i.

Etape $k+1$ **:** <u>détermination de α_{k+1} et λ_{k+1} connaissant α_i et λ_i pour $1 \leq i \leq k$.</u>

On suppose connus aussi tous les h^k_{m,s_i} pour m vérifiant $\min_{1\leq j\leq k} \{s_k + 1 - \alpha_j\} \leq m \leq s_k$ (et bien sûr pour $1 \leq i \leq k$).

On commence par calculer $h^k_{s_k+1,0}$ pour le comparer à μ_{s_k+1}; s'il y a égalité, on calcule $h^k_{s_k+2,0}$, puis on le compare à μ_{s_k+2} , et ainsi de suite jusqu'à ce qu'on rencontre un entier p tel que $h^k_{s_k+p,0} \neq \mu_{s_k+p}$.
(on expliquera plus tard le cas $h^k_{s_k+p,0} = \mu_{s_k+p}$ $\forall p \geq 1$)
On calcule les $h^k_{s_k+j,s_i}$ pour $1 \leq j \leq p$ et $0 \leq i \leq k$ par le système S^k_j suivant :

$$\begin{cases} h^k_{s_k+j,s_k} = h^k_{s_k+j-\alpha_k,s_{k-1}} \\ \\ h^k_{s_k+j,s_i} = \lambda_{i+1} h^k_{s_k+j,s_{i+1}} + h^k_{s_k+j-\alpha_i,s_{i-1}} \quad \text{pour} \quad 1 \leq i \leq k-1 \\ \\ h^k_{s_k+j,0} = \lambda_1 h^k_{s_k+j,s_1} \end{cases}$$

Ce système est illustré pour $j=1$ sur la figure 3.3.

Pour calculer $h^k_{s_k+j,s_i}$, on fait la somme des valuations de tous les chemins de type α bornés au niveau s_k et allant du point $(0,0)$ au point situé au niveau s_i de la droite D_{s_k+j}. Ces chemins sont de 2 sortes :

1. ceux finissant par un pas descendant du niveau s_{i+1} et de valuation λ_{i+1} ,
2. ceux finissant par un pas montant du niveau s_{i-1} et de valuation 1.

La somme des valuations des chemins du cas 1 est $\lambda_{i+1} h^k_{s_k+j,s_{i+1}}$, et celle des chemins du cas 2 est $h^k_{s_k+j-\alpha_i,s_{i-1}}$.
Remarque : pour $h^k_{s_k+j,s_k}$ se pose seulement le cas 2, et pour $h^k_{s_k+j,0}$ le cas 1.

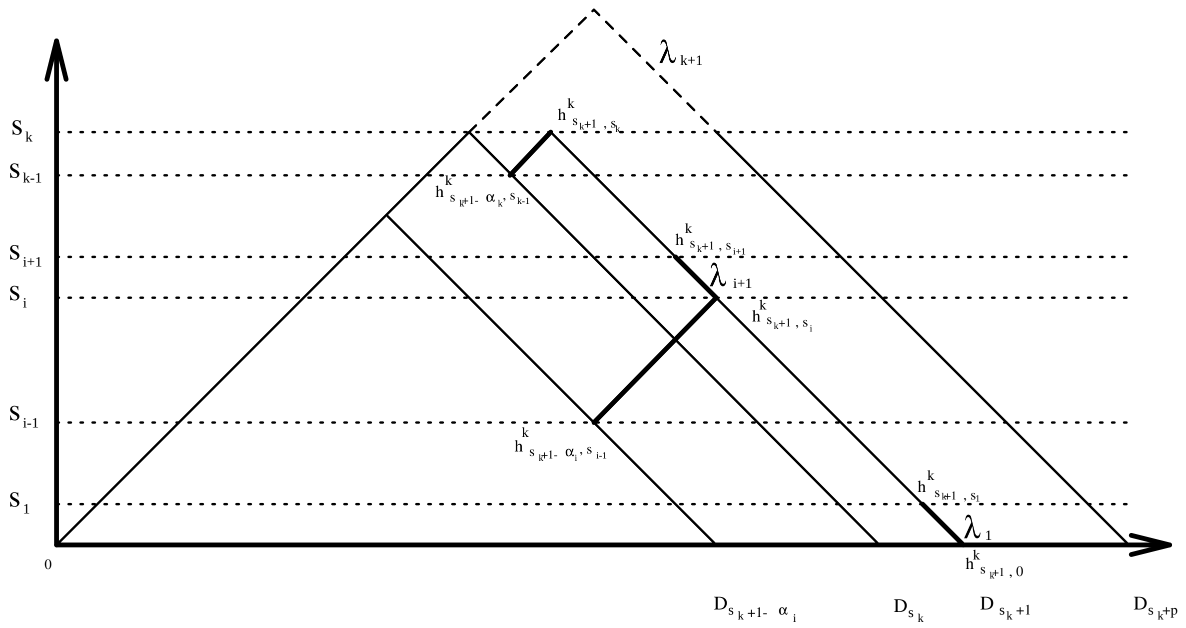


Figure 3.3: Illustration pour le calcul du système S_1^k.

L'entier p tel que $h^k_{s_k+p,0} \neq \mu_{s_k+p}$ permet de déterminer α_{k+1} et λ_{k+1} :

- $\alpha_{k+1} = p$,
- $(\lambda_1 \ldots \lambda_k)\lambda_{k+1} + h^k_{s_k+p,0} = \mu_{s_k+p}$.

Avant de passer à l'étape $k+2$, on calcule les h^{k+1}_{m,s_i} pour m tel que $\min\limits_{1 \leq j \leq k+1} \{s_{k+1} + 1 - \alpha_j\} \leq m \leq s_{k+1}$ par :

$$\begin{cases} h^{k+1}_{m,s_i} = h^k_{m,s_i} & \text{pour} \quad m \leq s_{k+1} \\ h^{k+1}_{s_{k+1},s_{k+1}} = 1 & \\ h^{k+1}_{s_{k+1},s_i} = h^k_{s_{k+1},s_i} + \lambda_{i+1} \ldots \lambda_k \lambda_{k+1} & \text{pour} \quad 0 \leq i \leq k \end{cases}$$

Cet algorithme se poursuit tant qu'il reste des coefficients μ_i de la série considérée qui n'ont pas été encore exploités, c'est à dire tant que $s_k + j \leq n$.

Le cas où $h^k_{s_k+p,0} = \mu_{s_k+p}\ \forall p \geq 1$:

Ceci traduit le fait que la série de départ $f = 1 + \sum_{i=1}^{n} \mu_i t^i$ est la fraction rationnelle :

$$f(t;(\alpha_1,\ldots,\alpha_k);(\lambda_1,\ldots,\lambda_k)) = \cfrac{1}{1-\cfrac{\lambda_1 t^{\alpha_1}}{\ddots \over 1-\lambda_k t^{\alpha_k}}} \; .$$

On ne peut bien sûr pas détecter ce phénomène de manière algorithmique, cela supposerait la connaissance de tous les coefficients μ_i de la série de départ et une infinité de calculs.

Notre hypothèse est que l'on connait les $(n+1)$ premiers termes de f : 1 , μ_1 , $\ldots$, μ_n.

Admettons qu'on applique l'algorithme ci-dessus, et qu'à la $k{+}1^{ième}$ étape, on obtienne : $h^k_{m,0} = \mu_m$ pour tout m tel que $s_k{+}1 \leq m \leq n$.

Si cela ne nous permet pas d'affirmer que f est une fraction rationnelle, on peut le supposer.
Et alors la fonction rationnelle définie par $f(t;(\alpha_1,\ldots,\alpha_k);(\lambda_1,\ldots,\lambda_k))$ est alors un bon candidat.

Remarque : plus l'intervalle $[s_k+1, n]$ auquel appartient m est grand, plus fortes sont les chances pour la série f d'être réellement une fraction rationnelle.
On approfondira cette remarque en 3.3.1, lors de l'implémentation de l'algorithme.

3.3 Une idée d'application de cet algorithme et son implémentation

A la suite de l'article [13], Viennot et Roblet n'ont pas effectué l'implémentation de l'algorithme, et par conséquent, on ne connait pas les performances exactes de ce nouveau test de rationalité par rapport aux méthodes déjà existantes : comme par exemple celle des approximants de Padé.

Nous avons alors eu l'idée de réaliser cette implémentation, et ceci pour deux principales raisons.

La première est qu'il pourrait être très intéressant d'étudier les éventuelles améliorations que cette méthode est susceptible d'apporter.
Dans le contexte de notre travail, la plus importante serait la diminution du nombre de termes exigés pour calculer la fraction rationnelle, lorsqu'elle existe.

La seconde raison est qu'on peut l'utiliser dans le cadre de la méthode d'approximations de séries génératrices de Plouffe.

Notre but est d'y inclure l'algorithme de Viennot et Roblet comme test de rationalité à la place des approximants de Padé.
C'est ainsi que nous avons choisi de programmer l'algorithme sous MAPLE, dans le souci d'en avoir une application directe avec le programme de Plouffe.

3.3.1 Problèmes concernant l'implémentation

L'algorithme a donc été programmé sous MAPLE, exactement de la façon dont il a été décrit en 3.2.

Les deux principaux problèmes que l'on a rencontrés concernent l'hypothèse de départ sur la série, et la conclusion, c'est à dire le critère indiquant si la série est rationnelle ou non.

Le premier problème est en fait très simple à résoudre.
L'hypothèse est que la série à laquelle s'applique l'algorithme est de la forme

$$f = 1 + \mu_1 x + \mu_2 x^2 + \cdots + \mu_n x^n \ . \tag{3.1}$$

Or, les séries que l'on étudie peuvent être de toute autre forme.
Il suffit alors de diviser la série par son premier monôme (son monôme de plus bas degré) pour obtenir la forme 3.1 :
Soit $a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$ une série.
On cherche le premier entier p tel que $a_p \neq 0$.
On applique l'algorithme sur la série $1 + \dfrac{a_{p+1}}{a_p}\, x + \dfrac{a_{p+2}}{a_p}\, x^2 + \cdots + \dfrac{a_k}{a_p}\, x^{k-p}$
et, à la fin, on multiplie la fraction trouvée par $a_p x^p$.

Par contre, pour le second problème, on ne peut pas proposer une solution totalement fiable.
En effet, on l'a vu à la fin de la section 3.2, le critère de rationalité ne peut être détecté algorithmiquement et ainsi, la conclusion de l'algorithme est déduite en fonction de la "*grandeur*" de l'intervalle $[s_{k+1}, n]$.
Se pose alors la question déjà suggérée par la dernière remarque de 3.2 :
"*comment déterminer la borne inférieure pour la grandeur de cet intervalle en dessous de laquelle on ne peut plus supposer que la série est une fraction rationnelle ?*"
On ne dispose pas de critère général permettant de répondre à cette question pour n'importe quelle série.

Nous avons donc tenté ici d'élaborer une première approche de ce problème.

La situation est la suivante :
on connait les $(n+1)$ premiers termes de f : 1 , μ_1 , ..., μ_n , et à la $r+1^{ième}$ étape, $\forall m \in [s_r+1, n]$, on a l'égalité $h^r_{m,0} = \mu_m$.
Que peut-on alors conclure ?

Notre idée est de comparer la largeur de ce dernier intervalle avec celle de l'intervalle de largeur maximum rencontrée au cours des précédentes étapes.

Remarque : l'intervalle de l'étape $k+1$ est $[s_k+1, s_k+p]$, si p est le premier entier tel que $h^k_{s_k+p,0} \neq \mu_{s_k+p}$; et sa largeur est p.

- Dans le cas où le dernier intervalle est plus grand que tous les autres, on accepte la fraction rationnelle construite.
- Dans le cas contraire, on peut estimer que l'on ne dispose pas d'assez d'informations pour conclure positivement.

Le choix de ce critère de décision a été inspiré du fait de la "régularité de décomposition en C-fraction" pour la plupart des séries que l'on manipule, en particulier en combinatoire.

La notion de "régularité de décomposition en C-fraction" traduit le fait que presque tous les éléments de la suite α sont égaux à 1 ou proche de 1, ce qui correspond à des intervalles pour la plupart réduits à 1 élément. (cf exemple ci-dessous)

D'où, la considération du fait que la décomposition en C-fraction est terminée lorsque le dernier intervalle est plus grand que les autres, et la supposition que l'on est en présence d'une fraction rationnelle.

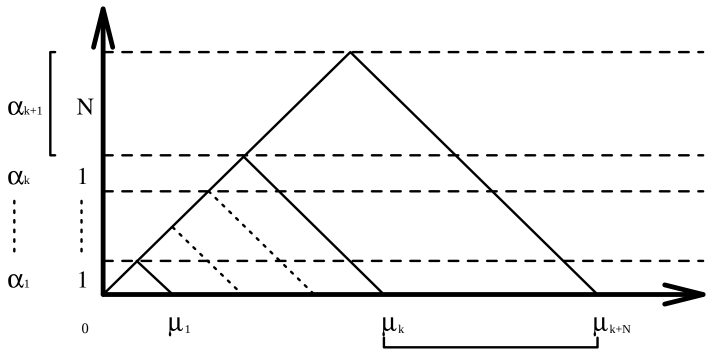
Exemple : la série des nombres de Fibonacci a pour fonction génératrice :

$$f = \cfrac{1}{1 - \cfrac{t}{1 - \cfrac{t}{1+t}}} = \frac{1}{1 - t - t^2}$$

où $\alpha = (1, 1, 1)$ et $\lambda = (-1, -1, 1)$.

Ce test peut bien sûr ne plus être adapté pour certains cas particuliers :

comme par exemple, une série dont la suite α de la décomposition commencerait par $\underbrace{1, 1, \ldots, 1}_{k}, N, 1, \ldots$ avec N grand devant 1.



Si le nombre de termes disponibles de la série est compris entre $k+2$ et N, le test aboutit à l'acceptation d'une fausse fraction rationnelle.

Ainsi, nous avons implémenté notre programme de façon à ce qu'un utilisateur averti puisse fixer lui-même la largeur minimale requise pour le dernier intervalle. Il aura donc la possibilité d'étudier des séries présentant certaines particularités.

3.3.2 Application et comparaisons

On a donc inclus notre algorithme dans le programme de Plouffe et Bergeron vu au chapitre I. Le programme utilise alors notre fonction au lieu de faire appel à la routine "convert/ratpoly" de MAPLE qui retourne la fraction rationnelle associée à une série.

Notre premier travail a été bien sûr d'appliquer le programme ainsi obtenu à chacune des séries des animaux 3D que l'on a calculées. Et ce fut malheureusement de nouveau un échec.
Ayant eu le même résultat avec le programme de Plouffe, l'échec de cette application ne permet pas de mesurer les performances de notre programme.

Ainsi, on a essayé d'effectuer quelques comparaisons entre les deux méthodes, en les appliquant à des séries bien connues.
Ces comparaisons s'appuient sur le nombre de termes de la série qui sont exigés pour calculer la fonction génératrice.

Exemples de "séries rationnelles"

On s'est tout d'abord penché sur le cas de séries admettant une fraction rationnelle comme fonction génératrice.

1 : Fibonacci

La série est $1+t+2t^2+3t^3+5t^4+O(t^5)$ et la fonction génératrice $\dfrac{1}{1-t-t^2}$.
Les deux programmes ont besoin des 5 premiers termes de la série pour déterminer cette fonction.

2 : Delmo

La série est

$$1{+}121t{+}12321t^2{+}1234321t^3{+}123454321t^4{+}12345654321t^5{+}1234567654321t^6{+}O(t^7)$$

et la fonction génératrice $\dfrac{1+10x}{(1-x)\,(1-10x)\,(1-100x)}$.
Dans les deux cas, les 7 premiers termes sont nécessaires pour trouver la fonction.

3 : Couper un gâteau en un maximum de parts (exemple 1 du chapitre I)

La série est $1 + 2t + 4t^2 + 7t^3 + 11t^4 + 16t^5 + 22t^6 + 29t^7 + 37t^8 + O(t^9)$
et la fonction génératrice $\dfrac{1 - x + x^2}{(x-1)^3}$.
Le programme de Plouffe nécessite la connaissance de 8 termes, et le notre la connaissance de 9 termes.

En fait, on a besoin d'un coefficient de plus à cause du test sur la largeur du dernier intervalle que l'on a décrit en 3.3.1. En effet, le développement en C-fraction de cette fonction est donnée par les deux suites α et λ suivantes :

α	:	1	2	1	1	1
λ	:	-2	$1/2$	-1	$1/2$	$-1/2$

On remarque que l'on doit avoir $1+2+1+1+1=6$ termes, plus le terme constant, ce qui fait 7 termes en tout. Mais notre programme ne détecte qu'il est en présence de la bonne fraction rationnelle que si le dernier intervalle est de largeur au moins 3 (puisque le maximum des autres est 2).
Ceci nous amène donc à 9 coefficients.

4 : Une dernière

On a construit une fraction rationnelle à partir des deux suites α et λ suivantes :

α	:	3	2	1
λ	:	-2	-3	-1

On a obtenu la fraction $\dfrac{1 - t - 3t^2}{1 - t - 3t^2 - 2t^3 + 2t^4}$ que l'on a développée en série pour tester les programmes.
La détermination de cette fonction demande 15 termes avec celui de Plouffe et 10 avec le notre.

Une première remarque est que le nombre de termes exigés par notre programme est augmenté de 3 par le test sur le dernier intervalle. Connaissant la décomposition en C-fraction, on s'aperçoit que 7 auraient suffi.
La deuxième remarque concerne le fait que 15 termes sont nécessaires au programme de Plouffe alors que 9 pour "convert/ratpoly" suffisent. Et ceci vient du test sur la longueur de la fraction trouvée par "convert/ratpoly" que Plouffe a rajouté dans son programme (voir chapitre I).

Exemples de "séries non rationnelles"

On traite ici des séries dont la fonction génératrice n'est pas une fraction rationnelle.

5 : Catalan

La série est $1 + t + t^2 + 2t^3 + 5t^4 + 14t^5 + O(t^6)$
et la fonction génératrice $\dfrac{1}{2} - \dfrac{1}{2}\sqrt{1-4t}$.
Les deux programmes ont besoin des 6 premiers termes de la série pour déterminer cette fonction.

C'est la dérivée de la dérivée logarithmique de la série qui est rationnelle :

$$\frac{2}{1-4t}$$

6 : Une fonction avec un exposant rationnel

La série est

$$1 + 24t + 270t^2 + 2240t^3 + 15750t^4 + 99792t^5 + 588588t^6 + 3294720t^7 + O(t^8)$$

et la fonction génératrice $\dfrac{1+10t+4t^2}{(1-4t)^{7/2}}$.

Pour retrouver cette fonction, le programme de Plouffe demande la connaissance de 12 termes et le notre celle de 8 termes.

C'est la dérivée logarithmique de la série qui est rationnelle :

$$-12\,\frac{2+9t+2t^2}{(4t-1)\,(1+10t+4t^2)}$$

7 : Une série exponentielle

La série ci-dessous est la série génératrice exponentielle des polynômes d'Hermite pour $t=-1$.

$$1 + x + x^2 + 2/3x^3 + 5/12x^4 + 13/60x^5 + O(x^6)$$

La fonction génératrice est $\exp\,(x+\dfrac{x^2}{2})$.

Le programme de Plouffe nécessite la connaissance de 5 termes, et le notre la connaissance de 4 termes.

C'est la dérivée logarithmique de la série qui est rationnelle :

$$1+x$$

Quelques remarques

Pour les cas 1, 2, et 5, il faut exactement le même nombre de termes aux deux programmes pour calculer la fonction génératrice.

Pour les cas 4, 6, et 7, le programme de Plouffe en demande plus que le notre, et pour le cas 3, c'est le contraire! ...

On n'a pas donné ici la liste de toutes les comparaisons que l'on a effectuées. Mais malgré tout, le nombre de séries que l'on a pu testées est vraiment faible par rapport au nombre total de celles qui sont connues et répertoriées dans [10].

Les résultats obtenus ici n'étant pas représentatifs, on ne peut donc pas tirer de conclusions fiables sur les performances de notre programme.

Cependant, le résultat de nos premières comparaisons semble prometteur, et il pourrait être vraiment très intéressant d'approfondir cette étude.

Conclusion

Le travail réalisé au cours de ce mémoire est intéressant de par les résultats et les conclusions auxquels il a aboutit.

Le fait d'avoir calculé les premiers termes des séries génératrices des animaux dirigés 3D constitue une base importante pour leur étude et la résolution des modèles de physique statistique associés.
Il serait cependant souhaitable de pouvoir obtenir un plus grand nombre de ces termes. On est actuellement limité à cause du temps excessif que prend l'exécution de l'algorithme que nous avons implémenté, basé sur la bijection animaux 3D / g-arbres.

Une solution idéale, évidemment, serait d'établir une nouvelle bijection avec des objets que l'on peut compter plus aisément, et surtout plus rapidement ...

Les intérêts présentés par la deuxième partie de ce mémoire ont dépassé le cadre des animaux 3D.
En effet, l'implémentation et l'application de l'algorithme de Viennot et Roblet [13] que nous avons effectuées pourraient tout à fait constituer une nouvelle approche dans le cadre plus général des approximations de séries génératrices ; c'est une étude qui mériterait d'être approfondie.

Pour ce qui est des animaux 3D, on a vu que des réserves pouvaient être émises quant à l'existence d'une formule close simple pour les fonctions génératrices des différents animaux.
Il pourrait être alors utile de tenter d'appliquer diverses méthodes d'analyse asymptotique aux animaux 3D, dans le but d'obtenir, à défaut de formules closes, des approximations asymptotiques pour les séries : ces résultats intéresseraient tout autant les physiciens, on l'a vu au travers de l'article de Guttmann [5].

En dernières remarques, je souhaite ici préciser les principaux intérêts personnels que m'a apportés la réalisation de ce mémoire.

- J'ai été amenée à me familiariser avec des logiciels de calculs formels : PARI, et puis surtout MAPLE. Ce dernier m'a vraiment été très utile pour la manipulation de séries.
- La plupart des problèmes rencontrés dans le domaine des animaux dirigés 3D sont encore à ce jour non résolus ; et travailler sur un tel sujet m'a permis de comprendre combien le travail de Recherche pouvait être à la fois si complexe et si captivant.

Bibliographie

[1] BAXTER R.J. : "Exactly solved models in statistical mechanics", Academic Press, New York (1982).

[2] BERGERON F. - PLOUFFE S. : "Computing the generating function of a series given its first terms", rapport de recherche n^o164, Université de Quebec à Montréal, 22 octobre 1991.

[3] BETREMA J. - PENAUD J.G. : "Modèles avec particules dures et séries en variables partiellement commutatives", en préparation.

[4] BETREMA J. - PENAUD J.G. : "Animaux et arbres guingois", rapport LaBRI n^o 90-60, Université Bordeaux I.

[5] BRAK R. - GUTTMANN A.J. : "Algebraic approximants : a new method of series analysis", J. Phys. A : Math. Gene. 23 (1990), L1331-L1337.

[6] CONIL P. : "Animaux dirigés", Mémoire de Dea, Université Bordeaux I (septembre 1990).

[7] CONWAY A. : "Algebraic approximants in statistical mechanics", supervisé par T. Guttmann, 15 novembre 1991.

[8] DELEST M. : "Polyominoes and animals : some recent results", J. of Math. Chemestry, n^o8, 1991, 3-18.

[9] PENAUD J.G. : "Arbres et Animaux", Mémoire d'habilitation, Université Bordeaux I (mai 1990).

[10] SLOANE N.J. : "A Handbook of integer sequences", Academic Press, New York (1979). (une nouvelle version doit paraître)

[11] VIENNOT X.G. : "Problèmes combinatoires posés par la physique statisque", Séminaire Bourbaki n^o626, $36^{ème}$ année, in Asterisque n^o121-122, (1985) 225-246, Soc. Math. France.

[12] VIENNOT X.G. : "A survey of polyominoes enumeration", Avril 1992, pour Colloque *Formal power series and algebraic combinatorics*, Montréal, juin 1992.

[13] VIENNOT X.G. - ROBLET E. : "Comment deviner qu'une série est rationnelle?", rapport interne du LaBRI n^o90-80.

[14] COHEN H. - BERNARDI D. - BATUT C. - OLIVIER M. : "User's guide to PARI-GP", version 1.36, décembre 1991.

[15] CHAR B.W. - GEDDES K.O. - GONNET G.H - MONAGAN M.B. - WATT S.M. : "Maple reference manual", $5^{ème}$ édition, mars 1988.