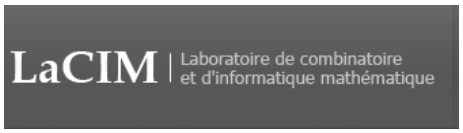
Rapport de Stage Ingénieur

Effectué au laboratoire de combinatoire et d'informatique mathématique (LaCIM, Montréal)

Pauline Hubert

Département Génie Mathématique
INSA de Rouen



Maître de stage : François Bergeron
Tuteur INSA : Bernard Gleyse

Mai à Octobre 2014

Résumé

Dans le cadre de ma formation à l'INSA de Rouen en Génie Mathématiques, j'ai effectué mon stage ingénieur au Laboratoire de Combinatoire et d'Informatique Mathématique de l'UQAM (Montréal). J'ai fait le choix d'un stage en recherche et plus précisément en combinatoire, pour faire suite aux cours suivis au master en informatique théorique (ITA, Université de Rouen). Ce stage a alors été constitué de recherche en combinatoire qui est l'étude des configurations des objets ou des méthodes permettant de compter des objets sur des ensembles finis. Et plus particulièrement, il a porté sur les fonctions symétriques.

Les fonctions symétriques sont un outils trés utilisé en combinatoire. Il existe entre autre une base des fonctions symétriques appelée la base des fonctions Schur qui est sûrement la base des fonctions symétriques la plus importante et la plus étudiée et c'est en grande partie sur cette base qu'a porté mon travail.

L'un des objectifs principaux de ce stage était d'écrire des programmes Maple permettant de manipuler et d'effectuer des calculs sur les fonctions symétriques. Un package Maple a déjà été développé par J. Stembridge et permet de manipuler les fonctions symétriques en utilisant les bases bien connues des fonctions symétriques. Le but était donc de partir des fonctions déjà existantes et de les utiliser pour en écrire d'autres en particulier sur l'évaluation plethystique des fonctions symétriques et sur les fonctions Schur gauches.

Une fois écrites, ces fonctions ont permis d'étudier l'évaluation plethystique de fonctions Schur de formes données sur des produits ou des sommes d'alphabets et d'en donner des formules explicites. Cette partie m'a amenée à étudier également les coefficients de Littlewood-Richardson qui interviennent entre autre dans la multiplication des fonctions Schur et dans le développement des fonctions Schur gauches dans la base des fonctions Schur classiques.

Enfin, dans la dernière partie de ce stage, je me suis intéressée aux fonctions Schur gauches. Tout d'abord, les fonctions Schur gauches ont été ajoutées aux fonctions manipulables en Maple, c'est-à-dire que l'on peut les intégrer dans les calculs plethystiques ou les exprimer facilement dans n'importe quelle base des fonctions symétriques, comme n'importe quelle autre fonction. Ensuite, le second objectif était d'étudier la décomposition d'une fonction Schur gauche en une somme de deux fonctions Schur gauches. Le cas de fonctions Schur gauche non connectées a déjà été résolu et a donc pu être implémenté.

Différentes étapes de travail ont été nécessaires pour chaque partie. Tout d'abord, un travail de documentation c'est-à-dire de lecture et d'étude d'articles, de notes de cours, etc., ensuite un travail de programmation et enfin l'expérimentation et la démonstration des formules.

Remerciements

Je souhaite remercier François Bergeron, mon maître de stage, qui m'a permis d'effectuer ce stage au LaCIM, pour son aide et ses conseils tout au long de ce stage.

J'aimerais aussi remercier les membres du LaCIM que j'ai pû côtoyer pendant mon stage pour leur accueil chaleureux au sein du laboratoire.

Table des matières

Introduction

Le stage de fin d'études présenté dans ce rapport s'incrit dans le cadre de ma formation d'ingénieur au département Génie Mathématiques de l'INSA de Rouen et fait également suite à un master en Informatique Théorique et Applications (ITA) à l'Université des Sciences et Techniques de Rouen.

J'ai effectué mon stage du 1er mai au 31 octobre 2014 au Laboratoire de Combinatoire et d'Informatique Mathématique (LaCIM) de l'Université du Québec à Montréal (UQAM) avec François Bergeron, où j'ai pu faire de la recherche sur les fonctions symétriques.

Ce rapport a pour but de présenter le travail réalisé pendant ce stage ainsi après une brève présentation de l'organisme d'accueil et des objectifs du stage, les chapitres 2 et 3 seront consacrés aux principaux résultats déjà établis sur les fonctions symétriques et en particulier sur les fonctions Schur et leur multiplication qui seront utiles tout au long de ce rapport.

Dans le chapitre 4, après une rapide présentation du package Maple SF de J. Stembridge, on abordera la programmation Maple réalisée dans le but de fournir de nouvelles fonctions permettant de manipuler les fonctions symétriques, en se basant sur le package de J. Stembridge.

Enfin, dans le chapitre 5, on présentera le travail effectué sur l'évaluation plethystique des fonctions Schur qui permet de mettre en évidence certaines formules sur ces fonctions ainsi que de valider le travail de programmation effectué auparavant. Et pour finir, le chapitre 6 concerne les fonctions Schur gauches et plus particulièrement la décomposition d'une fonction Schur gauche en une somme de deux Schur gauches.

Chapitre 1

Contexte

1.1 Présentation de l'organisme d'accueil

1.1.1 UQAM : Université du Québec à Montréal

Montréal est la plus grande ville du Québec et compte avec sa banlieue plus de 3 millions d'habitants. Elle compte également plus de 170 000 étudiants répartis essentiellement dans quatre grandes universités, dont deux sont anglophones et deux sont francophones.

L'Université du Québec à Montréal (UQÀM) est l'une des universités francophones de Montréal mais c'est surtout la plus grande université publique du Québec.

L'UQAM a été créé le 9 avril 1969 par le gouvernement du Québec. Jusque dans les années 1960, à Montréal, la plupart des étudiants suivaient une formation universitaire anglophone et toutes les universités étaient privées. C'est dans le but d'élargir l'offre de formation francophone à Montréal et de créer une université publique que l'UQAM a vu le jour en 1969.

Le campus de l'UQAM a été inauguré en 1979 mais son développement s'est poursuivit ensuite jusqu'en 1999. Le campus est actuellement reparti en deux sites principaux. Le premier est situé dans le Quartier latin, à proximité de la Grande Bibliotheque du Québec et le deuxième est situé dans le Quartier des spectacles.

L'UQAM offre aujourd'hui à plus de 40 000 étudiants environ 300 programmes d'études différents répartis sur les trois cycles et comptait en 2012 plus de 1100 professeurs et plus de 2300 chargés de cours.

En 2013, l'UQAM compte 7 facultés qui sont subdivisées en 40 départements et écoles dont le département de Mathématiques et celui d'Informatique de la faculté des sciences auxquels est associé le laboratoire de combinatoire et d'informatique mathématique. C'est également l'une des plus importantes universités en termes de montant alloué à la recherche.

L'UQAM fait partie du réseau de l'UQ (Université du Québec) qui est un réseau d'établissements publiques répartis sur l'ensemble du territoire québecois. Le caractère public de ces universités est particulièrement mis en avant.

1.1.2 LaCIM : Laboratoire de combinatoire et d'informatique mathématique

Le LaCIM est un centre institutionnel de recherche associé aux départements de Mathématiques et d'Informatique de l'UQAM. Il s'agit de l'un des plus importants en Amérique du Nord. Il est l'un des huit laboratoires associé Centre de Recherche Mathématiques (CRM). Le LaCIM constitue également le volet québécois de LIRCO (Laboratoire International Franco-Québécois de Recherche en Combinatoire) associé au CNRS. Enfin, il est associé à l'UMI (Unité Mixte Internationale) du CNRS-CRM.

Le LaCIM regroupe une quarantaines de chercheurs, ainsi que des étudiants et des stagiaires. L'actuel directeur du LaCIM est Srecko Brlek. Les locaux du LaCIM sont situés au Pavillon Kennedy, dans le quartier des spectacles au centre ville de Montréal.

Le LaCIM a été créé il y a environ 25 ans avec à ses débuts la Théorie des Espèces de Structures comme axe principal de recherche. Cette théorie permet d'unifier et d'approfondir la notion de constructions combinatoires comme celles d'arbres, de graphes, de permutations, ou encore de mots. Les recherches au LaCIM se sont par la suite grandement diversifiées et mêlent plusieurs disciplines dont l'algèbre, la combinatoire, l'informatique mathématique, la bio-informatique et l'analyse d'algorithmes.

FIGURE 1.1 – Pavillon Kennedy (Place des Arts, Montréal)

1.2 Présentation du stage

Ce stage porte sur la combinatoire, qui est l'étude des configurations des objets ou des méthodes permettant de compter des objets sur des ensembles finis, et a pour but d'explorer, au moyen d'outils de calcul formel tels que Maple ou Sage, les propriétés d'objets mathématiques.

Un des outils utilisé en combinatoire algébrique est les fonctions symétriques. Il s'agit de fonctions invariantes par permutation de leurs variables. Ces fonctions sont également utilisées dans d'autres domaines comme en théorie de la représentation du groupe symétrique.

La complexité de certains calculs sur ces fonctions rend essentielle l'utilisation et le développement d'outils de calcul adaptés. Le but de ce stage est donc double. D'abord de mettre au point les outils nécessaires, puis dans un deuxième temps, de mettre en évidence des propriétés intéressantes par une exploration expérimentale au moyen des outils développés.

Cela se concrétise tout d'abord par l'étude des fonctions symétriques et des notions liées telles que les partitions, les tableaux de Young et le plethysme ; puis par le développement en Maple des outils de calcul formel nécessaires, notamment concernant le plethysme, en se basant sur les travaux déjà effectués ; et enfin, par l'utilisation de ces outils pour explorer les propriétés de l'évaluation plethystique sur des fonctions symétriques particulières, les fonctions Schur.

Chapitre 2

Fonctions symétriques

Sont présentés dans cette partie les définitions et théorèmes concernant les fonctions symétriques utiles pour comprendre le travail effectué par la suite et que l'on peut retrouver dans [3, 4, 6].

2.1 Définition des fonctions symétriques

Définition 2.1.1 *Soit $f(x_1, x_2, ...x_n) \in K[x_1, x_2, ...x_n]$, une fonction de plusieurs variables sur un corps K. Et soit $\sigma \in \mathbb{S}_n$, une permutation sur n éléments. On note alors*

$$\sigma f = f(x_{\sigma_1}, x_{\sigma_2}, ...x_{\sigma_n})$$

Et on dit que f est une ***fonction symétrique*** *si et seulement si $\sigma f = f \ \forall \sigma \in \mathbb{S}_n$.*

f est donc une fonction symétrique si et seulement si sa valeur ne change par lorsque l'on permute l'ordre de ses variables. On considère Λ, l'ensemble des polynômes symétriques dans $\mathbb{Q}[\mathbf{x}]$ où $\mathbf{x} = (x_1, x_2, ...x_n)$.

Proposition 2.1.1 *Soient $f(\mathbf{x})$ et $g(\mathbf{x}) \in \Lambda$, on a alors*
i) $f(\mathbf{x}) + g(\mathbf{x}) \in \Lambda$
ii) $f(\mathbf{x})g(\mathbf{x}) \in \Lambda$
iii) $c \in \Lambda$ *si* $c \in \mathbb{Q}$

On considère également l'ensemble des polynômes symétriques ayant tous leurs termes de degré d. On dit qu'ils sont homogènes de degré d et on note Λ_d l'ensemble de ces polynômes. Et on a que :

$$\Lambda = \bigoplus_d \Lambda_d$$

Notation : Soit λ tel que $\lambda = \lambda_1\lambda_2\cdots\lambda_k$ avec $\lambda_i \in \mathbb{N} \ \forall i$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0$ et $\lambda_1 + \lambda_2 + \cdots + \lambda_k = n$. On dit alors que λ est une partition de n et on note $\lambda \vdash n$. La taille de λ est $|\lambda| = n$ et sa longueur est $l(\lambda) = k$.

On définit les ***fonctions monomiales***, notées m_λ, indexées par des partitions $\lambda = \lambda_1\lambda_2\cdots\lambda_k$, comme étant la somme de tous les monômes de degré n, avec $\lambda \vdash n$.
Soit $\alpha = (a_1, a_2, ..., a_n)$ où $a_i \in \mathbb{N} \ \forall i$, on a alors $\mathbf{x}^\alpha = \prod_i x_i^{a_i}$. Soit $\tau(\alpha)$, les valeurs de α triées dans l'ordre décroissant et sans les 0, alors $m_0 = 1$ et

$$m_\lambda = \sum_{\tau(\alpha)=\lambda} \mathbf{x}^\alpha$$

Exemples :
$m_2(x,y,z) = x^2 + y^2 + z^2$
$m_{22}(x,y,z) = 2x^2y^2 + 2y^2z^2 + 2x^2z^2$
$m_{31}(x,y,z) = x^3y + xy^3 + x^3z + xz^3 + y^3z + yz^3$
$m_{1111}(x,y,z) = 0$

Notons que $m_\lambda = 0$ quand $l(\lambda) > n$ où n est le nombre de variables. On peut également remarquer que $\{m_\lambda \mid \lambda \vdash d\}$ engendre Λ_d.

Il existe d'autres polynômes symétriques classiques qui sont les fonctions homogènes complètes (h_n), les fonctions élémentaires (e_n) et les fonctions sommes de puissances (p_n).

Les ***fonctions homogènes complètes*** correspondent à la somme de tous les monomiales d'un degré donné :

$$h_n = \sum_{\mu \vdash n} m_\mu$$

Notons que $h_0 = 1$ et que les fonctions homogènes complètes sont multiplicatives, c'est-à-dire que pour $\lambda = \lambda_1, \lambda_2, \cdots, \lambda_k$, $h_\lambda = h_{\lambda_1} h_{\lambda_2} \cdots h_{\lambda_k}$.

Exemples :
$h_2(x,y,z) = x^2 + y^2 + z^2 + xy + xz + yz$

$$\begin{aligned} h_{21}(x,y,z) &= h_2(x,y,z)h_1(x,y,z) \\ &= (x^2 + y^2 + z^2 + xy + xz + yz)(x + y + z) \\ &= x^3 + y^3 + z^3 + 2x^2y + 2xy^2 + 2y^2z + 2yz^2 + 2x^2z + 2xz^2 + 3xyz \end{aligned}$$

Les ***fonctions élémentaires*** correspondent à la somme de tous les monômes de degré n où chaque variable apparaît au plus une fois par monôme. Et on a $e_0 = 1$. Les fonctions élémentaires sont également multiplicatives. On remarque que $e_n = m_{1^n}$.

Exemples :
$e_2(x,y,z) = xy + yz + xz$
$e_3(x,y,z) = xyz$
$e_{21}(x,y,z) = e_2(x,y,z)e_1(x,y,z) = (xy + xz + yz)(x + y + z)$

Enfin, les ***fonctions sommes de puissances*** correspondent à la somme de toutes les variables élevées à une puissance donnée : $p_0 = 1$ et

$$p_n(\mathbf{x}) = \sum_i x_i^n$$

De plus $p_n = m_n$ $(n \in \mathbb{N})$, et les fonctions sommes de puissances sont également multiplicatives.

Exemples :
$p_6(x,y) = x^6 + y^6$
$p_4(x,y,z) = x^4 + y^4 + z^4$
$p_{322}(x,y,z) = (x^3 + y^3 + z^3)(x^2 + y^2 + z^2)^2$

Théorème 2.1.1 (Newton - Théorème fondamental des fonctions symétriques) *Toute fonction symétrique peut-être écrite comme un polynôme en les fonctions symétriques élémentaires e_k, $k = 1, 2, \cdots$.*
C'est-à-dire que $\{e_\mu\}_{\mu \vdash d}$ est une base de Λ_d.

2.2 Définition des fonctions anti-symétriques

2.2.1 Fonctions anti-symétriques

Définition 2.2.1 *Soit $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, ...x_n)$, une fonction de plusieurs variables. Et soit $\sigma \in \mathbb{S}_n$, une permutation sur n éléments.*

On dit que f est une fonction ***antisymétrique*** *si et seulement si*

$$\sigma f(\mathbf{x}) = (-1)^{\text{inv}(\sigma)} f(\mathbf{x}) \ \forall \sigma \in \mathbb{S}_n$$

Où $\text{inv}(\sigma)$ *est le nombre d'inversions de σ ie* $\text{inv}(\sigma) = \#\{i \mid i > j \text{ et } \sigma(i) < \sigma(j)\}$.

Soit $f_a(\mathbf{x}) := \det \begin{pmatrix} x_1^{a_1} & x_1^{a_2} & \cdots & x_1^{a_n} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{a_1} & x_n^{a_2} & \cdots & x_n^{a_n} \end{pmatrix}$ avec $a_1 < a_2 < \cdots < a_n \leq 0$

Le déterminant de Vandermonde correspond au plus petit déterminant non nul $f_a(\mathbf{x})$:

$$\Delta_n(\mathbf{x}) := \det \begin{pmatrix} x_1^{n-1} & \cdots & x_1^0 \\ \vdots & \ddots & \vdots \\ x_n^{n-1} & \cdots & x_n^0 \end{pmatrix} = \prod_{1 \leq i < j \leq n} (x_i - x_j)$$

Théorème 2.2.1 *$\Delta_n(\mathbf{x})$ divise sans reste tout polynôme antisymétrique et de plus, le résultat de cette division est un polynôme symétrique.*

Preuve
On peut tout d'abord remarquer que tout polynôme anti-symétrique $f(\mathbf{x})$ s'annule si deux de ses variables sont égales. En effet, soient i et j tels que $x_i = x_j$ et soit la permutation σ qui inverse x_i et x_j et ne fait rien d'autre. On a $\text{inv}(\sigma) = 1$, donc $\sigma f = -f$ puisque f est antisymétrique mais on a aussi $\sigma f = f$ puisque $x_i = x_j$, donc finalement f est nulle.
On en déduit donc que $(x_i - x_j)$ divise $f(\mathbf{x})$ pour tout couple (i, j), donc $\prod_{1 \leq i < j \leq n} (x_i - x_j) = \Delta_n(\mathbf{x})$ divise $f(\mathbf{x})$.
De plus, $\dfrac{f(\mathbf{x})}{\Delta_n(\mathbf{x})}$ est symétrique car $\forall \sigma$:

$$\sigma . \frac{f(\mathbf{x})}{\Delta_n(\mathbf{x})} = \frac{\sigma . f(\mathbf{x})}{\sigma . \Delta_n(\mathbf{x})} = \frac{(-1)^{\text{inv}(\sigma)} f(\mathbf{x})}{(-1)^{\text{inv}(\sigma)} \Delta_n(\mathbf{x})} = \frac{f(\mathbf{x})}{\Delta_n(\mathbf{x})}$$

□

2.2.2 Fonctions Schur

On définit les fonctions Schur de la façon suivante :

Définition 2.2.2 *Soient* $a = (a_1, a_2, \cdots, a_n)$, $\delta = (n-1, n-2, \cdots, 0)$ *et* λ *un partage* ($\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_k, 0, \cdots, 0)$, *avec* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = 0$). *Les fonctions Schur sont définies par :*

$$s_\lambda = \frac{f_{\delta+\lambda}(\mathbf{x})}{\Delta(\mathbf{x})} = \frac{\det(x_i^{\lambda_j+n-j})_{1\leq i,j\leq n}}{\prod\limits_{i<j}(x_i - x_j)}$$

$\{s_\lambda | \lambda \vdash d\}$ est également une base de Λ_d.

2.3 Approche combinatoire des fonctions Schur

2.3.1 Diagrammes et tableaux de Young

Avant d'aborder la définition combinatoire des fonctions Schur, il est utile de définir les diagrammes et les tableaux de Young qui nous seront également utiles par la suite.

Définition 2.3.1 *Un* ***diagramme de Young****, également appelé diagramme de Ferrer, est un ensemble de cases contiguës, alignées à gauche et dont le nombre de cases par ligne décroit du bas vers le haut.*

On utilise ici décroit au sens large, c'est-a-dire que la ligne i possède un nombre de cases inférieur ou égal à la ligne $i+1$, pour tout i.

Notons également que l'on a choisi d'utiliser ici la notation française. Les lignes du diagramme peuvent également être ordonnées dans le sens décroissant du haut vers le bas, il s'agit alors de la notation anglaise.

Les diagrammes de Young permettent de représenter les partitions. En effet, le nombre de cases de la ligne i correspond au $i^{\text{ème}}$ terme λ_i de la partition λ.

Par abus de langage, on notera également λ le diagramme associé à la partition λ.

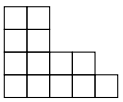


FIGURE 2.1 – Diagramme de Young de la partition $\lambda = 5, 4, 2, 2$

Le conjugué d'une partition peut d'ailleurs facilement s'obtenir à partir du diagramme de Young de la partition en comptant le nombre de cases des colonnes de gauche à droite et non plus des lignes. Ainsi, le conjugué de λ de la figure précédente est $\lambda' = 4, 4, 2, 2, 1$.

Un tableau de Young est un diagramme de Young dont on rempli les cases avec des entiers. On distingue deux type de tableaux de Young, les tableaux standards et les tableaux semi-standard.

Définition 2.3.2 *Un* ***tableau de Young standard*** *est un remplissage d'un diagramme de Young dont les entrées sont les entiers de 1 à n où n est le nombre de cases total du tableau. Toutes les valeurs des entrées sont donc distinctes deux à deux et elles sont strictement croissantes sur les lignes de gauche à droite et sur les colonnes de bas en haut.*

9	13			
7	8			
3	4	6	12	
1	2	5	10	11

FIGURE 2.2 – Exemple de remplissage de $\lambda = 5, 4, 2, 2$ donnant un tableau standard

Définition 2.3.3 *Un **tableau de Young semi-standard** est un remplissage d'un diagramme de Young dont les entrées sont les entiers de 1 à n où n est inférieur ou égal au nombre de cases total du tableau. Les entrées sont croissantes sur les lignes et strictement croissantes sur les colonnes.*

6	6			
3	5			
2	3	3	4	
1	1	2	3	4

FIGURE 2.3 – Exemple de remplissage de $\lambda = 5, 4, 2, 2$ donnant un tableau semi-standard

Nous aurons également besoin par la suite de la définition de tableau gauche.

Définition 2.3.4 *Soit λ et μ deux partitions telles que le diagramme de Young de μ soit inclus dans celui de λ (on note $\mu \subseteq \lambda$), alors λ/μ est une **forme gauche** (*skew shape *en anglais). Le **diagramme gauche** associé à λ/μ contient les cases du diagramme de λ qui ne sont pas dans le diagramme de μ.*

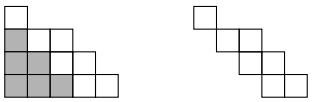


FIGURE 2.4 – Exemple de diagramme gauche : $\lambda = 5, 4, 3, 1$ (a gauche), $\mu = 3, 2, 1$ (en gris à gauche) et λ/μ (à droite)

De la même façon que pour les diagrammes de Young, on peut remplir un diagramme gauche pour obtenir un tableau gauche. Remarquons également que si μ est la partition vide, le diagramme associé à λ/μ est un diagramme de Young classique.

Soit ν une partition. On dit que l'on effectue un remplissage du tableau T de poids ν si le nombre de 1 dans le remplissage de T est ν_1, le nombre de 2 est ν_2, ...

Les fonctions `plot_partition` et `plot_partitionGauche` que l'on peut trouver en annexe A, permettent d'afficher dans Maple respectivement des partitions et des formes gauches.

2.3.2 Tableaux semi-standards et fonctions Schur

On peut définir combinatoirement les polynômes de Schur à l'aide des tableaux semi-standards.

Soit λ un tableau semi-standard dont le remplissage se fait donc en étant croissant sur les lignes de gauche à droite et strictement croissant sur les colonnes de bas en haut, $\lambda \subseteq \mathbb{N} \times \mathbb{N}$. (Voir figure 2.5)

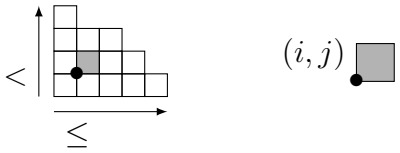


FIGURE 2.5 – Exemple de forme d'un tableau semi-standard

Soit $\tau : \lambda \to \mathbb{N}$ une fonction de remplissage qui à chaque case (i, j) de λ associe un entier.

Par définition, on a les relations suivantes :

$$\tau(i, j) < \tau(i, k) \text{ si } j < k$$
$$\tau(i, j) \leq \tau(l, k) \text{ si } i < l$$

De plus, on note $\mathbf{x}_\tau = \prod_{(i,j)\in\lambda} x_{\tau(i,j)}$. $\mathbf{x}_\tau$ est l'évaluation monomiale de τ.

On définit alors les polynômes de Schur de la façon suivante :

$$s_\lambda(\mathbf{x}) := \sum_{\tau:\lambda\to\mathbb{N}} \mathbf{x}_\tau$$

On peut également décrire les fonctions Schur dans la base des fonctions monomiales à l'aide des tableaux semi-standards. Considérons $a < b < c$ et tous les tableaux semi-standards de forme 21. On obtient alors les tableaux suivants :

b		b		c		b	
a	a	a	b	a	b	a	c

FIGURE 2.6 – Tableaux semi-standards de forme 21

La figure 2.6 montre que l'on peut écrire la fonction Schur s_{21} dans la base des monomiales : $s_{21} = m_{21} + 2m_{111}$. On retrouve m_{21} dans les deux premiers tableaux ($a^2b + b^2a$) et chacun des deux derniers tableaux correspondent à m_{111} (abc).

On exprime alors les polynômes de Schur dans la base des monomiales :

$$s_\lambda(\mathbf{x}) = \sum_{\mu\vdash d} K_{\lambda\mu} m_\mu(\mathbf{x})$$

Les $K_{\lambda\mu}$ sont des entiers appelés nombres de Kostka.

Exemples :
$s_4 = m_4 + m_{31} + m_{22} + m_{211} + m_{1111}$
$s_{31} = m_{31} + m_{22} + 2m_{211} + 3m_{1111}$
$s_{22} = m_{22} + m_{211} + m_{1111}$
$s_{1111} = m_{1111}$

On peut également remarquer que pour $n \in \mathbb{N}$:

$$s_n = h_n \quad \text{et} \quad s_{\underbrace{111\cdots1}_{\text{n fois}}} = e_n$$

2.4 Calcul plethystique

Soit f une fonction symétrique exprimée à l'aide des bases définies ci-dessus. On considère maintenant f comme un opérateur sur des expressions telles que X, Y, A et B où $X = x_1 + x_2 + x_3 + \cdots$, $Y = y_1 + y_2 + y_3 + \cdots$ et A et B sont des expressions plus générales, par exemple $A = \dfrac{X}{1-q}$.

On note alors $f[A]$ **l'évaluation plethystique** de f sur A.

Proposition 2.4.1 *On a les règles de calcul suivantes :*

i) $(f+g)[A] := f[A] + g[A]$
ii) $(f.g)[A] := f[A].g[A]$
iii) $c[A] := c$, *si* c *est une constante*
iv) *Calculer* $p_k[X]$, *revient à remplacer toutes les variables dans* X *par leur* $k^{\text{ième}}$ *puissance et*
   a) $p_k[A+B] := p_k[A] + p_k[B]$
      $p_k[A-B] := p_k[A] - p_k[B]$
   b) $p_k[A \times B] := p_k[A] \times p_k[B]$
      $p_k[A \div B] := p_k[A] \div p_k[B]$
   c) $p_k[x] = x^k$, *si* x *est une variable*
   d) $p_k[c] = c$, *si* c *est une constante*

On en déduit par exemple : $p_k[e^x] = e^{x^k}$ puisque e^x peut s'exprimer comme une somme. (e correspond ici à la fonction exponentielle.)

On peut remarquer que $f(X) = f[X]$ pour f une fonction symétrique et X une somme de variables aussi appelé alphabet. De plus, lorsqu'il n'y a pas d'ambiguité, on ne notera pas l'alphabet afin d'alléger l'écriture.

Il existe plusieurs théorèmes sur l'évaluation plethystique des fonctions symétriques. On utilisera souvent, entre autres, les formules de Cauchy et leurs formes duales.

Théorème 2.4.1 (Formules de Cauchy)

$$h_n[X \cdot Y] = \sum_{\mu \vdash n} m_\mu(X) h_\mu(Y) \tag{2.1}$$

$$h_n[X \cdot Y] = \sum_{\mu \vdash n} s_\mu(X) s_\mu(Y) \tag{2.2}$$

Théorème 2.4.2 (Formules de Cauchy duales)

$$e_n[X \cdot Y] = \sum_{\mu \vdash n} m_\mu(X) e_\mu(Y) \tag{2.3}$$

$$e_n[X \cdot Y] = \sum_{\mu \vdash n} s_\mu(X) s_{\mu'}(Y) \tag{2.4}$$

En utilisant le plethysme, on obtient également un moyen de déterminer si deux bases sont duales.

Proposition 2.4.2 *$\{f_\mu\}_{\mu\vdash n}$ est une **base duale** de $\{g_\mu\}_{\mu\vdash n}$ si et seulement si*

$$h_n[X \cdot Y] = \sum_{\mu\vdash n} f_\mu(X) g_\mu(Y)$$

On déduit du théorème 2.4.1 et de la proposition 2.4.2 que $\{m_\mu\}_{\mu\vdash n}$ est une base duale de $\{h_\mu\}_{\mu\vdash n}$.

De plus, $\{p_\mu\}_{\mu\vdash n}$ et $\{\frac{p_\mu}{z_\mu}\}_{\mu\vdash n}$ sont des bases duales, où $z_\mu = 1^{d_1} d_1! 2^{d_2} d_2! \cdots n^{d_n} d_n!$ avec μ un partage ayant d_i parts de taille i.

En utilisant la proposition 2.4.2 et en exprimant h_n uniquement sur X, on obtient que :

$$h_n = \sum_{\mu\vdash n} \frac{p_\mu}{z_\mu} \tag{2.5}$$

On a l'équivalence entre l'équation (2.5) et l'équation suivante :

$$\sum_{n\geq 0} h_n t^n = \exp\left(\sum_{k\geq 0} p_k t^k/k\right) \tag{2.6}$$

On a également une formule similaire à (2.5) pour e_n :

$$e_n = \sum_{\mu\vdash n} (-1)^{n-l(\mu)} \frac{p_\mu}{z_\mu} \tag{2.7}$$

où $l(\mu)$ est le nombre de parts de μ.

Et on a l'équivalence entre l'équation (2.7) et l'équation suivante :

$$\sum_{n\geq 0} e_n t^n = \exp\left(\sum_{k\geq 0} (-1)^{k-1} p_k t^k/k\right) \tag{2.8}$$

Voyons maintenant la formule de Jacobi-Trudi qui permet de calculer $s_\lambda[X]$.

Théorème 2.4.3 (Formule de Jacobi-Trudi)

$$s_\lambda = \det \begin{pmatrix} h_{\lambda_1} & h_{\lambda_1+1} & \cdots & h_{\lambda_1+k-1} \\ h_{\lambda_2-1} & h_{\lambda_2} & \cdots & h_{\lambda_2+k-2} \\ \vdots & & \ddots & \vdots \\ h_{\lambda_k-k+1} & \cdots & h_{\lambda_k-1} & h_{\lambda_k} \end{pmatrix} \tag{2.9}$$

où $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_k)$, $h_0 = 1$ et $h_{-k} = 0$ pour $k < 0$.

On peut aussi l'écrire de la façon suivante :

$$s_\lambda = \det(h_{\lambda_i+j-i})_{1\leq i,j\leq k} \tag{2.10}$$

Exemple : $s_{421} = \det \begin{pmatrix} h_4 & h_5 & h_6 \\ h_1 & h_2 & h_3 \\ 0 & 1 & h_1 \end{pmatrix}$

La formule de Jacobi-Trudi (2.4.3), ainsi que les équations (2.6) et (2.8) permettent d'effectuer des calculs sur les fonctions Schur, par exemple, on peut calculer $s_\lambda[u-1]$.

En effet, d'après 2.6 :

$$\begin{aligned}\sum_{n\geq 0} h_n[u-1]t^n &= e^{\sum\limits_{k\geq 0}\frac{p_k[u-1]}{k}t^k} = e^{\sum\limits_{k\geq 0}\frac{u^k-1}{k}t^k} \\ &= exp(-log(1-ut)+log(1-t)) = \frac{1-t}{1-ut}\end{aligned}$$

Ainsi, $\sum\limits_{n\geq 1} h_n[u-1]t^n + \underbrace{h_0[u-1]}_{=1} = \sum\limits_{n\geq 1}(u^n-u^{n-1})t^n + 1$

On a donc $h_n[u-1] = u^n - u^{n-1}$, $\forall n \geq 1$ et $h_0[u-1] = 1$ et d'après 2.4.3 :

$$s_\lambda[u-1] = \det\begin{pmatrix} h_{\lambda_1}[u-1] & h_{\lambda_1+1}[u-1] & \cdots & h_{\lambda_1+k-1}[u-1] \\ h_{\lambda_2-1}[u-1] & h_{\lambda_2}[u-1] & \cdots & h_{\lambda_2+k-2}[u-1] \\ \vdots & & \ddots & \vdots \\ h_{\lambda_k-k+1}[u-1] & \cdots & h_{\lambda_k-1}[u-1] & h_{\lambda_k}[u-1] \end{pmatrix}$$

On remplace alors $h_n[u-1]$ par la valeur trouvé précédemment :

$$s_\lambda[u-1] = \det\begin{pmatrix} u^{\lambda_1}-u^{\lambda_1-1} & u^{\lambda_1+1}-u^{\lambda_1} & \cdots & u^{\lambda_1+k-1}-u^{\lambda_1+k-2} \\ u^{\lambda_2-1}-u^{\lambda_2-2} & u^{\lambda_2}-u^{\lambda_2-1} & \cdots & u^{\lambda_2+k-2}-u^{\lambda_2+k-3} \\ \vdots & & \ddots & \vdots \\ u^{\lambda_k-k+1}-u^{\lambda_k-k} & \cdots & u^{\lambda_k-1}-u^{\lambda_k-2} & u^{\lambda_k}-u^{\lambda_k-1} \end{pmatrix}$$

Il est clair que ce déterminant est presque toujours nul puisque chaque colonne est égale à u fois la précédente sauf dans quelques cas particuliers. En effet, quand λ est une partition de la forme $\lambda = (n-k, 1^k)$, c'est-à-dire un nombre $n-k$ suivi de k 1, le déterminant n'est pas nul. Dans ce cas, il est de la forme :

$$s_{(n-k,1^k)}[u-1] = \det\begin{pmatrix} u^{n-k}-u^{n-k-1} & u^{n-k+1}-u^{n-k} & \cdots & u^n-u^{n-1} \\ 1 & u-1 & \cdots & u^{n-1}-u^{n-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 1 & u-1 \end{pmatrix}$$

Et on trouve que $s_{(n-k,1^k)}[u-1] = (-1)^k(u^{n-k}-u^{n-k-1})$.

Finalement, on a donc :

$$s_\lambda[u-1] = \begin{cases} (-1)^k(u^{n-k}-u^{n-k-1}) & \text{si } \lambda = (n-k,1^k) \\ 0 & \text{sinon} \end{cases} \tag{2.11}$$

□

Chapitre 3

Multiplication des fonctions Schur

3.1 Jeu de Taquin

Nous allons évoquer dans cette partie le Jeu de Taquin sur les tableaux gauches et définir ainsi une équivalence sur les tableaux tel que décrit dans [7].

Le Jeu de Taquin est un ensemble de règles qui permettent de transformer des tableaux gauches en considérant leurs cases comme des entités indépendentes que l'on peut déplacer les unes par rapport aux autres. Les règles sont définies telles que la transformation préserve la propriété d'être un tableau gauche.

La première étape d'un Jeu de Taquin consiste à déterminer les cases qui peuvent être "ajoutées" à un tableau gauche.

Soit λ/μ une forme gauche. On considère les cases b qui peuvent être ajoutées à λ/μ c'est-à-dire telles qu'elles partagent au moins un côté avec une case de λ/μ et que $\{b\} \cup \lambda/\mu$ soit une forme gauche valide.

On différencie alors deux types de cases. Celles qui se trouvent au-dessus ou à droite d'une case du diagramme de départ, on les appelent les cases supérieures droites et celles qui se trouvent au-dessous ou à gauche d'une case du diagramme de départ et qu'on appelle les cases inférieures gauches.

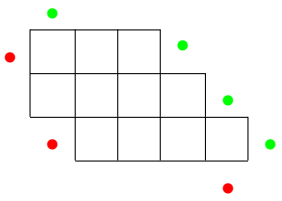


FIGURE 3.1 – Un exemple de diagramme gauche avec ses cases supérieures droites (•) et inférieures gauches (•)

On va maintenant décrire des règles qui permettent d'effectuer un glissement dans un jeu de Taquin. Précisons tout d'abord qu'un glissement est une série de mouvements qui peut démarrer de n'importe quelle case ajoutée b, supérieure ou inférieure. On note alors $jdt_b(T)$ la transformation du tableau T obtenue par glissement de jeu de Taquin à partir de la case b.

Partant d'une case supérieure droite marquée • du tableau T, il existe au moins une case b_1 de T qui soit adjacente à •. S'il en existe deux, b_1 sera celle contenant le plus petit entier. On glisse alors b_1 à la place de •, et • devient alors l'ancien emplacement de b_1. On regarde ensuite la case qui se trouve au-dessous de • et celle à sa gauche et on choisit b_2 celle qui contient le plus petit entier, on glisse alors b_2 à la place de •. On recommence le processus jusqu'à ce qu'il n'y ait plus de mouvement possible, c'est-à-dire qu'il n'y ait plus de case à gauche ou au-dessous de •. Le tableau ainsi obtenu est $T' = jdt_{\bullet}(T)$.

Partant d'une case inférieure gauche, la procédure est analogue mais au lieu de considérer la case qui se trouve au-dessous de • et celle à sa gauche et de choisir celle qui contient le plus petit entier, on regarde la case qui se trouve au-dessus de • et celle à sa droite et on choisit celle qui contient le plus grand entier.

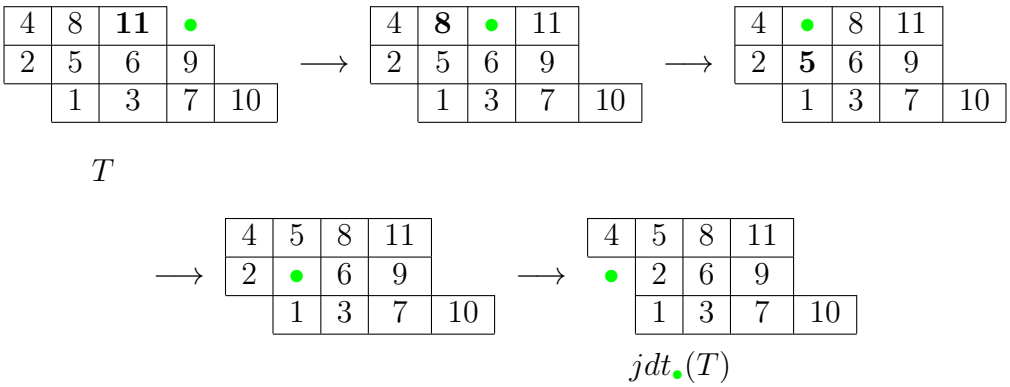


FIGURE 3.2 – Glissement dans un Jeu de Taquin, partant d'une case supérieure droite

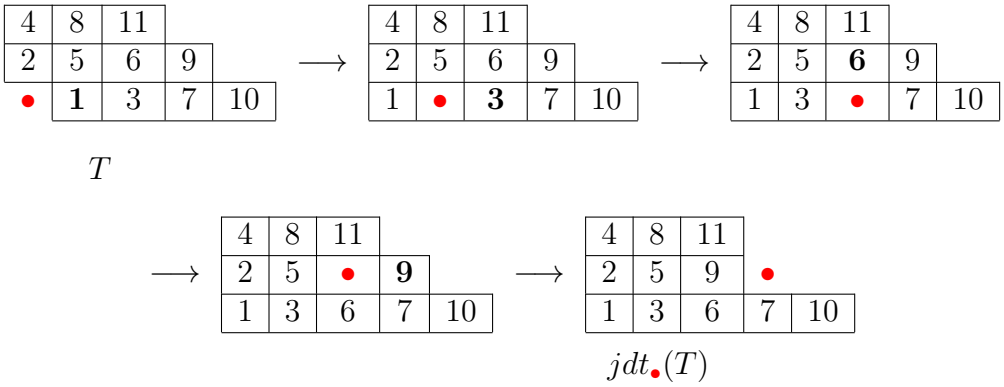


FIGURE 3.3 – Glissement dans un Jeu de Taquin, partant d'une case inférieure gauche

Si on note $\bullet_i$ la position initiale d'un glissement dans un tableau T et $\bullet_f$ la position finale, on peut alors noter que $jdt_{\bullet_f}(jdt_{\bullet_i}(T)) = T$.

Autrement dit, si partant d'une position initiale on effectue un glissement de jeu de Taquin, puis on effectue un second glissement sur le tableau obtenu en partant de la position finale dans laquelle on est arrivé après le premier glissement, on retrouve le tableau de départ.

La transformation que l'on vient de décrire nous permet de définir une équivalence sur les tableaux gauches.

Définition 3.1.1 *Deux tableaux T et T' sont dit "jeu de taquin equivalents" si et seulement si l'un peut être obtenu à partir de l'autre par une séquence de glissements de Taquin et on note alors $T \overset{jdt}{\sim} T'$.*

Notons que la relation $\overset{jdt}{\sim}$ est symétrique (puisque $jdt_{\bullet_f}(jdt_{\bullet_i}(T)) = T$) et transitive (évident d'après la définition).

3.2 Les coefficients de Littlewood-Richardson

L'équivalence sur les tableaux gauches définie dans la section précédente permet de donner une définition des coefficients de Littlewood-Richardson.

Théorème 3.2.1 *Soit T un tableau de Young standard de forme ν, alors le coefficient de Littlewood-Richardson $c^{\lambda}_{\mu\nu}$ est égal au nombre de tableaux gauches de forme λ/μ jeu de taquin équivalent à T.*

Nous n'aborderons pas ici la démonstration de ce théorème que l'on peut cependant retrouver dans [7].

Exemple : Soient $\nu = 4, 3, 1$, $\lambda = 4, 4, 2, 1$ et $\mu = 2, 1$

8			
5	6	7	
1	2	3	4

T de forme ν

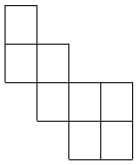


Diagramme gauche de λ/μ

Il existe exactement deux tableaux gauches de forme λ/μ équivalents à T :

5			
1	8		
	2	6	7
		3	4

8			
1	5		
	2	6	7
		3	4

Donc $c^{4421}_{21,431} = 2$.

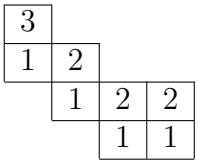
On peut également définir les coefficients de Littlewood-Richardson à l'aide des mots de Yamanouchi.

Soit w, le mot lu sur les lignes de λ/μ de droite à gauche et de bas en haut. Ce mot est appelé *reverse reading word* en anglais.

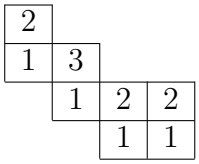
Le coefficient de Littlewood-Richardson $c^{\lambda}_{\mu\nu}$ est alors égal au nombre de tableaux semi-standards de forme λ/μ et de remplissage ν dont le mot associé w est un mot de Yamanouchi.

Et w est un mot de Yamanouchi si dans tout suffixe de w, le nombre de i est supérieur ou égal au nombre de $i + 1$.

Pour l'exemple précédent, on peut avoir



ou



où les mots sont respectivement 11221213 et 11221312 sont les deux seules possibilités de mots de Yamanouchi. On retrouve donc encore une fois $c_{21,431}^{4421} = 2$.

Ces coefficients ont étaient introduits par D.E. Littlewood et A.R. Richardson [5] et sont importants ici car ils interviennent dans la multiplication des fonctions Schur.

Théorème 3.2.2 (Littlewood-Richardson (1934))
Soient μ et ν deux partitions alors

$$s_\mu s_\nu = \sum_{\lambda \vdash |\mu|+|\nu|} c_{\mu\nu}^{\lambda} s_\lambda$$

En réalité, dans [5], Littlewood et Richardson donnent un théorème permettant de construire la multiplicaton de deux fonctions Schur de degrés différents. Il a fallu plus de 40 ans pour parvenir à une démonstration rigoureuse de ce théorème qui fut source de nombreuses erreurs.

Théorème 3.2.3 (Règle de Littlewood-Richardson (1934))
Soient deux fonctions Schur indicées par les partitions μ et ν.
On construit tout d'abord les tableaux standards correspondant à μ et à ν et contenant des symboles différents pour chaque tableau. On peut par exemple choisir $a, b, c, \cdots$ pour μ et $\alpha, \beta, \gamma, \cdots$ pour ν.

On prend ensuite le tableau de μ tel quel et on y ajoute les éléments de la première ligne du tableau de ν en respectant les règles suivantes :

1. *Les éléments de ν peuvent être ajoutés à une seule ligne de μ ou divisés entre plusieurs lignes mais l'ordre ne doit jamais être modifié.*
2. *Si on choisit de diviser les éléments en plusieurs ensembles, le premier sera ajouté à l'une des lignes de μ, le second à une des lignes au-dessus, etc...*
3. *Après l'ajout des symboles, aucune ligne ne doit contenir plus d'éléments que la précédente.*
4. *Deux éléments ajoutés ne peuvent pas se trouver dans la même colonne.*

Une fois la première ligne de ν correctement ajoutée à μ, on y ajoute la seconde ligne en respectant les règles précédentes et avec une nouvelle restriction. Chaque élément de la seconde ligne de ν doit se trouver sur une ligne située au-dessus de celle dans laquelle se trouve le symbole de la première ligne situé dans la même colonne de ν.
Prenons l'exemple où a est le premier élément de la première ligne et b le premier de la deuxième ligne. Alors dans le nouveau tableau construit, si a est placé si la ligne i, b pourra être placé sur la ligne $i+1$ ou une suivante.

On ajoute ensuite la troisième ligne de ν en respectant toutes les conditions précédentes et ainsi de suite jusqu'à avoir ajouté tous les symboles de ν à μ.

Alors, le coefficient de chaque s_λ apparaissant dans le produit $s_\mu \times s_\nu$ est donné par le nombre de tableaux de forme λ obtenus.

Exemple : On peut facilement obtenir sur Maple que

$$s_{321} \times s_{21} = s_{531} + s_{522} + s_{5211} + s_{441} + 2s_{432} + 2s_{4311} + 2s_{4221} + s_{42111} + s_{333} + 2s_{3321} + s_{33111} + s_{3222} + s_{32211}$$

On va maintenant retrouver ce résultat par la règle de Littlewood-Richardson.
$\mu = 321$ et $\nu = 21$, on a les tableaux suivants :

$$\begin{array}{ccc} & & \\ & & \\ \mathrm{d} & \mathrm{e} & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} \end{array} \qquad\qquad \begin{array}{ccc} & & \gamma \\ & & \\ \alpha & \beta & \end{array}$$

Tous les tableaux respectant les règles de construction de l'on peut obtenir sont les suivants, avec les lettres de μ en gris :

$$\begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & \gamma & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & \beta \end{array} \quad s_{531} \qquad \begin{array}{ccccc} & & & \gamma & \\ & & & & \\ \mathrm{d} & \mathrm{e} & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & \beta \end{array} \quad s_{522} \qquad \begin{array}{ccccc} & & & & \gamma \\ & & & & \\ \mathrm{d} & \mathrm{e} & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & \beta \end{array} \quad s_{5211}$$

$$\begin{array}{cccccc} & & & & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & \beta & \gamma & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{441} \qquad \begin{array}{cccccc} & & & \gamma & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & \beta & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{432} \qquad \begin{array}{cccccc} & & & & \gamma & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & \beta & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{4311} \qquad \begin{array}{cccccc} & & & \beta & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & \gamma & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{432}$$

$$\begin{array}{cccccc} & & & & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{4221} \qquad \begin{array}{cccccc} & & & & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & \gamma & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{4311} \qquad \begin{array}{cccccc} & & & & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{4221} \qquad \begin{array}{cccccc} & & & & & \\ & & & & & \\ \mathrm{d} & \mathrm{e} & & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & \alpha & & \end{array} \quad s_{42111}$$

$$\begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & \alpha & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & & \end{array} \quad s_{333} \qquad \begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & \alpha & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & & \end{array} \quad s_{3321} \qquad \begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & \alpha & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & & \end{array} \quad s_{3321} \qquad \begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & \alpha & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & & \end{array} \quad s_{33111}$$

$$\begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & & \end{array} \quad s_{3222} \qquad \begin{array}{ccccc} & & & & \\ & & & & \\ \mathrm{d} & \mathrm{e} & & & \\ \mathrm{a} & \mathrm{b} & \mathrm{c} & & \end{array} \quad s_{32211}$$

En additionnant toutes les formes obtenue, on retrouve bien le même résultat pour $s_{321} \times s_{21}$.

Revenons maintenant aux coefficients de Littlewood-Richardson, on a par exemple $c^{531}_{321,21} = 1$ ou encore $c^{432}_{321,21} = 2$ et $c^{54}_{321,21} = c^{321111}_{321,21} = 0$ puisque les tableaux des partitions 54 et 321111 n'apparaissent pas dans $s_{321} \times s_{21}$.

□

Il existe également des cas particuliers de multiplication des fonctions Schur dont la formule de Pieri.

Théorème 3.2.4 (Formule de Pieri)
Soient μ une partition et $n \in \mathbb{N}^*$ alors

$$s_n s_\mu = \sum_\lambda s_\lambda$$

où λ est une partition obtenue à partir de μ en ajoutant n éléments à son diagramme de Young, avec au plus un élément par colonne.

Remarque : $s_n s_\mu = h_n s_\mu$

Lemme 3.2.1 (Cas particulier de la formule de Pieri)
Soit μ une partition telle que $l(\mu) = n - 1$ alors

$$s_1 s_\mu = \sum_{\substack{\lambda \vdash n \\ \mu \subseteq \lambda}} s_\lambda$$

La règle de Littlewood-Richardson est par la suite étendue pour exprimer les fonctions Schur indicées par des formes gauches, appelées fonctions Schur gauches. Notons que ces fonctions ne forment pas une base de fonctions symétriques contrairement aux fonctions Schur vues précédemment.

Théorème 3.2.5 (Zelevinsky (1981))
Soient λ/μ une forme gauche alors

$$s_{\lambda/\mu} = \sum_{\nu \vdash |\lambda/\mu|} c^\lambda_{\mu\nu} s_\nu$$

On a programmé, entre autre, une fonction en Maple pour calculer les coefficients de Littlewood-Richardson que l'on peut retrouver dans l'annexe A. Cette fonction s'appuie sur la fonction `skew` du package SF (voir chapitre suivant) et sur la formule de Zelevinsky que l'on vient d'énoncer.

Chapitre 4

Fonctions symétriques en Maple

4.1 Package 'SF'

Développé par John Stembridge [8, 9, 10], le package SF regroupe un ensemble de 25 procédures pour Maple fournissant un environnement permettant de manipuler et d'effectuer un certain nombre de calculs sur les fonctions symétriques. La dernière version de ce package date de 2005 et peut-être téléchargée sur le site web de Stembridge [9] avant d'être chargée et utilisée dans Maple.

Le principe de ce package est de permettre un certain nombre de manipulations sur les fonctions symétriques en s'appuyant notamment sur les bases des fonctions symétriques définies au chapitre précédent.

Les bases par défaut programmées dans le package SF sont e, p, h, s, les élémentaires, les sommes de puissances, les homogènes complètes et les Schur. Il est cependant possible d'ajouter d'autres bases lors de l'utilisation de SF grâce aux fonctions `add_basis` et `dual_basis`. Notamment pour ajouter la base des monomiales, on utilisera la commande `dual_basis(m,h)` signifiant ainsi que l'on souhaite créer une base libellée par la lettre 'm' et qui soit la base duale de la base h.

Concernant les fonctions de ce package qui nous intéressent dans notre cas, on trouve plusieurs procédures qui permettent d'exprimer une fonction symétrique dans une base donnée. Il s'agit des fonctions `toe`, `top`, `toh` et `tos`. La fonction `tom` sera également crée lors de la création de la base m.

La fonction `evalsf(f,a)` qui calcule l'évaluation plethystique de la fonction symétrique f sur l'expression a sera également utilisée. La fonction `skew(`$s[\nu]$`,`$s[\mu]$`)` permet d'obtenir la valeur de la fonction Schur $s_{\mu/\nu}$ indicée par le diagramme gauche μ/ν.

Enfin, seront également très utiles des fonctions réalisant des opérations sur les partitions comme `Par` qui donne la liste de toutes les partitions d'un entier, `conjugate` qui prend en entrée une partition et calcule son conjugué ou encore `subPar` qui renvoie la liste de toutes les partitions incluses dans une partition donnée.

Enfin, on sera également amené à utiliser les fonctions effectuant des calculs sur les matrices de Jacobi-Trudi, c'est-a-dire `jt_matrix` qui donne la matrice de Jacobi-Trudi d'une partition donnée et `jt_det` qui donne le determinant de cette même matrice.

4.2 Implémentation de nouvelles fonctions

En se basant sur certaines fonctions déjà présentes dans le package SF, d'autres ont été développées dans le but de compléter le package initial et d'avoir de nouvelles fonctionnalités sur les fonctions symétriques.

4.2.1 Conventions d'écriture

Les fonctions descrites dans cette section sont des fonctions qui servent uniquement a réécrire de façon plus naturelle les fonctions symétriques utilisées dans SF.

En effet, dans le package SF, les bases dites multiplicatives, ie h, p, et e, sont considérées comme étant indicées uniquement par des entiers et non par des partitions. Cela est possible puisque $h_i * h_j = h_{ij}$ pour $i \geq j$ (de même pour e et p).

De plus, les entiers ne sont pas écrits en indices contrairement à ce dont on a l'habitude. Ainsi, h_1 est écrit $h1$ dans SF, par exemple.

On souhaite donc avoir une écriture des éléments appartenant à ces trois bases avec des partitions en indices.

Les fonctions non multiplicatives, c'est-à-dire les Schur et par la suite toute base ajoutée lors de l'utilisation et notamment la base des fonctions monomiales m, sont indicées par des partitions dont les différentes parts sont séparées par les virgules. On va donc obtenir la même écriture pour p, h et e.

Les fonctions décrites ci-dessous permettent de réaliser les différentes transformations voulues et également la transformation inverse permettant de retrouver l'écriture définie dans SF.

En effet, il est à noter qu'une fois réécrites comme souhaitées, les fonctions symétriques ne sont plus manipulables dans SF, on ajoute donc également une procédure permettant de revenir aux conventions d'écritures utilisées dans le package SF.

1. **De SF vers une écriture indicée** : Cette transformation est découpée en plusieurs étapes qui sont les suivantes :
   — La procédure `SFtoInd` permet de découper la fonction symétrique homogéne à transformer et d'identifier les termes à transformer. Quand un terme à transformer est identifié, on utilise alors la procédure `elmtInd` pour le transformer.
   — La fonction `elmtInd` permet donc d'obtenir un élément indicé. C'est-à-dire que, pour tout entier n, hn va devenir h_n, et de même pour p et e.
   — Il reste maintenant une dernière étape pour regrouper les termes appartenant à p, h ou e faisant partie d'un même produit et ainsi créer un élément indicé par une partition. Ce travail est effectué dans la procédure `SFtoPart`, qui fait d'abord appel à `SFtoInd`, puis à partir du résultat obtenu crée les élément indicés par des partitions.

   Exemples :
   SFtoPart$(h10 + e1 * e1 * e1) \rightarrow e_{1,1,1} + h_{10}$
   SFtoPart$(s[4,2,2] * p3 * p1) \rightarrow p_{3,1}s_{4,2,2}$

$\text{SFtoPart}(2 + 4 * p2 * e3 * e1 * e3 + h1 * h5 * h1) \to 2 + 4p_2e_{3,3,1} + h_{5,1,1}$

2. **D'une écriture indicée vers SF** : La fonction `partToSF` réalise la transformation inverse à celle décrite précédemment, à savoir transformer les éléments de p, h et e indicés par des partitions en produit d'éléments indicés par des entiers et dont les indices font partie du nom.

   Exemples :
   $\text{partToSF}(p[4,3,2,2,1]) \to p1p2^2p3p4$
   $\text{partToSF}(h[11] + h[1,1]) \to h1^2 + h11$
   $\text{partToSF}(s[4,2,1] + e[2] * e[1] + e[2,1]) \to 2e1e2 + s_{4,2,1}$

4.2.2 Évaluation plethystique sur un produit d'alphabets

On souhaite maintenant calculer l'évaluation plethystique d'une fonction symétrique homogène f sur un produit de deux alphabets virtuels X et Y et obtenir le résultat dans deux bases choisies $b^{(1)}$ et $b^{(2)}$. Plus précisément, on veut calculer $f[X \cdot Y]$ où $X = x_1 + x_2 + \cdots + x_n$ et $Y = y_1 + y_2 + \cdots + y_m$ et on veut que le résultat soit écrit dans les bases $b^{(1)}$ et $b^{(2)}$, c'est-à-dire :

$$f[X \cdot Y] = \sum_{\mu \vdash d} \left(\sum_{\nu \vdash d} C_{\mu\nu} b^{(2)}_\nu[Y] \right) b^{(1)}_\mu[X]$$

Les $C_{\mu\nu}$ sont donc à déterminer.

Pour obtenir cela, on va dans un premier temps faire abstraction des alphabets. La première étape sera donc d'exprimer f dans la base des sommes de puissances p_λ à l'aide de la fonction `top`. On exprime f de cette façon, car on sait d'après la proposition 2.4.1 b. que $p_k[A \times B] := p_k[A] \times p_k[B]$. Ainsi, lorsque l'on va calculer $f[X \cdot Y]$, tous les p_k vont simplement devenir $p_k[X] \times p_k[Y]$. On ne s'occupe pas ici de développer le résultat sur les alphabets puisque ce n'est pas le but recherché.

On obtient alors $f = \sum_{\mu \vdash d} \alpha_\mu p_\mu$ où d est le degré de f.

Et $f[X \cdot Y] = \sum_{\mu \vdash d} \alpha_\mu p_\mu[X \cdot Y] = \sum_{\mu \vdash d} \alpha_\mu p_\mu[X].p_\mu[Y]$

L'étape suivante consiste à traiter $p_k[X]$ et $p_k[Y]$ comme des bases différentes.

On va alors se concentrer sur les $p_k[X]$ et on effectue un changement de base pour écrire les $p_k[X]$ dans la base $b^{(1)}$ en considérant les $p_k[Y]$ comme des constantes. Pour cela, on utilisera la fonction `tob1` de SF. On obtient alors f de la forme suivante :

$f = \sum_{\mu \vdash d} \beta_\mu b^{(1)}_\mu$

où chaque β_μ est en réalité une combinaison linéaire de $p_k[Y]$.

Enfin, on réitère l'étape précédente sur les β_μ mais cette fois pour les exprimer dans la base $b^{(2)}$. On utilise la fonction `tob2` sur f mais cette fois en précisant en second argument p, signifiant ainsi que l'on ne veut changer que les p_k dans f. On obtient :

$$f = \sum_{\mu \vdash d} \left(\sum_{\nu \vdash d} C_{\mu\nu} b^{(2)}_\nu \right) b^{(1)}_\mu$$

Les alphabets n'ont pas une grande importance lorsque l'on effectue le calcul puisqu'on effectue jamais le développement sur les variables, mais on sait que l'on évalue les $b^{(1)}$ sur X et $b^{(2)}$ sur Y. Les alphabets sont donc implicites tout au long du calcul et ajoutés a la fin lors de l'affichage du résultat pour plus de clarté comme on le verra dans la section suivante.

Voici la fonction implementée en Maple.

```
> prodsf2alph:=proc(f,b1,b2) local d,s,res;
>    res:=top(f);
>    d:=stdeg(f);
>    s:=seq(cat('p',k)=cat('p',k)*cat('q',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b1),res,p);
>    s:=seq(cat('q',k)=cat('p',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b2),res,p);
> end:
```

Exemples :

```
> for i from 2 by 1 to 4 do
> cat('h',i):=prodsf2alph(cat('h',i),h,m);
> od;
```

$$h2 := h1^2 m_{1,1} + h2\, m_2$$

$$h3 := h1^3 m_{1,1,1} + h2\, h1\, m_{2,1} + h3\, m_3$$

$$h4 := h1^4 m_{1,1,1,1} + h2\, h1^2 m_{2,1,1} + h1\, h3\, m_{3,1} + h2^2 m_{2,2} + h4\, m_4$$

```
> prodsf2alph(s[2,1],h,m);
> prodsf2alph(s[2,1],s,s);
```

$$h1^3 m_{2,1} + 2\, h1^3 m_{1,1,1} + h2\, h1\, m_3 - h3\, m_3$$

$$2\, s_3 s_{2,1} + {s_{2,1}}^2 + 2\, s_{2,1} s_{1,1,1}$$

Cette fonction nous permet d'exprimer n'importe quelle fonction symétrique évaluée sur un produit d'alphabets dans deux bases (différentes ou non). En particulier, on peut retrouver la formule de Cauchy (2.4.1), en appliquant cette fonction à h_n pour les bases h et m (h2, h3 et h4, ci-dessus).

On peut vérifier que la fonction est correcte en retrouvant également la formule analogue pour e_n (2.4.2).

Enfin, on peut également expérimenter la seconde formule de Cauchy exprimée dans la base des Schur ainsi que sa forme duale (2.2 et 2.4).

Des exemples sont présentés en annexe A.

Remarque : Si l'on veut avoir la base p des sommes de puissance comme l'une des bases du résultat, il faut la spécifier en deuxième ($b^{(2)} = p$), sinon le résultat donné par la fonction sera incorrect puisque les p_k sont utilisés et modifiés plusieurs fois au cours de la procédure. On écrira donc `prodsf2alph(f,b1,p)` plutôt que `prodsf2alph(f,p,b1)` où $b1$ est n'importe qu'elle base, y compris p elle-même lorsque l'on souhaite exprimer le résultat dans la base des p pour les deux alphabets.

Aprés avoir implémenté le cas d'un produit de deux alphabets, on s'intéresse maintenant à une évaluation plethystique sur trois alphabets, c'est-à-dire $f[X \cdot Y \cdot Z]$.

Encore une fois, le cas est similaire à $f[X \cdot Y]$ mais requiert un alphabet et une étape supplémentaires. En effet, on se base toujours sur la même règle de calcul (2.4.1 b.) que l'on exprime ici sur trois alphabets : $p_k[A \times B \times C] := p_k[A] \times p_k[B] \times p_k[C]$.

Exemples :

```
> prodsf3alph(s[1],s,s,s);
```

$$s_1(X)\, s_1(Y)\, s_1(Z)$$

```
> prodsf3alph(s[2],s,s,s);
```

$$s_2(X)s_2(Y)s_2(Z) + s_2(X)s_{1,1}(Y)s_{1,1}(Z) + s_{1,1}(X)s_2(Y)s_{1,1}(Z) + s_{1,1}(X)s_{1,1}(Y)s_2(Z)$$

```
> prodsf3alph(s[1,1],s,s,s);
```

$$s_2(X)s_2(Y)s_{1,1}(Z) + s_2(X)s_{1,1}(Y)s_2(Z) + s_{1,1}(X)s_2(Y)s_2(Z) + s_{1,1}(X)s_{1,1}(Y)s_{1,1}(Z)$$

On peut retrouver d'autres exemples en annexe A.

Si l'implémentation de cette fonction n'a pas posé de problèmes majeurs et nous permet donc d'obtenir facilement une évaluation d'une fonction symétrique sur trois alphabets, nous verrons cependant par la suite qu'établir des formules théorique avec trois alphabets est en revanche bien plus complexe.

4.2.3 Évaluation plethystique sur une somme d'alphabets

De la même façon que pour le cas d'un produit de deux alphabets $f[X \cdot Y]$, on souhaite maintenant traiter le cas d'une somme de deux alphabets $f[X + Y]$.

En fait, il suffit de remarquer que $p_k[A + B] := p_k[A] + p_k[B]$ et on peut alors procéder de façon analogue au cas $f[X \cdot Y]$ pour écrire la procédure réalisant le calcul de $f[X + Y]$. On va ainsi obtenir successivement f de degré d dans les formes suivantes :

$$f[X+Y] = \sum_{\mu \vdash d} \alpha_\mu p_\mu[X+Y] = \sum_{\mu \vdash d} \alpha_\mu \sum_{\mu_i, \mu_j \in \mu} p_{\mu_i}[X] p_{\mu_j}[Y]$$

Puis on va transformer les $p_k[X]$ dans la base des $b^{(1)}$ et les $p_k[Y]$ dans la base des $b^{(2)}$

Pour le cas précédent, on ne s'est pas attardé à parler des alphabets, on va donc ici décrire de quelle façon on obtient un résultat avec alphabet.

L'ajout du premier alphabet se fait une fois le traitement sur la première base effectué, le deuxième alphabet une fois le traitement sur la seconde base effectué. L'idée est de repérer dans chaque cas la base sur laquelle on est en train de travailler et d'y appliquer le symbole choisi pour la base en question, s'il est précisé en argument ou le cas échéant le symbole X pour $b^{(1)}$ et le symbole Y pour $b^{(2)}$.

Enfin, une dernière étape survient dans le cas où $b^{(1)} = b^{(2)}$ pour s'assurer que chaque base est exprimée sur un seul alphabet.

Ci-dessous, la procédure pour le calcul de $f[X+Y]$, les fonctions `alph` et `correctAlph` utilisées sont respectivement les fonctions permettant d'ajouter les alphabets et d'effectuer la correction dans le cas $b^{(1)} = b^{(2)}$. Ces deux fonctions sont présentes en annexe A.

```
> addsf2alph:=proc(f,b1,b2) local d,s,res;
>    res:=top(f);  d:=stdeg(f);
>    s:=seq(cat('p',k)=(cat('p',k)+cat('q',k)),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b1),res,p);
>    res:=alph(res,b1,d,X);
>    s:=seq(cat('q',k)=cat('p',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b2),res,p);
>    res:=alph(res,b2,d,Y);
>    if b1=b2 then
>      correctAlph(res,b1,d);
>    else res;
>    fi;
> end:
```

Pour effectuer les vérifications de la procédure, on va utiliser des formules déjà établies dont la suivante.

$$s_\mu[X+Y] = \sum_{\nu\subseteq\mu} s_\nu(X) s_{\mu/\nu}(Y) \tag{4.1}$$

Idée de la preuve :
Pour montrer l'équation précédente, il faut se reporter à la définition combinatoire des fonctions Schur utilisant des tableaux semi-standard. On applique ensuite la transformation $i \mapsto z_i$ pour avoir un remplissage non plus avec des entiers mais avec les variables z_i.
Les règles de remplissage sont toujours respectées, à savoir que si $i \leq j$ alors $z_i \leq z_j$ si z_i et z_j sont sur la même ligne et si $i < j$ alors $z_i < z_j$ si z_i et z_j sont sur la même colonne. On a donc un ordre sur les variables de Z.
On peut ensuite définir $Z = z_1 + z_2 + \cdots$ comme étant la somme de deux alphabets $Z = X+Y$. On a alors un ordre sur X et Y, c'est-à-dire que $x_1 < x_2 < \cdots < y_1 < y_2 < \cdots$.
On peut alors représenter $s_\mu[Z] = s_\mu[X+Y]$ par la somme des remplissages de tableaux de Young standards de forme μ dans lesquels on remplirait la partie de forme ν par des x_i et la partie μ/ν par des y_i.

□

Puisque $s_n = h_n$ et $s_{1^k} = e_k$, il en découle également les formules suivantes :

$$h_n[X+Y] = \sum_{k=0}^{n} h_k(X) h_{n-k}(Y) \tag{4.2}$$

$$e_n[X+Y] = \sum_{k=0}^{n} e_k(X) e_{n-k}(Y) \tag{4.3}$$

Ces trois formules vont nous permettre de vérifier l'exactitude de la procédure implementée ainsi que de voir si la forme du résultat nous permet d'identifier facilement les formules.

```
> for i in Par(5) do
> cat('s',op(i)):=addsf2alph(s[op(i)],s,s);
> od;
```

$$s5 := s_5(Y) + s_5(X)\,s + 2\,s_3s_2 + 2\,s_4s_1$$

$$s41 := 2\,s_1s_{3,1} + s\left(s_{4,1}\right)(Y) + \left(s_{4,1}\right)(X)\,s + 2\,s_2s_{2,1} + 2\,s_3s_{1,1} + 2\,s_3s_2 + 2\,s_4s_1$$

$$s32 := 2\,s_1s_{3,1} + \left(s_{3,2}\right)(X)\,s + 2\,s_1s_{2,2} + s\left(s_{3,2}\right)(Y) + 2\,s_{1,1}s_{2,1} + 2\,s_2s_{2,1} + 2\,s_3s_2$$

...

Voir l'ensemble des résultats en annexe A.

```
> for i from 2 by 1 to 3 do
> cat('h',i):=addsf2alph(cat('h',i),h,h);
> cat('e',i):=addsf2alph(cat('e',i),e,e);
> od;
```

$$hh2 := h1(Y)\,h1(X) + h2(X) + h2(Y)$$

$$ee2 := e1(Y)\,e1(X) + e2(X) + e2(Y)$$

$$hh3 := h2(Y)\,h1(X) + h1(Y)\,h2(X) + h3(X) + h3(Y)$$

$$ee3 := e2(Y)\,e1(X) + e1(Y)\,e2(X) + e3(X) + e3(Y)$$

Dans le premier cas, on a utilisé la formule (4.1) et dans le deuxième cas, on s'est servi de (4.2) et (4.3) et on obtient bien les résultats attendus.

Comme pour le produit d'alphabets, on implémente également une évaluation sur une somme de trois alphabets $f[X + Y + Z]$ et l'on nomme la fonction produite `addsf3alph`. (Voir annexe A.)

4.2.4 Évaluation de $f[XY + XZ + YZ]$

La seconde évaluation sur trois alphabets à laquelle on s'intéresse combine maintenant sommes et produits de trois alphabets.

On va en fait pouvoir réutiliser pour ce cas la procédure développée pour le cas $f[X \cdot Y]$ que l'on va appliquer sur chacun des membres de la somme.

Comme dans chaque cas, on va commencer par convertir la fonction symétrique dans la base des sommes de puissance (p_k). Puis on va considéré les $p_i[XY]$, $p_j[XZ]$ et $p_k[YZ]$ comme des bases différentes.

Considérons f comme étant la fonction de départ et d son degré, on va alors avoir :

$$f[XY + XZ + YZ] = \sum_{i,j,k} \alpha_{ijk}(p_i[XY]p_j[XZ]p_k[YZ])$$

On va alors pourvoir appliquer une première fois la fonction calculant $f[X \cdot Y]$ avec les deux premières bases. Le traitment va alors être effectué uniquement sur les p_i et laissera les p_j et p_k inchangés.

On va ensuite répéter le même processus sur les p_j et enfin on fera une troisième et dernière fois la même manipulation sur les p_k.

Il faudra encore une fois prêter une attention particulière aux alphabets, surtout dans le cas où plusieurs bases sont les mêmes.

La fonction Maple que l'on vient de décrire se trouve en Annexe A.

4.2.5 Fonctions Schur gauches et changements de base

On souhaite pouvoir travailler avec les fonctions Schur gauches. Il existe dans SF la fonction `skew` qui permet de calculer des Schur gauches dans la base des p, mais le package SF ne permet pas plus de manipulation sur les Schur gauches.

On décide de d'écrire les Schur gauches $s[[\mu],[\nu]]$ pour $s_{\mu/\nu}$ et d'avoir ainsi des Schur indicés par deux partitions.

Ayant définit cette écriture, on peut maintenant exprimer $s_\mu[X+Y]$ avec des fonctions Schur et des fonctions Schur gauches à l'aide de (4.1). Une telle fonction est utile pour effectuer le développement de $s_\mu[X+Y]$, mais elle ne permet toujours pas de manipuler des Schur gauches.

Pour manipuler les Schur gauches, on veut pouvoir les transformer dans différentes bases et ainsi les prendre en compte dans les calculs. Pour cela, on va définir la fonction `Tos` qui nous permet d'effectuer les mêmes manipulations que la fonction `tos` du package SF mais qui permet aussi de transformer des Schur gauches dans la base des Schur à l'aide de (3.2.5). Cette nouvelle procédure prend en paramètre n'importe quelle fonction symétrique homogène quelque soit l'écriture utilisée (celle de SF ou celle définie précédemment) qui peut contenir des Schur gauches écrit de la façon définie ci-dessus. Et le résultat obtenu est donc la fonction écrite dans la base des Schur.

A l'aide de la fonction `Tos` ainsi que des fonctions de SF `top`, `toh` et `toe`, on implémente également les fonctions `Top`, `Toh` et `Toe` qui permettent la même chose que `Tos` mais qui donnent le résultat dans la base des p, h et e respectivement, avec les termes indicés par des partitions.

Exemples :

```
> Tos(s[[2,1],[1]]);
> Tos(s[[4,3,2],[1,1]]);
```

$$s_2 + s_{1,1}$$

$$s_{4,2,1} + s_{3,3,1} + s_{3,2,2}$$

Utilisons par exemple la fonction $f := p_3 s_{2,1/1} + h_{2,1} + s_3$:

```
> Tos(p3*s[[2,1],[1]]+h[2,1]+s[3]);
> tos(%);
```

$$(s_3 - s_{2,1} + s_{1,1,1})(s_2 + s_{1,1}) + 2s_3 + s_{2,1}$$

$$s_5 + s_{4,1} - s_{3,2} + 2s_3 - s_{2,2,1} + s_{2,1,1,1} + s_{2,1} + s_{1,1,1,1,1}$$

```
> Top(p3*s[[2,1],[1]]+h[2,1]+s[3]);
> Toh(p3*s[[2,1],[1]]+h[2,1]+s[3]);
> Toe(p3*s[[2,1],[1]]+h[2,1]+s[3]);
```

$$p_{3,1,1} + 2p_{1,1,1}(1/3) + p_{2,1} + (1/3)p_3$$
$$h_{1,1,1,1,1} - 3h_{2,1,1,1} + 3h_{3,1,1} + h_{2,1} + h_3$$
$$e_{1,1,1,1,1} - 3e_{2,1,1,1} + 2e_{1,1,1} + 3e_{3,1,1} - 3e_{2,1} + e_3$$

Notons que l'implémentation de ces nouvelles fonctionnalités permet également de faire des calculs plethystiques sur des fonctions Schur gauches. En effet, il suffit de les exprimer dans la base des Schur puis de calculer l'évaluation plethystique la nouvelle forme obtenue.

On obtient par exemple que

$$s_{32/11}[X+Y] = h_1(Y)h_1(X)^2 + h_1(Y)^2h_1(X) + h_1(X)h_2(X) - h_3(X) + h_1(Y)h_2(Y) - h_3(Y)$$

dans la base des h ou encore

$$s_{32/11}[X+Y] = s_{21}(X) + s_2(X)s_1(Y) + s_{11}(X)s_1(Y) + s_1(X)s_2(Y) + s_1(X)s_{11}(Y) + s_{21}(Y)$$

dans la base des Schur.

4.2.6 Création de nouvelles bases

On dispose maintenant des fonctions `Tob` où $b \in \{s, p, h, e\}$, cependant, si on crée de nouvelles bases, par exemple la base des monomiales m, on aimerait également disposer d'une fonction `Tom` en plus de la fonction `tom` qui va être créée au moment de l'ajout de m.

On implémente donc deux nouvelles fonctions `Dual_basis` et `Add_basis` qui dans un premier temps appliquent respectivement `dual_basis` et `add_basis` et dans un second temps, créent la fonction `Tob` correspondant à la nouvelle base.

Cette fonction `Tob` s'appuie sur le même principe que `Top`, `Toh` et `Toe`. Elle utilise tout d'abord `Tos` qui permet de gérer les Schur gauches, puis, elle applique la fonction nouvellement créé `tob` afin de donner le résultat dans la base voulue.

Exemple :

 # On commence par créer la base m

```
> Dual_basis(m,h);
```

Okay

```
> tom(s[2,1]*h2);
```

$$m_{4,1} + 2m_{3,2} + 4m_{3,1,1} + 6m_{2,2,1} + 11m_{2,1,1,1} + 20m_{1,1,1,1,1}$$

> `Tom(s[2,1]*h2);`

$$m_{4,1} + 2m_{3,2} + 4m_{3,1,1} + 6m_{2,2,1} + 11m_{2,1,1,1} + 20m_{1,1,1,1,1}$$

> `Tom(s[[4,3],[2,1]]);`

$$m_{3,1} + 2m_{2,2} + 3m_{2,1,1} + 5m_{1,1,1,1}$$

4.2.7 Evaluation plethystique générale

Finalement, on souhaite regrouper la plupart des outils développés précédemment dans une seule et unique fonction qui permettrait donc de calculer une fonction qui serait la somme et/ou le produit d'évaluations plethystiques sur des sommes ou produits de deux alphabets.

De plus, les fonctions que l'on souhaite évaluer pourrait être exprimées dans n'importe quelle base ou avec des Schur gauche.

On voudrait pouvoir calculer directement, par exemple, $f[X \cdot Y] - g[X + Y]$ où f serait un produit de Schur gauche et g une somme de fonctions élémentaires e et monomiale m, sans avoir à appeler une par une chacun des fonctions nécessaires.

On crée donc une fonction prenant en paramètre une fonction comme écrite au paragraphe précédent, une base pour l'alphabet X et une pour Y et effectuant le calcul voulu.

Exemple : $s_{2,2/1}[X \cdot Y] - (e_2 + m_{1,1})[X + Y]$

> `evalpl((s[[2,2],[1]])[X*Y]-(e2+m[1,1])[X+Y],s,s);`

$s_3(X)s_{2,1}(Y) + s_{2,1}(X)s_3(Y) + s_{2,1}(X)s_{2,1}(Y) + s_{2,1}(X)s_{1,1,1}(Y) + s_{1,1,1}(X)s_{2,1}(Y) - 2s_{1,1}(Y) - 2s_1(X)s_1(Y) - 2s_{1,1}(X)$

Notons que si l'on souhaite calculer une évaluation plethystique simple, c'est-à-dire uniquement sur un produit ou uniquement sur une somme d'alphabets, il est plus judicieux d'utiliser la fonction appropriée parmis celles décrites précédemment, car le calcul sera plus efficace.

Toutes les fonctions décrites dans ce chapitres ont été regroupées dans une feuille de travail Maple contenant également des exemples d'utilisation et que l'on peut retrouver en annexe A de ce rapport.

Chapitre 5

Evaluation plethystique de fonctions symétriques

5.1 Evaluation de $s_\mu[X \cdot Y]$

On souhaite maintenant tester la fiabilité des différentes fonctions Maple décrites dans la section précédente en les utilisant pour déterminer et confirmer des calculs sur les fonctions Schur. Le but de ce chapitre est donc double puisque l'on souhaite établir des formules sur les fonctions Schur et vérifier que les procédures implementées donnent des résultats sous une forme permettant de retrouver les formules obtenues.

5.1.1 Cas $s_{(n-1,1)}[X \cdot Y]$

On commence par s'intéresser au cas $s_\mu[X \cdot Y]$ où $\mu = (n-1, 1)$.

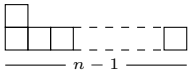


FIGURE 5.1 – μ de forme $(n-1, 1)$

Etant donné que l'on connait une formule pour $h_n[X \cdot Y]$, on va utiliser le déterminant de Jacobi-Trudi (2.4.3) pour exprimer $s_{(n-1,1)}$ en fonction de h_n.

$$s_{(n-1,1)} = \begin{vmatrix} h_{n-1} & h_n \\ 1 & h_1 \end{vmatrix} = h_1 h_{n-1} - h_n$$

On passe en notation plethystique et on utilise l'équation (2.2) du théorème (2.4.1) pour parvenir à l'équation suivante.

$$\begin{aligned} s_{(n-1,1)}[X \cdot Y] &= h_1[X \cdot Y] h_{n-1}[X \cdot Y] - h_n[X \cdot Y] \\ &= s_1(X)s_1(Y) \sum_{\lambda \vdash n-1} s_\lambda(X)s_\lambda(Y) - \sum_{\theta \vdash n} s_\theta(X)s_\theta(Y) \end{aligned}$$

En applicant le cas particulier de la formule de Pieri (3.2.1), on obtient :

$$
\begin{aligned}
s_{(n-1,1)}[X \cdot Y] &= \sum_{\lambda \vdash n-1} s_1(X)s_\lambda(X)s_1(Y)s_\lambda(Y) - \sum_{\theta \vdash n} s_\theta(X)s_\theta(Y) \\
&= \sum_{\lambda \vdash n-1} \left(\sum_{\substack{\mu \vdash n \\ \lambda \subseteq \mu}} s_\mu(X) . \sum_{\substack{\mu \vdash n \\ \lambda \subseteq \mu}} s_\mu(Y) \right) - \sum_{\theta \vdash n} s_\theta(X)s_\theta(Y) \\
&= \sum_{\lambda \vdash n-1} \sum_{\substack{\mu,\nu \vdash n \\ \lambda \subseteq \mu \\ \lambda \subseteq \nu}} s_\mu(X)s_\nu(Y) - \sum_{\theta \vdash n} s_\theta(X)s_\theta(Y) \qquad (5.1)
\end{aligned}
$$

On peut séparer en deux les différents termes intervenant dans (5.1). En effet, on a les termes $s_\mu(X)s_\nu(Y)$ tels que $\mu \neq \nu$, qui n'interviennent que dans la première somme et les termes $s_\mu(X)s_\mu(Y)$ qui apparaissent positivement dans la première somme et négativement dans la deuxième.

Concernant les termes $s_\mu(X)s_\nu(Y)$ ($\mu \neq \nu$), on peut remarquer que les différents μ et ν correspondent à tous les couples de partitions de n tels que μ et ν aient exactement $n-1$ cases en commun.

Ensuite, les termes $s_\mu(X)s_\mu(Y)$ sont obtenus dans la première somme à partir d'une partition λ de taille $n-1$ à laquelle on ajoute une case. On souhaite savoir combien de fois un certain μ est obtenu afin de déterminer le coefficient associé au terme $s_\mu(X)s_\mu(Y)$.

Il suffit de noter que μ est obtenu à chaque fois qu'on a λ tel que $\lambda + \square = \mu$. Dit autrement, étant donné μ, le coefficient associé à $s_\mu(X)s_\mu(Y)$ est donc le nombre de partitions λ qui permettent d'obtenir μ en ajoutant une case, ou encore le nombre de tableaux de taille $n-1$ qui peuvent être obtenus à partir du tableau de μ en lui enlevant une seule case.

On note d_μ le nombre de coins de μ et une case est un coin de μ, si le diagramme obtenu en supprimant cette case est un diagramme de Young c'est-à-dire $d_\mu = \#\{\square \mid \mu \backslash \square \text{ est un diagramme de Young}\}$. d_μ est donc le nombre de façon d'enlever une case a μ. (Voir figure 5.2.)

Notons qu'il faut soustraire 1 à d_μ dans notre cas pour prendre en compte la seconde somme de l'équation, ce qui explique que certains $s_\mu(X)s_\mu(Y)$ n'apparaissent pas dans le résultat final. Par exemple, le terme $s_{33}(X)s_{33}(Y)$ n'apparait pas dans $s_{51}[X \cdot Y]$ puisque $d_{33} = 1$.

Cela explique aussi pourquoi il n'y a pas de terme négatif dans le résultat, puisque le nombre de coins d'un tableau est d'au moins 1.

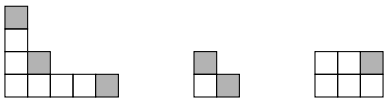


FIGURE 5.2 – $d_{5211} = 3$, $d_{21} = 2$ et $d_{33} = 1$

On parvient finalement à la formule suivante :

$$
\boxed{s_{(n-1,1)} = \sum_{\substack{\mu,\nu \vdash n \\ |\mu \cap \nu| = n-1}} s_\mu(X)s_\nu(Y) + \sum_{\lambda \vdash n} (d_\lambda - 1)s_\lambda(X)s_\lambda(Y)} \qquad (5.2)
$$

Exemples :

$s_{11}[X \cdot Y] = s_2(X)s_{11}(Y) + s_{11}(X)s_2(Y)$

$s_{21}[X \cdot Y] = s_3(X)s_{21}(Y)+s_{21}(X)s_3(Y)+s_{21}(X)s_{21}(Y)+s_{21}(X)s_{111}(Y)+s_{111}(X)s_{21}(Y)$

$s_{31}[X \cdot Y] = s_4(X)s_{31}(Y)+s_{31}(X)s_{31}(Y)+s_{31}(X)s_4(Y)+s_{1111}(X)s_{211}(Y)+s_{22}(X)s_{31}(Y)+ s_{31}(X)s_{211}(Y)+s_{31}(X)s_{22}(Y)+s_{211}(X)s_{22}(Y)+s_{211}(X)s_{31}(Y)+s_{211}(X)s_{1111}(Y)+s_{211}(X)s_{211}(Y)+ s_{22}(X)s_{211}(Y)$

$s_{41}[X \cdot Y] = s_{41}(X)s_5(Y)+s_5(X)s_{41}(Y)+s_{32}(X)s_{32}(Y)+s_{32}(X)s_{41}(Y)+s_{11111}(X)s_{2111}(Y)+ s_{311}(X)s_{221}(Y)+s_{311}(X)s_{311}(Y)+s_{41}(X)s_{311}(Y)+s_{41}(X)s_{32}(Y)+s_{41}(X)s_{41}(Y)+s_{2111}(X)s_{311}(Y)+ s_{311}(X)s_{32}(Y) + s_{311}(X)s_{41}(Y) + s_{2111}(X)s_{11111}(Y) + s_{32}(X)s_{311}(Y) + s_{221}(X)s_{221}(Y) + s_{2111}(X)s_{2111}(Y) + s_{221}(X)s_{2111}(Y) + s_{221}(X)s_{311}(Y) + s_{221}(X)s_{32}(Y) + s_{311}(X)s_{2111}(Y) + s_{32}(X)s_{221}(Y) + s_{2111}(X)s_{221}(Y)$

Les exemples ci-dessus ont été calculés dans Maple et nous permettent de vérifier la cohérence entre les résultats expérimentaux calculés par Maple et la formule théorique obtenue.

Ils permettent également de se rendre compte que du nombre important de termes que peux contenir le résultat.

5.1.2 Cas $s_{(n-k,k)}[X \cdot Y]$

Après le cas où $\mu = (n-1, 1)$, on va s'intéresser au cas où $\mu = (n-k, k)$. On procède alors de façon similaire pour $s_{(n-k,k)}[X \cdot Y]$.

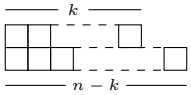


FIGURE 5.3 – μ de forme $(n-k, k)$

$$s_{(n-k,k)} = \begin{vmatrix} h_{n-k} & h_{n-k+1} \\ h_{k-1} & h_k \end{vmatrix} = h_{n-k}h_k - h_{n-k+1}h_{k-1}$$

On passe en notation plethystique et on applique la formule de Cauchy.

$$\begin{aligned} s_{(n-k,k)}[X \cdot Y] &= h_{n-k}[X \cdot Y]h_k[X \cdot Y] - h_{n-k+1}[X \cdot Y]h_{k-1}[X \cdot Y] \\ &= \sum_{\mu \vdash n-k} s_\mu(X)s_\mu(Y). \sum_{\nu \vdash k} s_\nu(X)s_\nu(Y) - \sum_{\mu \vdash n-k+1} s_\mu(X)s_\mu(Y). \sum_{\nu \vdash k-1} s_\nu(X)s_\nu(Y) \\ &= \sum_{\substack{\mu \vdash n-k \\ \nu \vdash k}} s_\mu(X)s_\nu(X)s_\mu(Y)s_\nu(Y) - \sum_{\substack{\mu \vdash n-k+1 \\ \nu \vdash k-1}} s_\mu(X)s_\nu(X)s_\mu(Y)s_\nu(Y) \end{aligned}$$

On utilise maintenant le théorème de Littlewood-Richardson sur la multiplication des fonctions Schur (3.2.2).

$$
\begin{aligned}
s_{(n-k,k)}[X\cdot Y] &= \sum_{\substack{\mu\vdash n-k\\ \nu\vdash k}}\sum_{\lambda\vdash n} c^{\lambda}_{\mu\nu}s_\lambda(X).\sum_{\theta\vdash n} c^{\theta}_{\mu\nu}s_\theta(Y) - \sum_{\substack{\mu\vdash n-k+1\\ \nu\vdash k-1}}\sum_{\lambda\vdash n} c^{\lambda}_{\mu\nu}s_\lambda(X).\sum_{\theta\vdash n} c^{\theta}_{\mu\nu}s_\theta(Y)\\
&= \sum_{\substack{\mu\vdash n-k\\ \nu\vdash k}}\sum_{\lambda,\theta\vdash n} c^{\lambda}_{\mu\nu}c^{\theta}_{\mu\nu}s_\lambda(X)s_\theta(Y) - \sum_{\substack{\mu\vdash n-k+1\\ \nu\vdash k-1}}\sum_{\lambda,\theta\vdash n} c^{\lambda}_{\mu\nu}c^{\theta}_{\mu\nu}s_\lambda(X)s_\theta(Y) \qquad (5.3)
\end{aligned}
$$

$$
\boxed{s_{(n-k,k)}[X\cdot Y] = \sum_{\lambda,\theta\vdash n}\sum_{\substack{\mu_1\vdash n-k\\ \mu_2\vdash n-k+1\\ \nu_1\vdash k\\ \nu_2\vdash k-1}} (c^{\lambda}_{\mu_1\nu_1}c^{\theta}_{\mu_1\nu_1} - c^{\lambda}_{\mu_2\nu_2}c^{\theta}_{\mu_2\nu_2})s_\lambda(X)s_\theta(Y)} \qquad (5.4)
$$

D'après les observations faites sur Maple, il semblerait que pour tous $\lambda, \theta \vdash n$

$$
\sum_{\substack{\mu_1\vdash n-k\\ \mu_2\vdash n-k+1\\ \nu_1\vdash k\\ \nu_2\vdash k-1}} c^{\lambda}_{\mu_1\nu_1}c^{\theta}_{\mu_1\nu_1} - c^{\lambda}_{\mu_2\nu_2}c^{\theta}_{\mu_2\nu_2} \geq 0
$$

Cette inégalité n'a cependant pas été prouvée.

Exemples :

$s_{22}[X\cdot Y] = s_{22}(X)s_{1111}(Y)+s_{1111}(X)s_{22}(Y)+s_{22}(X)s_{22}(Y)+s_{22}(X)s_4(Y)+s_{31}(X)s_{211}(Y)+ s_4(X)s_{22}(Y) + s_{31}(X)s_{31}(Y) + s_{211}(X)s_{211}(Y) + s_{211}(X)s_{31}(Y)$

$s_{32}[X\cdot Y] = s_{32}(X)s_5(Y)+s_{221}(X)s_{41}(Y)+s_{2111}(X)s_{32}(Y)+s_{11111}(X)s_{221}(Y)+s_5(X)s_{32}(Y)+ s_{41}(X)s_{221}(Y)+s_{311}(X)s_{41}(Y)+s_{311}(X)s_{32}(Y)+s_{221}(X)s_{32}(Y)+s_{2111}(X)s_{311}(Y)+s_{221}(X)s_{311}(Y)+ s_{2111}(X)s_{221}(Y)+s_{41}(X)s_{311}(Y)+s_{41}(X)s_{32}(Y)+s_{41}(X)s_{41}(Y)+s_{32}(X)s_{32}(Y)+s_{32}(X)s_{311}(Y)+ s_{32}(X)s_{221}(Y) + s_{311}(X)s_{221}(Y) + 2s_{311}(X)s_{311}(Y) + s_{311}(X)s_{2111}(Y) + s_{221}(X)s_{221}(Y) + s_{221}(X)s_{2111}(Y) + s_{32}(X)s_{41}(Y) + s_{2111}(X)s_{2111}(Y) + s_{32}(X)s_{2111}(Y) + s_{221}(X)s_{11111}(Y)$

Encore une fois, ces résultats ont été obtenus en Maple et sont confirmés par la formule théorique. Ne sont présentés ici que deux cas relativement simples car les développements pour des partitions de taille ou de poids plus importants deviennent rapidement très longs.

On peut aussi appliquer la formule obtenue pour le cas particulier $k = 1$ et vérifier ainsi la cohérence entre les deux formules.

On peut préalablement remarquer que si $\mu \not\subseteq \lambda$ alors $c^{\lambda}_{\mu\nu} = 0$, on peut donc sans modifier le résultat rajouter les conditions que $\mu \subseteq \lambda$ et $\mu \subseteq \theta$.

En repartant de (5.3), on a donc :

$$
\begin{aligned}
s_{(n-1,1)}[X\cdot Y] &= \sum_{\substack{\mu\vdash n-1\\ \nu\vdash 1}}\sum_{\substack{\lambda,\theta\vdash n\\ \mu\subseteq\lambda\\ \mu\subseteq\theta}} c^{\lambda}_{\mu\nu}c^{\theta}_{\mu\nu}s_\lambda(X)s_\theta(Y) - \sum_{\substack{\mu\vdash n\\ \nu\vdash 0}}\sum_{\substack{\lambda,\theta\vdash n\\ \mu\subseteq\lambda\\ \mu\subseteq\theta}} c^{\lambda}_{\mu\nu}c^{\theta}_{\mu\nu}s_\lambda(X)s_\theta(Y)\\
&= \sum_{\mu\vdash n-1}\sum_{\substack{\lambda,\theta\vdash n\\ \mu\subseteq\lambda\\ \mu\subseteq\theta}} c^{\lambda}_{\mu,1}c^{\theta}_{\mu,1}s_\lambda(X)s_\theta(Y) - \sum_{\mu\vdash n}\sum_{\substack{\lambda,\theta\vdash n\\ \mu\subseteq\lambda\\ \mu\subseteq\theta}} c^{\lambda}_{\mu,\varnothing}c^{\theta}_{\mu,\varnothing}s_\lambda(X)s_\theta(Y) \qquad (5.5)
\end{aligned}
$$

$\varnothing$ représente la partition vide.

De plus, dans les sommes du second terme, on remarque que μ, λ et θ sont des partitions du meme entier. Or comme $\mu \subseteq \lambda$ et $\mu \subseteq \theta$, on en déduit que $\mu = \lambda = \theta$.

Enfin, par définition des coefficients de Littlewood-Richardson, on a que $\forall \mu\ c^{\mu}_{\mu,\varnothing} = 1$.

L'équation devient donc :

$$s_{(n-1,1)}[X \cdot Y] \quad = \quad \sum_{\mu \vdash n-1} \sum_{\substack{\lambda,\theta \vdash n \\ \mu \subseteq \lambda \\ \mu \subseteq \theta}} c^{\lambda}_{\mu,1} c^{\theta}_{\mu,1} s_{\lambda}(X) s_{\theta}(Y) - \sum_{\mu \vdash n} s_{\mu}(X) s_{\mu}(Y) \tag{5.6}$$

En comparant l'équation (5.6) que l'on vient d'obtenir à l'équation (5.1) obtenue précédemment pour $s_{(n-1,1)}$, on en déduit le résultat suivant.

Proposition 5.1.1 *$c^{\lambda}_{\mu,1} = 1\ \forall \mu, \lambda$ tels que $\lambda = \mu + \square$ (ie $\mu \vdash (n-1), \lambda \vdash n$ et $\mu \subseteq \lambda$).*

Et finalement, on retrouve bien :

$$s_{(n-1,1)}[X \cdot Y] = \sum_{\mu \vdash n-1} \sum_{\substack{\lambda,\theta \vdash n \\ \mu \subseteq \lambda \\ \mu \subseteq \theta}} s_{\lambda}(X) s_{\theta}(Y) - \sum_{\mu \vdash n} s_{\mu}(X) s_{\mu}(Y)$$

5.1.3 Cas $s_{(n-k,1^k)}[X \cdot Y]$

Enfin, le dernier cas auquel on s'intéresse pour $s_{\mu}[X \cdot Y]$ est le cas où $\mu = (n-k, 1^k)$.

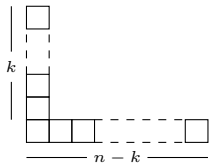


FIGURE 5.4 – μ de forme $(n-k, 1^k)$

Notons que des partitions de la forme $(n-k, 1^k)$ sont appelées *hooks*.

Tout d'abord, on cherche à exprimer $s_{n-k,1^k}$ en fonction de h_n, on va donc l'écrire sous forme du déterminant de Jacodi-Trudi (2.4.3).

$$s_{(n-k,1^k)} = \begin{vmatrix} h_{n-k} & h_{n-k+1} & \cdots & \cdots & h_n \\ 1 & h_1 & \cdots & \cdots & h_{k-1} \\ 0 & 1 & h_1 & \cdots & h_{k-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & h_1 \end{vmatrix}$$

$$= h_{n-k} \underbrace{\begin{vmatrix} h_1 & \cdots & \cdots & h_{k-1} \\ 1 & h_1 & \cdots & h_{k-2} \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & h_1 \end{vmatrix}}_{(1)} - \underbrace{\begin{vmatrix} h_{n-k+1} & \cdots & \cdots & h_n \\ 1 & h_1 & \cdots & h_{k-2} \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & h_1 \end{vmatrix}}_{(2)}$$

On peut alors remarquer que

$$(1) = \begin{vmatrix} h_1 & \cdots & \cdots & h_{k-1} \\ 1 & h_1 & \cdots & h_{k-2} \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & h_1 \end{vmatrix} = s_{1^k} = e_k$$

Puis, en développant le second determinant, on obtient les termes suivants :

$$(2) = h_{n-k+1} \begin{vmatrix} h_1 & \cdots & \cdots & h_{k-2} \\ 1 & h_1 & \cdots & h_{k-3} \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & h_1 \end{vmatrix} - \begin{vmatrix} h_{n-k+2} & \cdots & \cdots & h_n \\ 1 & h_1 & \cdots & h_{k-3} \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & h_1 \end{vmatrix}$$

On peut alors faire les mêmes remarques qu'à l'étape précédente. En continuant de la même façon, on obtient que :

$$s_{(n-k,1^k)} = \sum_{i=0}^{k} (-1)^i h_{n-k+i} e_{k-i} \tag{5.7}$$

On peut maintenant calculer $s_{(n-k,1^k)}[X \cdot Y]$:

$$s_{(n-k,1^k)}[X \cdot Y] = \sum_{i=0}^{k} (-1)^i h_{n-k+i}[X \cdot Y] e_{k-i}[X \cdot Y]$$

On développe $h_{n-k+i}[X \cdot Y]$ et $e_{k-i}[X \cdot Y]$, en utilisant respectivement la formule de Cauchy et la formule de Cauchy duale.

$$\begin{aligned} s_{(n-k,1^k)}[X \cdot Y] &= \sum_{i=0}^{k} (-1)^i \left[\sum_{\mu \vdash n-k+i} s_\mu(X) s_\mu(Y) . \sum_{\nu \vdash k-i} s_{\nu'}(X) s_\nu(Y) \right] \\ &= \sum_{i=0}^{k} (-1)^i \sum_{\substack{\mu \vdash n-k+i \\ \nu \vdash k-i}} s_\mu(X) s_{\nu'}(X) s_\mu(Y) s_\nu(Y) \\ &= \sum_{i=0}^{k} (-1)^i \sum_{\substack{\mu \vdash n-k+i \\ \nu \vdash k-i}} \sum_{\lambda \vdash n} c^\lambda_{\mu\nu'} s_\lambda(X) . \sum_{\theta \vdash n} c^\theta_{\mu\nu} s_\theta(Y) \end{aligned}$$

$$\boxed{s_{(n-k,1^k)}[X \cdot Y] = \sum_{\lambda,\theta \vdash n} \sum_{i=0}^{k} \sum_{\substack{\mu \vdash n-k+i \\ \nu \vdash k-i}} (-1)^i c^\lambda_{\mu\nu'} c^\theta_{\mu\nu} s_\lambda(X) s_\theta(Y)} \tag{5.8}$$

D'après les observations faites sur Maple, il semblerait que pour tous $\lambda, \theta \vdash n$

$$\sum_{i=0}^{k} \sum_{\substack{\mu \vdash n-k+i \\ \nu \vdash k-i}} (-1)^i c_{\mu\nu'}^{\lambda} c_{\mu\nu}^{\theta} \geq 0$$

Cette inégalité n'a cependant pas été prouvée.

Exemples :

$s_{211}[X \cdot Y] = s_{211}(X)s_{22}(Y)+s_{211}(X)s_{31}(Y)+s_{211}(X)s_4(Y)+s_{211}(X)s_{211}(Y)+s_4(X)s_{211}(Y)+ s_{31}(X)s_{22}(Y)+s_{31}(X)s_{31}(Y)+s_{1111}(X)s_{31}(Y)+s_{31}(X)s_{1111}(Y)+s_{31}(X)s_{211}(Y)+s_{22}(X)s_{211}(Y)+ s_{22}(X)s_{31}(Y)$

$s_{2111}[X \cdot Y] = s_{311}(X)s_{311}(Y) + s_{311}(X)s_{32}(Y) + s_{2111}(X)s_5(Y) + s_{41}(X)s_{2111}(Y) + s_{221}(X)s_{311}(Y)+s_{41}(X)s_{221}(Y)+s_{221}(X)s_{32}(Y)+s_{221}(X)s_{41}(Y)+s_{11111}(X)s_{41}(Y)+s_{311}(X)s_{221}(Y)+ s_5(X)s_{2111}(Y) + s_{41}(X)s_{11111}(Y) + s_{311}(X)s_{2111}(Y) + s_{32}(X)s_{2111}(Y) + s_{221}(X)s_{221}(Y) + s_{2111}(X)s_{32}(Y)+s_{2111}(X)s_{41}(Y)+s_{32}(X)s_{221}(Y)+s_{32}(X)s_{32}(Y)+s_{32}(X)s_{311}(Y)+s_{2111}(X)s_{311}(Y)+ s_{311}(X)s_{41}(Y) + s_{41}(X)s_{311}(Y)$

$s_{311}[X \cdot Y] = s_{311}(X)s_5(Y)+s_{311}(X)s_{41}(Y)+2s_{311}(X)s_{32}(Y)+s_{221}(X)s_{32}(Y)+s_{2111}(X)s_{311}(Y)+ 2s_{221}(X)s_{311}(Y)+s_{2111}(X)s_{221}(Y)+s_{41}(X)s_{311}(Y)+s_{41}(X)s_{32}(Y)+s_{41}(X)s_{41}(Y)+s_{32}(X)s_{32}(Y)+ 2s_{32}(X)s_{311}(Y) + s_{32}(X)s_{221}(Y) + 2s_{311}(X)s_{221}(Y) + s_{311}(X)s_{311}(Y) + s_{311}(X)s_{2111}(Y) + s_{221}(X)s_{221}(Y) + s_{221}(X)s_{2111}(Y) + s_{2111}(X)s_{2111}(Y) + s_{41}(X)s_{2111}(Y) + s_{41}(X)s_{221}(Y) + s_{221}(X)s_{41}(Y) + s_{11111}(X)s_{311}(Y) + s_{32}(X)s_{2111}(Y) + s_{2111}(X)s_{32}(Y) + s_5(X)s_{311}(Y) + s_{311}(X)s_{11111}(Y) + s_{32}(X)s_{41}(Y) + s_{2111}(X)s_{41}(Y)$

Encore une fois, les calculs ont été faits sur Maple et sont cohérents avec la formule. Bien que les exemples présentés ici donnent des développements longs, il est à noter qu'il s'agit là des cas les plus simples.

On a établi précédemment plusieurs formules permettant de calculer $s_\mu[X \cdot Y]$ pour des cas particuliers relativement simples. Cependant, il est difficile d'obtenir une formule pour le cas général, néanmoins, le résultat est facilement accessible puisque le programme Maple d'écrit et implémenté permet d'obtenir $s_\mu[X \cdot Y]$ quelque soit μ.

5.2 Evaluation de $f[XY + XZ + YZ]$

On souhaite étudier $s_\mu[XY+XZ+YZ]$ mais pour cela, on a d'abord besoin de connaître $h_n[XY + XZ + YZ]$ ainsi on pourra utiliser la formule de Jacobi-Trudi comme on l'a fait plusieurs fois dans la section précédentes.

5.2.1 $h_n[XY + XZ + YZ]$

Pour le calcul de $h_n[XY + XZ + YZ]$, on va tout d'abord décomposer les sommes en utilisant le fait que $h_n[X + Y] = \sum_{k=0}^{n} h_k(X)h_{n-k}(Y)$ (4.2).

$$\begin{aligned} h_n[XY+XZ+YZ] &= \sum_{k=0}^{n} h_k[XY]h_{n-k}[XZ+YZ] \\ &= \sum_{k=0}^{n} h_k[XY] \sum_{l=0}^{n-k} h_l[XZ]h_{n-k-l}[YZ] \\ &= \sum_{\substack{p+q+r=n \\ 0\leq p,q,r\leq n}} h_p[XY]h_q[XZ]h_r[YZ] \end{aligned} \tag{5.9}$$

(5.9) nous donne une expression de $h_n[XY+XZ+YZ]$ dans la base des h, on utilise maintenant la formule de Cauchy (2.2) sur chacun des h_i pour exprimer $h_n[XY+XZ+YZ]$ dans la base des Schur.

$$\boxed{h_n[XY+XZ+YZ] = \sum_{\substack{p+q+r=n \\ 0\leq p,q,r\leq n}} \sum_{\mu\vdash p} s_\mu(X)s_\mu(Y) \sum_{\nu\vdash q} s_\nu(X)s_\nu(Z) \sum_{\lambda\vdash r} s_\lambda(Y)s_\lambda(Z)} \tag{5.10}$$

De façon analogue, et en utilisant la formule de Cauchy duale, on obtient des formules similaires pour $e_n[XY+XZ+YZ]$.

$$\begin{aligned} e_n[XY+XZ+YZ] &= \sum_{\substack{p+q+r=n \\ 0\leq p,q,r\leq n}} h_p[XY]h_q[XZ]h_r[YZ] \\ &= \sum_{k=0}^{n} \sum_{l=0}^{n-k} \sum_{\mu\vdash k} s_\mu(X)s_{\mu'}(Y) \sum_{\nu\vdash l} s_\nu(X)s_{\nu'}(Z) \sum_{\lambda\vdash n-k-l} s_\lambda(Y)s_{\lambda'}(Z) \end{aligned} \tag{5.11}$$

A partir de (5.10), on peut facilement calculer $h_1[XY+XZ+YZ]$.

$$h_1[XY+XZ+YZ] = s_1(X)s_1(Y) + s_1(X)s_1(Z) + s_1(Y)s_1(Z) \tag{5.12}$$

Cependant, pour $n \geq 2$, le développement est déjà long et on constate l'utilité de la fonction Maple.

$h_2[XY+XZ+YZ] = s_2(X)s_2(Y)+s_{11}(X)s_{11}(Y)+s_1(X)^2s_1(Y)s_1(Z)+s_1(X)s_1(Y)^2s_1(Z)+ s_2(X)s_2(Z) + s_{11}(X)s_{11}(Z) + s_1(X)s_1(Z)^2s_1(Y) + s_2(Y)s_2(Z) + s_{11}(Y)s_{11}(Z)$

$e_2[XY+XZ+YZ] := s_2(X)s_{11}(Y)+s_{11}(X)s_2(Y)+s_1(X)^2s_1(Y)s_1(Z)+s_1(X)s_1(Y)^2s_1(Z)+ s_2(X)s_{11}(Z) + s_{11}(X)s_2(Z) + s_1(X)s_1(Z)^2s_1(Y) + s_2(Y)s_{11}(Z) + s_{11}(Y)s_2(Z)$

On rappelle que $h_n = s_n$ et $e_{1^n} = s_{1^n}$, on a donc $h_1[XY+XZ+YZ] = s_1[XY+XZ+YZ] = e_1[XY+XZ+YZ]$, $h_2[XY+XZ+YZ] = s_2[XY+XZ+YZ]$, $e_2[XY+XZ+YZ] = s_{11}[XY+XZ+YZ]$, etc...

5.2.2 $s_{(n-1,1)}[XY+XZ+YZ]$

Grâce aux deux formules précédentes (5.10) et (5.12) et à la formule de Jacobi-Trudi (2.4.3), on va pouvoir obtenir $s_{n-1,1}[XY+XZ+YZ]$ en utilisant un raisonnement analogue à celui utilisé pour obtenir $s_{n-1,1}[XY]$.

$$s_{(n-1,1)}[XY+XZ+YZ] = h_1[XY+XZ+YZ]h_{n-1}[XY+XZ+YZ]-h_n[XY+XZ+YZ]$$

$$= (s_1(X)s_1(Y) + s_1(X)s_1(Z) + s_1(Y)s_1(Z))$$
$$\times \left[\sum_{k=0}^{n-1}\sum_{l=0}^{n-k-1}\sum_{\mu\vdash k} s_\mu(X)s_\mu(Y)\sum_{\nu\vdash l} s_\nu(X)s_\nu(Z)\sum_{\lambda\vdash n-k-l-1} s_\lambda(Y)s_\lambda(Z)\right]$$
$$-\sum_{k=0}^{n}\sum_{l=0}^{n-k}\sum_{\substack{\mu\vdash k\\ \nu\vdash l\\ \lambda\vdash n-k-l}} s_\mu(X)s_\mu(Y)s_\nu(X)s_\nu(Z)s_\lambda(Y)s_\lambda(Z)$$

On va maintenant utiliser la formule de Pieri (3.2.1) afin de multiplier chaque $s_1(A)s_1(B)$ avec la somme $\sum s_\mu(A)s_\mu(B)$ correspondente. ($(A,B)\in\{(X,Y),(X,Z),(Y,Z)\}$)

$$s_{n-1,1}[XY+XZ+YZ] = \sum_{k=0}^{n-1}\sum_{l=0}^{n-k-1}\sum_{\substack{\mu\vdash k\\ \nu\vdash l\\ \lambda\vdash n-k-l-1}}\left[\sum_{\substack{\mu_1,\mu_2\vdash k+1\\ \mu\subseteq\mu_1,\mu_2}} s_{\mu_1}(X)s_{\mu_2}(Y)s_\nu(X)s_\nu(Z)s_\lambda(Y)s_\lambda(Z)\right.$$
$$\left.+\sum_{\substack{\nu_1,\nu_2\vdash l+1\\ \nu\subseteq\nu_1,\nu_2}} s_\mu(X)s_\mu(Y)s_{\nu_1}(X)s_{\nu_2}(Z)s_\lambda(Y)s_\lambda(Z) + \sum_{\substack{\lambda_1,\lambda_2\vdash n-k-l\\ \lambda\subseteq\lambda_1,\lambda_2}} s_\mu(X)s_\mu(Y)s_\nu(X)s_\nu(Z)s_{\lambda_1}(Y)s_{\lambda_2}(Z)\right]$$
$$-\sum_{k=0}^{n}\sum_{l=0}^{n-k}\sum_{\substack{\mu\vdash k\\ \nu\vdash l\\ \lambda\vdash n-k-l}} s_\mu(X)s_\mu(Y)s_\nu(X)s_\nu(Z)s_\lambda(Y)s_\lambda(Z)$$

L'équation obtenue présente des similitudes avec 5.1. En effet, dans chacune des trois sommes entre crochets, on peut distinguer les cas où $\mu_1=\mu_2$ (resp. $\nu_1=\nu_2$, resp. $\lambda_1=\lambda_2$) dans $\sum s_{\mu_1}(X)s_{\mu_2}(Y)$ (resp. $s_{\nu_1}(X)s_{\nu_2}(Y)$, resp. $s_{\lambda_1}(X)s_{\lambda_2}$), des cas où les deux partitions sont différentes.

Comme pour (5.1), les deux partitions sont différentes lorsque l'on compte les partitions ayant exactement k (resp. l, resp. $n-k-l-1$) cases en commun.

Lorsque les deux partitions sont identiques, on va décrire le cas $\mu_1=\mu_2$ (les autres cas étant similaires). On remarque que l'on va obtenir un terme ayant trois couples de Schur où les deux éléments de chaque paire sont indicés par les mêmes partitions :
$s_{\mu_1}(X)s_{\mu_1}(Y)s_\nu(X)s_\nu(Y)s_\lambda(X)s_\lambda$.
Une fois encore, le nombre de fois où le cas $\mu_1=\mu_2$ va se produire est compté par le nombre de coins de μ dans la première somme entre crochets (d_μ) ; le nombre de fois où le cas $\nu_1=\nu_2$ va se produire est compté par le nombre de coins de ν dans la seconde somme entre crochets (d_ν) ; et enfin le nombre de fois où le cas $\lambda_1=\lambda_2$ va se produire est compté par le nombre de coins de λ dans la troisième somme entre crochets (d_λ).
Enfin, on remarque les termes soustraits dans la dernière partie de la formule sont ceux dans lesquels l'un des cas précédent s'est produit, ainsi le nombre de termes de la forme $s_\mu(X)s_\mu(Y)s_\nu(X)s_\nu(Y)s_\lambda(X)s_\lambda$ que l'on va obtenir est $d_\mu+d_\nu+d_\lambda-1$.

On obtient ainsi le résultat suivant.

$$s_{n-1,1}[XY+XZ+YZ] = \sum_{k=0}^{n-1}\sum_{l=0}^{n-k-1}\sum_{\substack{\mu\vdash k\\ \nu\vdash l\\ \lambda\vdash n-k-l-1}}\Bigg[\underbrace{\sum_{\substack{\mu_1,\mu_2\vdash k+1\\ |\mu_1\cap\mu_2|=k}} s_{\mu_1}(X)s_{\mu_2}(Y)s_\nu(X)s_\nu(Z)s_\lambda(Y)s_\lambda(Z)}$$

$$+\underbrace{\sum_{\substack{\nu_1,\nu_2\vdash l+1\\ |\nu_1\cap\nu_2|=l}} s_\mu(X)s_\mu(Y)s_{\nu_1}(X)s_{\nu_2}(Z)s_\lambda(Y)s_\lambda(Z)} + \underbrace{\sum_{\substack{\lambda 1,\lambda 2\vdash n-k-l\\ |\lambda_1\cap\lambda_2|=n-l-k-1}} s_\mu(X)s_\mu(Y)s_\nu(X)s_\nu(Z)s_{\lambda_1}(Y)s_{\lambda_2}(Z)}\Bigg]$$

$$+\sum_{k=0}^{n}\sum_{l=0}^{n-k}\sum_{\substack{\mu\vdash k\\ \nu\vdash l\\ \lambda\vdash n-k-l}}(d_\mu+d_\nu+d_\lambda-1)s_\mu(X)s_\mu(Y)s_\nu(X)s_\nu(Z)s_\lambda(Y)s_\lambda(Z) \quad (5.13)$$

On peut vérifier la cohérence entre la formule obtenue et le programme Maple sur un exemple, le plus simple étant $s_{21}[XY+XZ+YZ]$.

$$\begin{aligned}
s_{21}[XY+XZ+YZ] &= s_1(X)s_1(Y)s_2(X)s_{11}(Z) + s_2(X)s_{11}(Z)s_1(Y)s_1(Z)\\
&+ s_1(X)s_1(Y)s_2(Y)s_{11}(Z) + s_1(X)s_1(Y)s_{11}(Y)s_2(Z) + s_2(X)s_{11}(Y)s_1(Y)s_1(Z)\\
&+ s_{11}(X)s_2(Y)s_1(Y)s_1(Z) + s_2(X)s_{11}(Y)s_1(X)s_1(Z) + s_{11}(X)s_2(Y)s_1(X)s_1(Z)\\
&+ s_1(X)s_1(Z)s_2(Y)s_{11}(Z) + s_1(X)s_1(Z)s_{11}(Y)s_2(Z) + s_1(X)s_1(Y)s_{11}(X)s_2(Z)\\
&+ s_{11}(X)s_2(Z)s_1(Y)s_1(Z) + s_1(X)s_1(Z)s_{11}(Y)s_{11}(Z) + s_1(X)s_1(Z)s_2(Y)s_2(Z)\\
&+ s_2(X)s_2(Z)s_1(Y)s_1(Z) + s_{11}(X)s_{11}(Y)s_1(Y)s_1(Z) + s_1(X)s_1(Y)s_2(X)s_2(Z)\\
&+ s_2(X)s_2(Y)s_1(X)s_1(Z) + s_1(X)s_1(Y)s_{11}(X)s_{11}(Z) + s_{11}(X)s_{11}(Y)s_1(X)s_1(Z)\\
&+ s_{11}(X)s_{11}(Z)s_1(Y)s_1(Z) + s_1(X)s_1(Y)s_{11}(Y)s_{11}(Z) + s_1(X)s_1(Y)s_2(Y)s_2(Z)\\
&+ s_2(X)s_2(Y)s_1(Y)s_1(Z) + s_{21}(X)s_3(Y) + s_{111}(X)s_{21}(Y) + s_3(X)s_{21}(Y) + s_{21}(X)s_{111}(Y)\\
&+ s_{21}(X)s_{21}(Z) + s_{21}(Y)s_{21}(Z) + s_{21}(X)s_{21}(Y) + s_{21}(Y)s_3(Z) + s_{111}(Y)s_{21}(Z) + s_3(Y)s_{21}(Z)\\
&+ s_{21}(Y)s_{111}(Z) + s_{21}(X)s_3(Z) + s_3(X)s_{21}(Z) + s_{111}(X)s_{21}(Z) + s_{21}(X)s_{111}(Z)\\
&+ 2s_1(X)^2s_1(Y)^2s_1(Z)^2
\end{aligned}$$

Les termes apparaissant dans $s_{21}[XY+XZ+YZ]$ ci-dessus, on été colorés en fonction de la somme qui les produit. Ainsi, les termes en rouge sont ceux issus de la première somme entre crochets, en effet, on constate que tous les termes rouges contiennent les éléments $s_{\mu_1}(X)s_{\mu_2}(Y)$ avec $\mu_1 \neq \mu_2$.

Les termes en bleu sont ceux produits par la seconde somme entre crochets dans laquelle $\nu_1 \neq \nu_2$. Et les termes en vert, sont ceux issus de la troisième somme entre crochets dans laquelle $\lambda_1 \neq \lambda_2$.

Enfin, dans chaque terme laissé en noir, les deux éléments de chaque paire sont indicés par les mêmes partitions. Les termes en noir sont donc produits par la dernière des sommes.

On peut par exemple détailler le cas du terme $2s_1(X)^2s_1(Y)^2s_1(Z)^2$ qui vient de la dernière somme lorsque μ, ν et λ sont des partitions de 1. Dans ce cas, $\mu = \nu = \lambda = \square$ et

$d_\mu = d_\nu = d_\lambda = 1$, donc le coefficient est bien $1 + 1 + 1 - 1 = 2$.

Bien que l'on ait travaillé sur une partition de forme à priori simple $(n-1, 1)$, on obtient une formule complexe. Il semble donc que trouver une formule générale soit un problème difficile.

On constate donc encore une fois l'intérêt de la programmation d'une fonction permettant de réaliser ces calculs sans même connaitre de formule explicite.

De plus, on constate que le résultat donné par le programme permet de se rapporter à la formule théorique sans manipulations préalables.

5.3 Spécialisation des alphabets

On a vu que l'on peut facilement obtenir des formules très longues et peu agréables. Un moyen efficace pour réduire la taille de ces formules consiste à considérer la taille des alphabets sur lesquels on évalue la fonction.

En effet, pour X un alphabet de k variables, on sait que $s_\mu[X] = 0$ si le nombre de parts de μ est strictement supérieur à k c'est-à-dire si $l(\mu) > k$.

De même, en considérant toujours X un alphabet de k variables, on a que $e_n[X] = 0$ si $n > k$.

On ajoute donc aux fonctions déjà développées une fonction qui permet en spécifiant la taille des alphabets de simplifier les formules obtenues en supprimant les termes qui s'annulent. On peut retrouver un exemple d'utilisation de cette fonction où la taille peut-être significativement réduite en annexe A.

On peut aussi reprendre l'exemple de $s_{21}[XY + XZ + YZ]$ dans le cas où X et Y sont des alphabets de taille 2 et Z un alphabet de taille 1 , on a alors :

$$\begin{aligned}
s_{21}[XY + XZ + YZ] = {} & s_{21}(X)s_3(Z) + s_{21}(Y)s_3(Z) + s_{21}(X)s_3(Y) + s_3(X)s_{21}(Y) \\
& + s_{21}(X)s_{21}(Y) + s_{11}(X)s_{11}(Y)s_1(X)s_1(Z) + s_2(X)s_2(Z)s_1(Y)s_1(Z) \\
& + s_{11}(X)s_2(Z)s_1(Y)s_1(Z) + s_1(X)s_1(Y)s_2(X)s_2(Z) + s_1(X)s_1(Y)s_{11}(X)s_2(Z) \\
& + 2s_1(X)^2s_1(Y)^2s_1(Z)^2 + s_1(X)s_1(Y)s_2(Y)s_2(Z) + s_1(X)s_1(Y)s_{11}(Y)s_2(Z) \\
& + s_2(X)s_2(Y)s_1(Y)s_1(Z) + s_2(X)s_{11}(Y)s_1(Y)s_1(Z) + s_{11}(X)s_2(Y)s_1(Y)s_1(Z) \\
& + s_{11}(X)s_{11}(Y)s_1(Y)s_1(Z) + s_1(X)s_1(Z)s_2(Y)s_2(Z) + s_1(X)s_1(Z)s_{11}(Y)s_2(Z) \\
& + s_2(X)s_2(Y)s_1(X)s_1(Z) + s_2(X)s_{11}(Y)s_1(X)s_1(Z) + s_{11}(X)s_2(Y)s_1(X)s_1(Z)
\end{aligned}$$

Chapitre 6

Décomposition de fonctions Schur gauches

6.1 Constructions sur les fonctions Schur gauches

Dans ce chapitre, on s'intéresse à exprimer une fonction Schur gauche $s_{\mu/\nu}$ en une somme de deux fonctions Schur gauches (ou plus) tel que

$$s_{\mu/\nu} = s_{\alpha/\beta} + s_{\gamma/\delta}$$

où α/β et γ/δ sont des formes gauches.

Nous allons tout d'abord définir plusieurs constructions sur les fonctions Schur gauches que l'on peut retrouver dans [11].

Afin de simplifier l'écriture, on va écrire les formes gauches avec des lettres majuscules, ainsi $D = \mu/\nu$ par exemple.

Soient D_1 et D_2 deux diagrammes gauches, l'***union disjointe*** de D_1 et D_2, noté $D_1 \oplus D_2$, est le diagramme obtenu en plaçant D_2 une position au sud et une position à l'est de D_1 de façon à ce que D_1 et D_2 n'occupent ni les mêmes lignes, ni les mêmes colonnes.

Par exemple, $D_1 = 22$ et $D_2 = 32/1$ alors

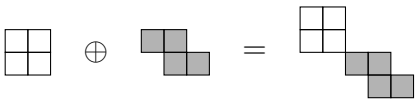
$D_1 \oplus D_2 =$ 

FIGURE 6.1 – Exemple d'union disjointe de Schur gauches

L'union disjointe de diagrammes gauche nous permet de donner la définition suivante sur les diagrammes gauches.

Définition 6.1.1 *On dit qu'un diagramme gauche D est* connecté *s'il ne peut pas s'écrire comme l'union disjointe $D_1 \oplus D_2$, où D_1 et D_2 sont deux sous-diagrammes de D.*

FIGURE 6.2 – $D = 332/21$

Par exemple $D = 332/21$ est connecté alors que l'union disjointe figure 6.5 est bien évidemment non connectée.

Dans la suite, lorsque l'on parlera de diagramme ou de forme non connecté(e), on considèrera toujours que l'on aura une forme compacte, c'est-à-dire que $D = \mu/\nu$ avec $l(\mu) > l(\nu)$ et D ne contient pas de ligne ou de colonne vide.

En effet, si $l(\mu) = l(\nu)$ un décalage à gauche nous permet de nous ramener à une forme compacte sans changer la valeur de la fonction Schur associée. De même, si D contient des lignes et/ou des colonnes vides, on peut supprimer ces lignes et/ou ces colonnes pour se ramener à une forme compacte sans changer la valeur de la fonction Schur associée.

Proposition 6.1.1 *Soit D un diagramme non connecté tel que $D = D_1 \oplus D_2$, alors*

$$s_D = s_{D_1} s_{D_2}$$

Preuve

D'après la définition combinatoire des fonctions Schur vue au chapitre 2, $s_D = \sum_\tau X_\tau$ où τ est un remplissage de D et $X_\tau = \prod_{(i,j)\in D} x_{\tau(i,j)}$.

Si on différencie les cases qui sont dans D_1 de celles qui sont dans D_2, on obtient que

$$s_D = \sum_\tau \prod_{(i,j)\in D_1} x_{\tau(i,j)} \prod_{(i,j)\in D_2} x_{\tau(i,j)}$$

On peut également décomposer τ en τ_1 et τ_2 qui sont respectivement les remplissages de D_1 et D_2. Ce qui nous donne finalement que

$$s_D = \sum_{\tau_1} \prod_{(i,j)\in D_1} x_{\tau_1(i,j)} \sum_{\tau_2} \prod_{(i,j)\in D_2} x_{\tau_2(i,j)} = s_{D_1} s_{D_2}$$

□

On va maintenant définir deux autres constructions sur les fonctions Schur gauches qui sont la concaténation, notée $\cdot$ et la quasi-concaténation, notée $\odot$.

Soient A et B deux formes gauches, la ***concaténation*** $A \cdot B$ est obtenue à partir de l'union disjointe $A \oplus B$ en déplaçant B d'une colonne vers la gauche de façon à ce que la colonne la plus à droite de A et celle la plus à gauche de B soient maintenant dans la même colonne.

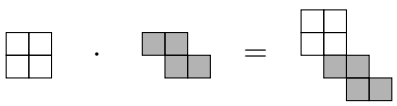


FIGURE 6.3 – Exemple d'une concaténation de Schur gauches

De façon analogue, la ***quasi-concaténation*** $A \odot B$ est obtenue à partir de l'union disjointe $A \oplus B$ en déplaçant B d'une ligne vers le haut de façon à ce que la ligne la plus basse de A et celle la plus haute de B soient maintenant dans la même ligne.

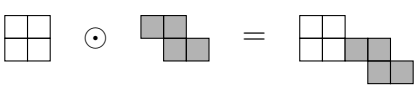


FIGURE 6.4 – Exemple d'une quasi-concaténation de Schur gauches

6.2 Décomposition de Schur gauches non connectés

Grâce aux constructions de [11] que l'on vient de rappeler, on va maintenant pouvoir donner la proposition issue du même article qui va nous permettre d'écrire des fonctions Schur gauches non connectés en somme de deux fonctions Schur gauches.

Proposition 6.2.1 *Si A et B sont deux diagrammes gauches alors*

$$s_A s_B = s_{A \cdot B} + s_{A \odot B}$$

Preuve
Soient A et B deux formes gauches. Soit a l'entier contenu dans la case la plus au nord-est de A et b l'entier contenu dans la case la plus au sud-ouest de B.
On sait que $s_A s_B$ correspond à une somme sur les remplissages possibles de A et B. Deux cas peuvent alors se présenter lors de ces remplissages :

— $a > b$ et alors la concaténation $A \cdot B$ va donner un tableau semi-standard avec le même remplissage.
— $a \leq b$ et alors la quasi-concaténation $A \odot B$ va donner un tableau semi-standard avec le même remplissage.

□

Les propositions 6.1.1 et 6.2.1 nous permettent d'obtenir une décomposition des fonctions Schur gauches non connectés en une somme de deux Schur gauches connectés.

Théorème 6.2.1 *Soit D une forme non connectée telle que $D = D_1 \oplus D_2$ alors*

$$s_D = s_{D_1 \cdot D_2} + s_{D_1 \odot D_2}$$

Exemple : $s_{5422/32} = s_{4322/21} + s_{542/3}$

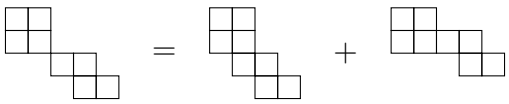


FIGURE 6.5 – Exemple : décomposition de $D = 5422/32$

Pour les formes gauches ayant plus de un point de non connexion, il est possible de décomposer la fonction Schur associée en plus de deux termes. En effet, on commence par effectuer la décomposition en deux termes comme on vient de le voir, puis on peut encore décomposer l'une des fonctions obtenues ou les deux et obtenir ainsi plus de termes jusqu'au nombre voulu ou jusqu'à ce que toutes les formes obtenues soient connectées. Cela n'est cependant possible que dans un petit nombre de cas puisqu'il faut chaque fois avoir un point de non connexion pour effectuer une décomposition.

Le théorème 6.2.1 nous permet de programmer sur Maple une procédure qui prend en paramètre une fonction Schur gauche et qui, si elle est non connectée, renvoie en sortie sa décomposition.

Une forme gauche $D = \mu/\nu$ est non connectée si il existe un entier i tel que $\mu_{i+1} = \nu_i$. Dans notre cas, on choisit de garder le plus petit i s'il en existe plusieurs, c'est-à-dire que

l'on va effectuer la décomposition au niveau du premier point de non connexion à partir du haut de la forme gauche considérée. Ainsi, les lignes de D situées au-dessus de i feront partie de D_2 alors que celles en dessous formeront D_1.

Ayant identifié D_1 et D_2, il nous faut donc maintenant construire $D_1 \cdot D_2$ et $D_1 \odot D_2$.

Pour construire $D_1 \cdot D_2$, il faut remarquer que toutes les lignes de D_2 sont décalées d'une case vers la gauche. Donc pour $1 \leq j \leq i$, on va soustraire 1 aux μ_j et ν_j.

Pour construire $D_1 \odot D_2$, on considère que cette fois c'est D_1 qui est déplacé d'une case vers le haut. Ainsi, pour $i+1 \leq j \leq l(\mu)-1$, μ_j va devenir μ_{j+1} et $\mu_{l(\mu)}$ devient nul ; et pour pour $i \leq j \leq l(\nu)-1$, ν_j va devenir ν_{j+1} et $\nu_{l(\nu)}$ devient nul.

La fonction Maple obtenue est la suivante :

```
> decompose:=proc(skew) local mu,nu,imu,inu,res,i,j,snu,smu;
>    mu:=op(1,skew); imu:=nops(mu);
>    nu:=op(2,skew); inu:=nops(nu);
>    if not imu>inu then
>       ERROR("Skew Schur not disconnected or not compact"); fi;
>    i:=0;
>    for j from 1 by 1 to inu while i=0 do
>        if nu[j]=mu[j+1] then i:=j; fi;
>    od;
>    if i=0 or (inu>i and mu[i+1]-nu[i+1]=0) then
>      ERROR("Skew Schur non disconnected or non compact");
>    else
>      # Construction de la concatenation
>      smu:=seq(j=mu[j]-1,j=1..i);
>      snu:=seq(j=nu[j]-1,j=1..i);
>      res:=s[subsop(smu,mu),subsop(snu,nu)];
>      # Construction de la quasi-concatenation
>      smu:=seq(j=mu[j+1],j=i+1..imu-1),imu=NULL;
>      snu:=seq(j=nu[j+1],j=i..inu-1),inu=NULL;
>      res:=res+s[subsop(smu,mu),subsop(snu,nu)];
>    fi;
> end:
```

6.3 Décomposition de Schur gauches connectés

(Dans cette section, on ne présente pas une façon de décomposer les Schur gauches non connctés mais seulement quelques remarques.)

Suite à quelques manipulations sur Maple, on se rend compte rapidement que la décomposition des fonctions Schur gauches en une sommes de deux fonctions Schur gauches n'est pas unique.

En effet, soit D une forme gauche (ou un diagramme de Young), la rotation antipodale de D, notée D°, est la rotation à 180 degrés de D. Et d'après [7] on a que $s_D = s_{D^\circ}$. Il est donc clair que si $s_D = s_A + s_B$ alors, on a aussi $s_D = s_{A^\circ} + s_B$ ainsi que toutes les autres égalités que l'on peut écrire en utilisant ou non la rotation antipodale de A et/ou de B.

Cependant, sans même utiliser la rotation antipodale, il est parfois possible de trouver plusieurs décompositions pour une même fonction Schur.

Exemple :

$$s_{5332/221} = s_{521} + s_{5111} + s_{431} + s_{422} + 2s_{4211} + s_{332} + s_{3311} + s_{3221}$$

On peut décomposer $s_{5332/221}$ en : $s_{5332/221} = s_{4321/11} + s_{5554/443}$
Par rotation antipodale, $(5554/443)^\circ = 5211/1$, donc on a aussi que
$s_{5332/221} = s_{4321/11} + s_{5211/1}$

Mais on peut aussi décomposer $s_{5332/221}$ en : $s_{5332/221} = s_{5443/332} + s_{333/1}$

$s_{4321/11} = s_{431} + s_{422} + s_{4211} + s_{332} + s_{3311} + s_{3221}$
$s_{5554/443} = s_{5211/1} = s_{521} + s_{5111} + s_{4211}$
$s_{5443/332} = s_{521} + s_{5111} + s_{431} + s_{422} + 2s_{4211} + s_{3311} + s_{3221}$
$s_{333/1} = s_{332}$

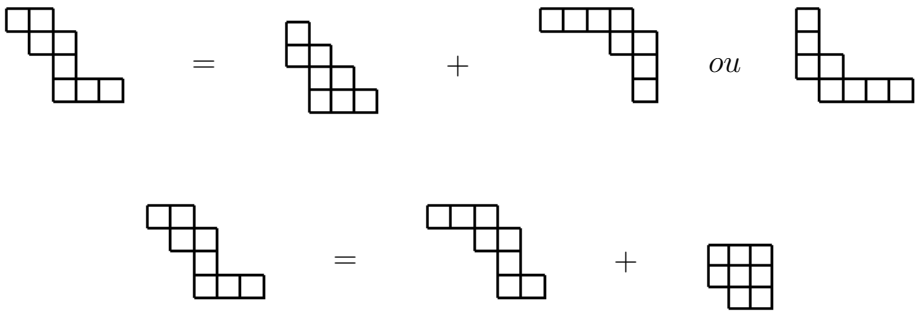


FIGURE 6.6 – Décomposition de $s_{5332/221}$

Une autre conséquence du fait que $s_D = s_{D^\circ}$, est que certaines fonctions Schur gauches ne sont pas décomposables. En effet, dans le cas où $D = \mu/\nu$ et μ est un rectangle, c'est-à-dire que toutes les lignes de μ font la même taille, alors D n'est pas décomposable.

Cela s'explique par le fait que la rotation antipodale de μ/ν dans ce cas, est un diagramme de Young classique et non plus une forme gauche. D s'ecrit donc dans la base des Schur comme un seul élément et n'est donc pas décomposable en plusieurs Schur gauches.

Dans l'exemple précédent, figure 6.6, 333/1 est une forme gauche construite à partir d'un rectangle et on constate en effet que $s_{333/1} = s_{332}$, sa fonction Schur égale à celle de sa rotation antipodale qui est une fonction Schur classique, on ne pourra donc pas décomposer $s_{333/1}$ en une somme de Schur gauches.

Conclusion

Ce stage de fin d'études au LaCIM fut très enrichissant à différents niveaux. Tout d'abord, il m'a permis d'utiliser les compétences acquises tout au long de ma formation dans un cadre différent mais également de rencontrer des personnes ayant des points de vues différents sur les mathématiques et des façons de travailler différentes. Cela m'a permis d'avoir une autre vision des mathématiques et une nouvelle façon d'aborder mon travail.

De plus, ce stage m'a permis d'acquérir non seulement des connaissances en combinatoire, notamment sur les fonctions symétriques à travers mon travail, mais aussi sur d'autres notions grâce à des séminaires ou à des rencontres. J'ai aussi eu l'occasion de lire de nombreux articles ou textes scientifiques ce que j'avais peu fait auparavant et d'apprendre plus en détails la programmation en Maple que je connaissais peu. Cette expérience m'a également permise d'améliorer mon anglais aussi bien scientifique que quotidien à travers des échanges, des lectures et des séminaires en anglais. Ce stage fut donc enrichissant d'un point de vue à la fois professionnel et personnel.

Enfin, ce stage fut une expérience particulièrement positive pour moi puisqu'il m'a donné l'envie de poursuivre mes études en doctorat et a confirmé mon intérêt pour la combinatoire.

Bibliographie

[1] Site web de l'UQAM pour le département mathématiques. `http://www.math.uqam.ca`.

[2] Site web du LaCIM. `http://lacim.uqam.ca`.

[3] F. Bergeron. Introduction to symetric function.

[4] J. Haglund. The q,t-Catalan numbers and the space of diagonal harmonics with an appendix on the combinatorics of Macdonald polynomials. *Mathematics Subject Classification*, 1991.

[5] D. E. Littlewood and A. R. Richardson. Group characters and algebra. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, pages pp. 115–124, 1934.

[6] I. G. Macdonald. *Symmetric functions and Hall polynomials*. Oxford mathematical monographs. Oxford University Press, 1979.

[7] R.P. Stanley. *Enumerative Combinatorics vol. 2*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

[8] J. Stembridge. SF : the symetric functions package. Help texts for the Vanilla edition of SF 2.4.

[9] J. Stembridge. The SF package. Disponible sur `http://www.math.lsa.umich.edu/~jrs/maple.html`.

[10] J. Stembridge. A Maple package for symetric function. nov 2005.

[11] S. van Willigenburg V. Reiner, K.M. Shaw. Coincidences among skew schur functions. *Adv. Math. 216 :118–152*, 2007.

Annexe A

Code Maple

Extension au package SF de J. Stembridge

```
> read("/Volumes/phubert/SF2.4v.txt");
> withSF();
```

SF 2.4v loaded.

A.1 Création de bases

Création des bases de la même façon que dans le package SF (fonctions dual_basis et add_basis) mais en définissant en plus la fonction To'b' où b est le nom de la nouvelles base. To'b' permet de convertir une fonctions symétrique homogène écrite dans n'importe laquelle des deux conventions dans la base b. La fonction de départ peut contenir des Schur gauches.

```
> Dual_basis:=proc(b1,b2) local a;
>    dual_basis(b1,b2);
>    if type(b1,'indexed') then a:=op(0,b1) else a:=b1 fi;
>    assign(cat(`To`,a)=proc(f) local res;
>       res:=Tos(f);
>       res:=cat(`to`,a)(res);
>       end);
>    print(Okay);
> end:

> Add_basis:=proc(b,ip) local a;
>    add_basis(b,ip);
>    if type(b,'indexed') then a:=op(0,b) else a:=b fi;
>    assign(cat(`To`,a)=proc(f) local res;
>       res:=Tos(f);
>       res:=cat(`to`,a)(res);
>       end);
>    print(Okay);
> end:
```

Creation de la base des monomiales m

```
> Dual_basis(m,h);
```

Okay

A.2 Transformation d'écriture

A.2.1 De SF package vers SF avec élément indexés par des partitions

Converti les elements de e, p et h en elements indexés. Si l'élément est déjà un élément de type indexé il reste inchangé

```
> elmtInd:=proc(m) local n,i,name,c;
>    if type(m,'indexed') then
>       return m;
>    else
>       c:=convert(op(m),'string');
>       name:=convert(substring(c,1),'symbol');
>       if member(name,`SF/Bases`) then
>          n:=parse(substring(c,2..length(c)));
>          name[n];
>       else return m;
>       fi;
>    fi;
> end:
```

Utilise la fonction précédente pour convertir chaque élément d'une expression en élément indexé

```
> SFtoInd:=proc(exp)
>    if op(0,exp)=Integer or op(0,exp)=Fraction or type(exp,'indexed')then
>        return exp;
>    elif nops(exp)=2 and op(0,exp)=`^` then
>        SFtoInd(op(1,exp))^op(2,exp);
>    elif nops(exp)=1  then
>       elmtInd(exp);
>    else
>       convert([seq(SFtoInd(k),k in op(exp))],op(0,exp));
>    fi;
> end:
```

Utilise la fonction précédente pour convertir tous les éléments d'une expression en éléments indicés puis indexe les bases multiplicatives par des partitions

```
> SFtoPart:=proc(f) local exp,s;
>    exp:=expand(SFtoInd(f));
>    if op(0,exp)=`+` then
>       s:=op(exp);
>       convert([seq(SFtoPart(i),i in s)],`+`);
>    elif op(0,exp)=`*` then
>       exp:=search_basis(exp,p);
>       exp:=search_basis(exp,h);
>       exp:=search_basis(exp,e);
>    elif op(0,exp)=`^` then
>       return transPower(exp);
>    else return exp;
>    fi;
> end:
```

```
> search_basis:=proc(f,b) local s,sq,i,pw,t,res;
>    if type(f,indexed) then return f; fi;
>    s:=op(f); sq:=NULL; res:=1;
>    for i in s do
>       if op(0,i)=b then
>          sq:=sq,op(1,i);
>       elif op(0,i)=`^` then
>          t:=op(1,i);
>          if op(0,t)=b then
>             sq:=sq,seq(op(1,t),j=1..op(2,i));
>          else res:=res*i; fi;
>       else res:=res*i;
>       fi;
>    od;
>    if sq=NULL then return res;
>    else return res*b[op(sort([sq],`>`))];
>    fi;
> end:

> transPower:=proc(f) local t,b,sq;
>    t:=op(1,f);
>    b:=op(0,t);
>    sq:=seq(op(1,t),i=1..op(2,f));
>    b[sq];
> end:
```

A.2.2 Transformation inverse

Effectue la transformation d'une expression à éléments indicés par des partitions en une expression écrite selon les conventions définies dans SF

```
> partToSF:=proc(exp)
>    if op(0,exp)=`+` or op(0,exp)=`*` then
>       convert([seq(partToSF(i),i in op(exp))],op(0,exp));
>    elif op(0,exp)=Integer or op(0,exp)=Fraction then
>       return exp;
>    elif type(exp,'indexed') then
>       if member(op(0,exp),`SF/Bases`) then
>          mul(cat(op(0,exp),n),n in op(exp));
>       else return exp; fi;
>    else return exp;
>    fi;
> end:
```

A.2.3 Exemples

```
> SFtoPart(2+8*(1/2)*p2*e3*e1*e3+(1/3)*h1*h5*h1);
> SFtoPart(h10+s[4,2,2]+p3*p1);
> SFtoPart(h10+s[4,2,2]*p3*p1+e1*e1*e1);
```

$$2 + 4\, p_2 e_{3,3,1} + 1/3\, h_{5,1,1}$$

$$p_{3,1} + h_{10} + s_{4,2,2}$$

$$s_{4,2,2} p_{3,1} + e_{1,1,1} + h_{10}$$

```
> partToSF(p[4,3,2,2,1]+h[1,1]);
> partToSF(s[4,2,1,1]+e[2]*e[1]+e[2,1]);
> partToSF(h[10]+s[4, 2, 2]*p[3, 1]);
```

$$p1\ p3\ p2^2 p4 + h1^2$$
$$2\ e2\ e1 + s_{4,2,1,1}$$
$$s_{4,2,2} p3\ p1 + h10$$

A.3 Changements de base

A.3.1 Tos

Converti une fonction symétrique homogène dans la base des Schur. La fonction de départ peut contenir des Schur gauches, et être écrite dans n'importe laquelle des deux conventions.

```
> Tos:=proc(f) local res;
>    res:=f;
>    if nops(res)=2 and op(0,res)=‘^‘ then
>       Tos(op(1,res))^op(2,res);
>    elif op(0,res)=‘+‘ or op(0,res)=‘*‘ then
>       convert([seq(Tos(k),k in op(res))],op(0,res));
>    elif res=s[] then return 1;
>    elif type(res,indexed) and op(0,res)=s then
>       if type(op(1,res),sequential) and type(op(2,res),sequential) then
>           res:=tos(skew(s[op(op(2,res))],s[op(op(1,res))]));
>       else tos(res);
>       fi;
>    elif type(res,indexed) and member(op(0,res),‘SF/Bases‘) then
>       tos(partToSF(res));
>    else tos(res);
>    fi;
> end:
```

A.3.2 Top, Toh, Toe

Converti une fonction symétrique homogène dans la base des p. La fonction de départ peut contenir des Schur gauches, et être écrite dans n'importe laquelle des deux conventions. Le résultat obtenu est écrit avec des éléments indicés

```
> Top:=proc(f) local res;
>    res:=Tos(f);
>    res:=top(res);
>    res:=SFtoPart(res);
> end:
```

Converti une fonction symétrique homogène dans la base des h. La fonction de départ peut contenir des Schur gauches, et être écrite dans n'importe laquelle des deux conventions. Le résultat obtenu est écrit avec des éléments indicés

```
> Toh:=proc(f) local res;
>    res:=Tos(f);
>    res:=toh(res);
>    res:=SFtoPart(res);
> end:
```

Converti une fonction symétrique homogène dans la base des e. La fonction de départ peut contenir des Schur gauches, et être écrite dans n'importe laquelle des deux conventions. Le résultat obtenu est écrit avec des éléments indicés

```
> Toe:=proc(f) local res;
>    res:=Tos(f);
>    res:=toe(res);
>    res:=SFtoPart(res);
> end:
```

A.3.3 Exemples

```
> toh(e2);Toh(e2);
```

$$h1^2 - h2$$

$$h_{1,1} - h_2$$

```
> Tom(h3+p2*p1);
> Tom(h3+p[2,1]);
```

$$2\,m_3 + 2\,m_{2,1} + m_{1,1,1}$$

$$2\,m_3 + 2\,m_{2,1} + m_{1,1,1}$$

```
> s[[5,4,2],[3,1,1]]:=Tos(s[[5,4,2],[3,1,1]]);
> s[[5,4,2],[3,1,1]]:=Top(s[[5,4,2],[3,1,1]]);
```

$$s_{[5,4,2],[3,1,1]} := s_{4,2} + s_{4,1,1} + s_{3,3} + s_{3,2,1}$$

$$s_{[5,4,2],[3,1,1]} := 1/6\,p_6 - 1/4\,p_{4,1,1} + 1/18\,p_{3,3} - 1/9\,p_{3,1,1,1} - 1/24\,p_{2,2,2} + 1/8\,p_{2,1,1,1,1} + 1/18\,p_{1,1,1,1,1,1}$$

A.4 Calcul plethystique

A.4.1 Produit d'alphabets

f[XY]

Evaluation plethystique de f sur le produit de 2 alphabets. f est une fonction symétrique écrite en terme de p, h, e, m, s ou Schur gauche Il est possible de donner en paramètre les noms des alphabets. Si rien n'est précisé les noms par défaut sont X et Y

```
> prodsf2alph:=proc(f,b1,b2) local d,s,res,X,Y;
>    if nargs=5 then X:=args[4]; Y:=args[5]; fi;
>    res:=Top(f); res:=partToSF(res);
>    d:=stdeg(res);
>    res:=prodsf2(res,b1,b2,X,Y);
> end:
```

```
> prodsf2:=proc(f,b1,b2) local d,s,res,X,Y;
>    if nargs=5 then X:=args[4]; Y:=args[5]; fi;
>    d:=stdeg(f); res:=f;
>    s:=seq(cat('p',k)=cat('p',k)*cat('q',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b1),res,p);
>    res:=alph(res,b1,d,X);
>    s:=seq(cat('q',k)=cat('p',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b2),res,p);
>    res:=alph(res,b2,d,Y);
>    if b1=b2 then
>       res:=correctAlph(res,b1,d,X,Y);
>    fi;
>    res;
> end:
```

Permet d'ajouter l'alphabet symb dans f sur les termes écrits dans la base b. d est le degré de f

```
> alph:=proc(f,b,d,symb) local s;
>    if member(b,`SF/Bases`) then
>       s:=seq(cat(b,k)=cat(b,k)(symb),k=1..d);
>       subs(s,f);
>    elif member(b[],`SF/Bases`) then
>       s:=seq(seq(b[op(k)]=b[op(k)](symb),k in Par(l)),l=1..d);
>       subs(s,f);
>    else return f;
>    fi;
> end:
```

Permet de corriger un résultat écrit avec des alphabets pour s'assurer qu'il n'y a pas de termes auxquels sont affectés deux alphabets

```
> correctAlph:=proc(f,b,d,symb1,symb2) local s,res;
>    if member(b,`SF/Bases`) then
>       s:=seq(cat('b',k)(symb2)(symb1)=cat('b',k)(symb1),k=1..d);
>       res:=subs(s,f);
>    elif member(b[],`SF/Bases`) then
>       s:=seq(seq(b[op(k)](symb2)(symb1)=b[op(k)](symb1),k in Par(l)),l=1..d);
>       res:=subs(s,f);
>    fi;
>    res:=subs([s[]=1,m[]=1],res);
>    res;
> end:
```

f[XYZ]

Evaluation plethystique de f sur le produit de 3 alphabets. f est une fonction symétrique écrite en terme de p, h, e, m, s ou Schur gauche. Il est possible de donner en paramètre le noms des alphabets. Si rien n'est précisé les noms par défaut sont X et Y

```
> prodsf3alph:=proc(f,b1,b2,b3) local d,s,res;
>    res:=Top(f); res:=partToSF(res);
>    d:=stdeg(res);
>    s:=seq(cat('p',k)=cat('p',k)*cat('q',k)*cat('r',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b1),res,p);
>    res:=alph(res,b1,d,X);
>    s:=seq(cat('q',k)=cat('p',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b2),res,p);
>    res:=alph(res,b2,d,Y);
>    s:=seq(cat('r',k)=cat('p',k),k=1..d);
>    res:=subs(s,res);
>    res:=map(cat(`to`,b3),res,p);
>    res:=alph(res,b3,d,Z);
>    if b2=b3 then
>       res:=correctAlph(res,b2,d,Y,Z);
>    fi;
>    if b1=b3 then
>       res:=correctAlph(res,b1,d,X,Z);
>    fi;
>    if b1=b2 then
>       res:=correctAlph(res,b1,d,X,Y);
>    fi;
>    res;
> end:
```

Exemples

```
# Formule de Cauchy pour les bases h et m
> for i from 2 to 4 do
> prodsf2alph(h[i],h,m);
> od;
```

$$h1X^2m_{1,1}(Y) + h2Xm_2(Y)$$

$$h1(X)^3m_{1,1,1}(Y) + h2(X)h1(X)m_{2,1}(Y) + h3(X)m_3(Y)$$

$$h4(X)m_4(Y) + h3(X)h1(X)m_{3,1}(Y) + h2(X)h1(X)^2m_{2,1,1}(Y) + h2(X)^2m_{2,2}(Y) + h1(X)^4m_{1,1,1,1}(Y)$$

```
# Formule de Cauchy pour les Schur
> for i from 2 to 4 do
> prodsf2alph(h[i],s,s);
> od;
```

$$s_2(X)s_2(Y) + s_{1,1}(X)s_{1,1}(Y)$$

$$s_3(X)s_3(Y) + s_{2,1}(X)s_{2,1}(Y) + s_{1,1,1}(X)s_{1,1,1}(Y)$$

$$s_{1,1,1,1}(X)s_{1,1,1,1}(Y) + s_{2,1,1}(X)s_{2,1,1}(Y) + s_{2,2}(X)s_{2,2}(Y) + s_{3,1}(X)s_{3,1}(Y) + s_4(X)s_4(Y)$$

```
# Evaluation de fonctions Schur
> for i from 2 to 4 do
> cat(`s`,i,1):=prodsf2alph(s[i,1],s,s);
> od;
```

$$s21 := s_3(X)s_{2,1}(Y) + s_{2,1}(X)s_3(Y) + s_{2,1}(X)s_{2,1}(Y) + s_{2,1}(X)s_{1,1,1}(Y) + s_{1,1,1}(X)s_{2,1}(Y)$$

$s31 := s_4(X)s_{3,1}(Y)+s_{3,1}(X)s_{2,1,1}(Y)+s_{3,1}(X)s_{2,2}(Y)+s_{3,1}(X)s_{3,1}(Y)+s_{3,1}(X)s_4(Y)+$
$s_{2,2}(X)s_{2,1,1}(Y)+s_{2,2}(X)s_{3,1}(Y)+s_{2,1,1}(X)s_{1,1,1,1}(Y)+s_{2,1,1}(X)s_{2,1,1}(Y)+s_{2,1,1}(X)s_{2,2}(Y)+$
$s_{2,1,1}(X)s_{3,1}(Y) + s_{1,1,1,1}(X)s_{2,1,1}(Y)$

$s41 := s_{3,1,1}(X)s_{3,2}(Y)+s_{3,1,1}(X)s_{2,2,1}(Y)+s_{3,2}(X)s_{3,2}(Y)+s_{3,2}(X)s_{4,1}(Y)+s_{3,2}(X)s_{3,1,1}(Y)+$
$s_{3,2}(X)s_{2,2,1}(Y)+s_{4,1}(X)s_{3,2}(Y)+s_{1,1,1,1,1}(X)s_{2,1,1,1}(Y)+s_{2,1,1,1}(X)s_{3,1,1}(Y)+s_{2,1,1,1}(X)s_{2,1,1,1}(Y)+$
$s_{2,1,1,1}(X)s_{1,1,1,1,1}(Y)+s_{3,1,1}(X)s_{3,1,1}(Y)+s_{3,1,1}(X)s_{4,1}(Y)+s_{3,1,1}(X)s_{2,1,1,1}(Y)+s_{4,1}(X)s_5(Y)+$
$s_{4,1}(X)s_{3,1,1}(Y)+s_{4,1}(X)s_{4,1}(Y)+s_{2,1,1,1}(X)s_{2,2,1}(Y)+s_{2,2,1}(X)s_{3,1,1}(Y)+s_{2,2,1}(X)s_{3,2}(Y)+$
$s_{2,2,1}(X)s_{2,2,1}(Y) + s_{2,2,1}(X)s_{2,1,1,1}(Y) + s_5(X)s_{4,1}(Y)$

```
> s211:=prodsf2alph(s[2,1,1],s,s);
> s2111:=prodsf2alph(s[2,1,1,1],s,s);
> s32:=prodsf2alph(s[3,2],s,s);
```

$s211 := s_4(X)s_{2,1,1}(Y)+s_{3,1}(X)s_{1,1,1,1}(Y)+s_{3,1}(X)s_{2,1,1}(Y)+s_{3,1}(X)s_{2,2}(Y)+s_{3,1}(X)s_{3,1}(Y)+$
$s_{2,2}(X)s_{2,1,1}(Y) + s_{2,2}(X)s_{3,1}(Y) + s_{2,1,1}(X)s_{2,1,1}(Y) + s_{2,1,1}(X)s_{2,2}(Y) + s_{2,1,1}(X)s_{3,1}(Y) +$
$s_{2,1,1}(X)s_4(Y) + s_{1,1,1,1}(X)s_{3,1}(Y)$

$s2111 := s_{4,1}(X)s_{3,1,1}(Y)+s_{3,2}(X)s_{2,1,1,1}(Y)+s_{3,2}(X)s_{2,2,1}(Y)+s_{3,2}(X)s_{3,2}(Y)+s_{2,2,1}(X)s_{2,2,1}(Y)+$
$s_{2,2,1}(X)s_{3,2}(Y)+s_{2,2,1}(X)s_{4,1}(Y)+s_{2,1,1,1}(X)s_{3,2}(Y)+s_{2,1,1,1}(X)s_{4,1}(Y)+s_{2,1,1,1}(X)s_5(Y)+$
$s_{1,1,1,1,1}(X)s_{4,1}(Y)+s_{3,1,1}(X)s_{3,2}(Y)+s_{3,1,1}(X)s_{4,1}(Y)+s_{2,2,1}(X)s_{3,1,1}(Y)+s_{2,1,1,1}(X)s_{3,1,1}(Y)+$
$s_{3,1,1}(X)s_{3,1,1}(Y)+s_{3,1,1}(X)s_{2,2,1}(Y)+s_{3,1,1}(X)s_{2,1,1,1}(Y)+s_{3,2}(X)s_{3,1,1}(Y)+s_5(X)s_{2,1,1,1}(Y)+$
$s_{4,1}(X)s_{1,1,1,1,1}(Y) + s_{4,1}(X)s_{2,1,1,1}(Y) + s_{4,1}(X)s_{2,2,1}(Y)$

$s32 := s_{3,2}(X)s_5(Y)+s_{3,2}(X)s_{4,1}(Y)+s_{2,1,1,1}(X)s_{2,2,1}(Y)+s_{1,1,1,1,1}(X)s_{2,2,1}(Y)+s_{2,2,1}(X)s_{4,1}(Y)+$
$s_{2,1,1,1}(X)s_{3,2}(Y)+s_{3,2}(X)s_{2,1,1,1}(Y)+s_{2,2,1}(X)s_{1,1,1,1,1}(Y)+s_{2,1,1,1}(X)s_{2,1,1,1}(Y)+s_{3,1,1}(X)s_{3,2}(Y)+$
$s_{3,1,1}(X)s_{2,2,1}(Y)+s_{3,2}(X)s_{3,2}(Y)+s_{3,2}(X)s_{3,1,1}(Y)+s_{3,2}(X)s_{2,2,1}(Y)+s_{2,1,1,1}(X)s_{3,1,1}(Y)+$
$s_{2,2,1}(X)s_{2,1,1,1}(Y)+s_{4,1}(X)s_{2,2,1}(Y)+s_{2,2,1}(X)s_{3,1,1}(Y)+s_{2,2,1}(X)s_{3,2}(Y)+s_{2,2,1}(X)s_{2,2,1}(Y)+$
$2s_{3,1,1}(X)s_{3,1,1}(Y)+s_{3,1,1}(X)s_{4,1}(Y)+s_{3,1,1}(X)s_{2,1,1,1}(Y)+s_{4,1}(X)s_{3,1,1}(Y)+s_5(X)s_{3,2}(Y)+$
$s_{4,1}(X)s_{4,1}(Y) + s_{4,1}(X)s_{3,2}(Y)$

Sur trois alphabets

```
> prodsf3alph(h2,s,s,s);
> prodsf3alph(h3,s,s,s);
```

$s_2(X)s_2(Y)s_2(Z) + s_2(X)s_{1,1}(Y)s_{1,1}(Z) + s_{1,1}(X)s_2(Y)s_{1,1}(Z) + s_{1,1}(X)s_{1,1}(Y)s_2(Z)$

$s_{1,1,1}(X)s_{1,1,1}(Y)s_3(Z)+s_{1,1,1}(X)s_{2,1}(Y)s_{2,1}(Z)+s_{1,1,1}(X)s_3(Y)s_{1,1,1}(Z)+s_{2,1}(X)s_{1,1,1}(Y)s_{2,1}(Z)+$
$s_{2,1}(X)s_{2,1}(Y)s_3(Z)+s_{2,1}(X)s_{2,1}(Y)s_{2,1}(Z)+s_{2,1}(X)s_{2,1}(Y)s_{1,1,1}(Z)+s_{2,1}(X)s_3(Y)s_{2,1}(Z)+$
$s_3(X)s_{1,1,1}(Y)s_{1,1,1}(Z) + s_3(X)s_{2,1}(Y)s_{2,1}(Z) + s_3(X)s_3(Y)s_3(Z)$

```
> prodsf3alph(s[2,1],s,s,s);
> prodsf3alph(s[1,1,1],s,s,s);
```

$s_{1,1,1}(X)s_{1,1,1}(Y)s_{2,1}(Z)+s_{1,1,1}(X)s_{2,1}(Y)s_{1,1,1}(Z)+s_{1,1,1}(X)s_{2,1}(Y)s_{2,1}(Z)+s_{1,1,1}(X)s_{2,1}(Y)s_3(Z)+$
$s_{1,1,1}(X)s_3(Y)s_{2,1}(Z)+s_{2,1}(X)s_{1,1,1}(Y)s_{2,1}(Z)+s_{2,1}(X)s_{1,1,1}(Y)s_3(Z)+s_{2,1}(X)s_{1,1,1}(Y)s_{1,1,1}(Z)+$
$s_{2,1}(X)s_{2,1}(Y)s_3(Z)+3s_{2,1}(X)s_{2,1}(Y)s_{2,1}(Z)+s_{2,1}(X)s_{2,1}(Y)s_{1,1,1}(Z)+s_{2,1}(X)s_3(Y)s_{1,1,1}(Z)+$
$s_{2,1}(X)s_3(Y)s_{2,1}(Z) + s_{2,1}(X)s_3(Y)s_3(Z) + s_3(X)s_{1,1,1}(Y)s_{2,1}(Z) + s_3(X)s_{2,1}(Y)s_{2,1}(Z) +$
$s_3(X)s_{2,1}(Y)s_3(Z) + s_3(X)s_{2,1}(Y)s_{1,1,1}(Z) + s_3(X)s_3(Y)s_{2,1}(Z)$

$s_{1,1,1}(X)s_{1,1,1}(Y)s_{1,1,1}(Z)+s_{1,1,1}(X)s_{2,1}(Y)s_{2,1}(Z)+s_{1,1,1}(X)s_3(Y)s_3(Z)+s_{2,1}(X)s_{1,1,1}(Y)s_{2,1}(Z)+$
$s_{2,1}(X)s_{2,1}(Y)s_3(Z)+s_{2,1}(X)s_{2,1}(Y)s_{2,1}(Z)+s_{2,1}(X)s_{2,1}(Y)s_{1,1,1}(Z)+s_{2,1}(X)s_3(Y)s_{2,1}(Z)+$
$s_3(X)s_{1,1,1}(Y)s_3(Z) + s_3(X)s_{2,1}(Y)s_{2,1}(Z) + s_3(X)s_3(Y)s_{1,1,1}(Z)$

A.4.2 Somme d'alphabets

f[X+Y]

Evaluation plethystique de f sur la somme de 2 alphabets# f est une fonction symétrique écrite en terme de p, h, e, m, s ou Schur gauche# Il est possible de donner en paramètre le noms des alphabets# Si rien n'est précisé les noms par défaut sont X et Y

```
> addsf2alph:=proc(f,b1,b2) local d,sq,res,X,Y;
>    if nargs=5 then X:=args[4]; Y:=args[5]; fi;
>    res:=Top(f); res:=partToSF(res);
>    d:=stdeg(res);
>    sq:=seq(cat('p',k)=(cat('p',k)+cat('q',k)),k=1..d);
>    res:=subs(sq,res);
>    res:=map(cat(`to`,b1),res,p);
>    res:=alph(res,b1,d,X);
>    sq:=seq(cat('q',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=map(cat(`to`,b2),res,p);
>    res:=alph(res,b2,d,Y);
>    res:=subs(s[]=1,res);
>    if b1=b2 then
>       res:=correctAlph(res,b1,d,X,Y);
>    else res;
>    fi;
> end:
```

f[X+Y+Z]

Evaluation plethystique de f sur la somme de 3 alphabets. f est une fonction symétrique écrite en terme de p, h, e, m, s ou Schur gauche. Il est possible de donner en paramètre le noms des alphabets. Si rien n'est précisé les noms par défaut sont X et Y

```
> addsf3alph:=proc(f,b1,b2,b3) local d,sq,res;
>    res:=Top(f); res:=partToSF(res);
>    d:=stdeg(res);
>    sq:=seq(cat('p',k)=(cat('p',k)+cat('q',k))+cat('r',k),k=1..d);
>    res:=subs(sq,res);
>    res:=map(cat(`to`,b1),res,p);
>    res:=alph(res,b1,d,X);
>    sq:=seq(cat('q',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=map(cat(`to`,b2),res,p);
>    res:=alph(res,b2,d,Y);
>    sq:=seq(cat('r',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=map(cat(`to`,b3),res,p);
>    res:=alph(res,b3,d,Z);
>    res:=subs(s[]=1,res);
>    if b2=b3 then res:=correctAlph(res,b2,d,Y,Z); fi;
>    if b1=b3 then res:=correctAlph(res,b1,d,X,Z); fi;
>    if b1=b2 then res:=correctAlph(res,b1,d,X,Y); fi;
>    res;
> end:
```

Exemples

```
> for i from 2 to 4 do
> addsf2alph(h[i],h,h);
> od;
```

$$h1(Y)h1(X) + h2(X) + h2(Y)$$

$$h2(Y)h1(X) + h1(Y)h2(X) + h3(X) + h3(Y)$$

$$h3(Y)h1(X) + h1(Y)h3(X) + h2(Y)h2(X) + h4(X) + h4(Y)$$

```
> for i from 2 to 4 do
> addsf2alph(e[i],e,e);
> od;
```

$$e1(Y)e1(X) + e2(X) + e2(Y)$$

$$e2(Y)e1(X) + e1(Y)e2(X) + e3(X) + e3(Y)$$

$$e4(Y) + e1(Y)e3(X) + e2(Y)e2(X) + e3(Y)e1(X) + e4(X)$$

```
> for i in Par(5) do
> cat(`s`,op(i)):=addsf2alph(s[op(i)],s,s);
> od;
```

$$s5 := s_5(Y) + s_1(X)s_4(Y) + s_2(X)s_3(Y) + s_3(X)s_2(Y) + s_4(X)s_1(Y) + s_5(X)$$

$$s41 := s_{1,1}(X)s_3(Y) + s_4(X)s_1(Y) + s_{4,1}(X) + s_1(X)s_{3,1}(Y) + s_{4,1}(Y) + s_1(X)s_4(Y) + s_2(X)s_{2,1}(Y) + s_2(X)s_3(Y) + s_{2,1}(X)s_2(Y) + s_3(X)s_{1,1}(Y) + s_3(X)s_2(Y) + s_{3,1}(X)s_1(Y)$$

$$s32 := s_2(X)s_{2,1}(Y) + s_{3,2}(X) + s_{3,2}(Y) + s_1(X)s_{3,1}(Y) + s_1(X)s_{2,2}(Y) + s_2(X)s_3(Y) + s_{2,2}(X)s_1(Y) + s_{3,1}(X)s_1(Y) + s_3(X)s_2(Y) + s_{2,1}(X)s_{1,1}(Y) + s_{2,1}(X)s_2(Y) + s_{1,1}(X)s_{2,1}(Y)$$

$$s311 := s_1(X)s_{2,1,1}(Y) + s_1(X)s_{3,1}(Y) + s_{1,1}(X)s_{2,1}(Y) + s_{1,1}(X)s_3(Y) + s_{1,1,1}(X)s_2(Y) + s_2(X)s_{1,1,1}(Y) + s_{3,1,1}(Y) + s_{3,1}(X)s_1(Y) + s_{3,1,1}(X) + s_3(X)s_{1,1}(Y) + s_2(X)s_{2,1}(Y) + s_{2,1}(X)s_{1,1}(Y) + s_{2,1}(X)s_2(Y) + s_{2,1,1}(X)s_1(Y)$$

$$s221 := s_{2,1}(X)s_{1,1}(Y) + s_{2,1}(X)s_2(Y) + s_{1,1,1}(X)s_{1,1}(Y) + s_{2,2}(X)s_1(Y) + s_{2,1,1}(X)s_1(Y) + s_{2,2,1}(X) + s_1(X)s_{2,2}(Y) + s_{2,2,1}(Y) + s_1(X)s_{2,1,1}(Y) + s_2(X)s_{2,1}(Y) + s_{1,1}(X)s_{1,1,1}(Y) + s_{1,1}(X)s_{2,1}(Y)$$

$$s2111 := s_{2,1,1,1}(Y) + s_{2,1,1,1}(X) + s_1(X)s_{1,1,1,1}(Y) + s_1(X)s_{2,1,1}(Y) + s_{1,1}(X)s_{1,1,1}(Y) + s_{1,1}(X)s_{2,1}(Y) + s_{1,1,1}(X)s_{1,1}(Y) + s_{1,1,1}(X)s_2(Y) + s_{1,1,1,1}(X)s_1(Y) + s_2(X)s_{1,1,1}(Y) + s_{2,1}(X)s_{1,1}(Y) + s_{2,1,1}(X)s_1(Y)$$

$$s11111 := s_{1,1,1,1,1}(Y) + s_1(X)s_{1,1,1,1}(Y) + s_{1,1,1,1}(X)s_1(Y) + s_{1,1,1}(X)s_{1,1}(Y) + s_{1,1}(X)s_{1,1,1}(Y) + s_{1,1,1,1,1}(X)$$

```
> expand(addsf2alph(s[[3,2],[1,1]],h,h));
> expand(addsf2alph(s[[3,2],[1,1]],s,s));
```

$$h1(X)h2(X) - h3(X) + h1(Y)h2(Y) - h3(Y) + h1(Y)h1(X)^2 + h1(Y)^2h1(X)$$

$$s_{2,1}(X) + s_{2,1}(Y) + s_2(X)s_1(Y) + s_{1,1}(X)s_1(Y) + s_1(X)s_2(Y) + s_1(X)s_{1,1}(Y)$$

```
> addsf3alph(s[2,2],s,s,s);
```

$s_{2,2}(Z)+s_1(X)s_1(Y)s_{1,1}(Z)+s_1(X)s_1(Y)s_2(Z)+s_1(X)s_{2,1}(Z)+s_{2,1}(Y)s_1(Z)+s_{2,1}(X)s_1(Z)+$
$s_{1,1}(Y)s_{1,1}(Z) + s_{1,1}(X)s_{1,1}(Z) + s_{1,1}(X)s_1(Y)s_1(Z) + s_1(X)s_{1,1}(Y)s_1(Z) + s_2(Y)s_2(Z) +$
$s_2(X)s_2(Z)+s_1(X)s_2(Y)s_1(Z)+s_2(X)s_1(Y)s_1(Z)+s_1(Y)s_{2,1}(Z)+s_{2,2}(Y)+s_{2,1}(X)s_1(Y)+$
$s_{2,2}(X) + s_1(X)s_{2,1}(Y) + s_2(X)s_2(Y) + s_{1,1}(X)s_{1,1}(Y)$

A.4.3 f[XY+XZ+YZ]

f[XY+XZ+YZ]

Evaluation plethystique de f sur la composition XY+XZ+YZ de 3 alphabets. f est une fonction symétrique écrite en terme de p, h, e, m, s ou Schur gauche

```
> add_prodsf:=proc(f,b1,b2,b3) local d,res,sq;
>    res:=Top(f); res:=partToSF(res);
>    d:=stdeg(res);
>    sq:=seq(cat('p',k)=cat('p',k)+cat('qa',k)+cat('qb',k),k=1..d);
>    res:=subs(sq,res);
>    res:=prodsf2(res,b1,b2,X,Y);
>    sq:=seq(cat('qa',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=prodsf2(res,b1,b3,X,Z);
>    sq:=seq(cat('qb',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=prodsf2(res,b2,b3,Y,Z);
>    if b1=b2 then
>       res:=correctAlph(res,b2,d,X,Y);
>       res:=correctAlph(res,b1,d,Y,X);
>    fi;
>    res:=correctAlph(res,b1,d,X,X);
>    if b2=b3 then
>       res:=correctAlph(res,b2,d,Z,Y);
>       res:=correctAlph(res,b3,d,Y,Z);
>    fi;
>    res:=subs(s[]=1,res);
>    res:=subs(m[]=1,res);
> end:
```

Effectue la même chose que la fonction précédente mais dans le cas où les trois bases spécifiées sont égales

```
> add_prodsf1:=proc(f,b) local d,res,sq;
>    res:=Top(f); res:=partToSF(res); d:=stdeg(res);
>    sq:=seq(cat('p',k)=cat('p',k)+cat('qa',k)+cat('qb',k),k=1..d);
>    res:=subs(sq,res);
>    res:=prodsf2(res,b,b,X,Y);
>    sq:=seq(cat('qa',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=prodsf2(res,b,b,X,Z);
>    sq:=seq(cat('qb',k)=cat('p',k),k=1..d);
>    res:=subs(sq,res);
>    res:=prodsf2(res,b,b,Y,Z);
>    res:=correctAlph(res,b,d,X,Y); res:=correctAlph(res,b,d,Y,X);
>    res:=correctAlph(res,b,d,X,X);
>    res:=correctAlph(res,b,d,Z,Y); res:=correctAlph(res,b,d,Y,Z);
>    res:=subs(s[]=1,res);
> end:
```

Exemples

```
> add_prodsf(h1,h,h,h);
> add_prodsf1(h1,s);
```

$$h1(X)h1(Y) + h1(X)h1(Z) + h1(Y)h1(Z)$$
$$s_1(X)s_1(Y) + s_1(X)s_1(Z) + s_1(Y)s_1(Z)$$

```
> add_prodsf1(s[2,1],s);
```

$s_{21}(X)s_{111}(Z) + s_{111}(Y)s_{21}(Z) + s_3(Y)s_{21}(Z) + 2s_1(X)^2s_1(Y)^2s_1(Z)^2 + s_{21}(Y)s_3(Z) + s_{21}(X)s_{21}(Z)+s_1(X)s_1(Z)s_2(Y)s_{11}(Z)+s_1(X)s_1(Z)s_{11}(Y)s_{11}(Z)+s_1(X)s_1(Z)s_{11}(Y)s_2(Z)+ s_1(X)s_1(Z)s_2(Y)s_2(Z) + s_3(X)s_{21}(Z) + s_{111}(X)s_{21}(Z) + s_{111}(X)s_{21}(Y) + s_3(X)s_{21}(Y) + s_{21}(X)s_{111}(Y) + s_{21}(X)s_{21}(Y) + s_2(X)s_2(Z)s_1(Y)s_1(Z) + s_1(X)s_1(Y)s_2(Y)s_2(Z) + s_1(X)s_1(Y)s_2(Y)s_{11}(Z) + s_1(X)s_1(Y)s_{11}(Y)s_{11}(Z) + s_1(X)s_1(Y)s_{11}(Y)s_2(Z) + s_2(X)s_2(Y)s_1(Y)s_1(Z)+s_2(X)s_{11}(Y)s_1(Y)s_1(Z)+s_{11}(X)s_2(Y)s_1(Y)s_1(Z)+s_{21}(X)s_3(Y)+ s_2(X)s_{11}(Z)s_1(Y)s_1(Z) + s_{11}(X)s_{11}(Z)s_1(Y)s_1(Z) + s_{21}(Y)s_{21}(Z) + s_{21}(Y)s_{111}(Z) + s_{11}(X)s_{11}(Y)s_1(Y)s_1(Z) + s_2(X)s_2(Y)s_1(X)s_1(Z) + s_2(X)s_{11}(Y)s_1(X)s_1(Z) + s_{11}(X)s_2(Y)s_1(X)s_1(Z) + s_{11}(X)s_{11}(Y)s_1(X)s_1(Z) + s_{11}(X)s_2(Z)s_1(Y)s_1(Z) + s_1(X)s_1(Y)s_2(X)s_2(Z)+s_1(X)s_1(Y)s_{11}(X)s_2(Z)+s_{21}(X)s_3(Z)+s_1(X)s_1(Y)s_2(X)s_{11}(Z)+ s_1(X)s_1(Y)s_{11}(X)s_{11}(Z)$

```
> add_prodsf1(s[3,1],s);
```

...

Certains résultats trop longs ont volontairement enlevés de ce rapport.

A.4.4 Evaluation générale

Cette fonction prend en entrée des sommes et produits de plethysme fi[X+Y] ou fi[X*Y] où X et Y sont deux alphabets dont le nom peut varier et dont on ne précise ni la taille ni les variables et les fi sont des fonctions symétriques écrites en termes de p, h, e, s et Schur gauche. Deux bases b1 et b2 doivent aussi être précisées. La fonction suivante donne alors l'évaluation plethystique correspondante exprimée dans la base b1# sur l'alphabet X et dans la base b2 sur l'alphabet Y.

```
> evalpl:=proc(sf) local b,f,a;
>    if nargs>1 then b:=[args]; b:=op(subs(b[1]=NULL,b));
>    else ERROR("Specify the bases"); fi;
>    if op(0,sf)=‘+‘ or op(0,sf)=‘*‘ then
>       convert([seq(evalpl(i,b),i in op(sf))],op(0,sf));
>    elif op(0,sf)=Integer or op(0,sf)=Fraction then
>        return sf;
>    elif op(0,sf)=‘^‘ then
>        evalpl(op(1,sf),b)^op(2,sf);
>    else
>       a:=op(sf); f:=op(0,sf);
>    if op(0,a)=‘+‘ then addsf2alph(f,b,op(a));
>    elif op(0,a)=‘*‘ then prodsf2alph(f,b,op(a));
>    else return sf; fi;
>    fi:
> end:
```

A.4.5 Spécialisation d'alphabet

Supprime les termes nuls dans une expression plethystique en fonction de la taille des alphabets

```
> specialise:=proc(f) local a;
>    a:=op({args} minus {f});
>    if op(0,f)=‘+‘ or op(0,f)=‘*‘ then
>       convert([seq(specialise(i,a),i in op(f))],op(0,f));
>    elif op(0,f)=Integer or op(0,f)=Fraction then return f;
>    elif nops(f)=1 then speElmt(f,a);
>    else return f; fi;
> end:

> speElmt:=proc(elmt) local a,alph,x,res;
>    a:={args} minus {elmt};
>    x:=op(op(0,elmt));
>    res:=elmt;
>    for alph in a do
>       if op(elmt)=op(1,alph) then
>          if op(0,op(0,elmt))=s and nops([x])>op(2,alph) then
>             res:=0;
>          elif op(0,op(0,elmt))=e and op(1,[x])>op(2,alph) then
>             res:=0;
>          fi;
>       fi;
>    od;
>    res;
> end:
```

A.4.6 Exemples

Evaluation générale

```
> expand(evalpl(h4[X*Y]-h4[X+Y],h,h));
```

$-4h_1(X)^2h_2(X)h_2(Y)^2-3h_1(Y)^4h_1(X)^2h_2(X)+2h_1(Y)^4h_3(X)h_1(X)+4h_4(X)h_2(Y)h_1(Y)^2-4h_4(X)h_3(Y)h_1(Y)-3h_1(Y)^2h_1(X)^4h_2(Y)+2h_1(Y)h_1(X)^4h_3(Y)-4h_2(Y)h_2(X)^2h_1(Y)^2+2h_3(Y)h_2(X)^2h_1(Y)+4h_4(Y)h_1(X)^2h_2(X)+2h_3(X)h_1(X)h_2(Y)^2-4h_4(Y)h_1(X)h_3(X)-h_4(X)-7h_1(Y)h_1(X)^2h_2(X)h_3(Y)-7h_1(Y)^2h_3(X)h_1(X)h_2(Y)+10h_1(Y)^2h_1(X)^2h_2(X)h_2(Y)+7h_1(Y)h_3(X)h_1(X)h_3(Y)+3h_2(X)^2h_2(Y)^2-2h_4(Y)h_2(X)^2+4h_4(X)h_4(Y)-h_1(Y)h_3(X)+h_1(X)^4h_1(Y)^4+h_1(X)^4h_2(Y)^2-h_4(Y)h_1(X)^4-h_4(Y)-h_3(Y)h_1(X)-h_2(Y)h_2(X)-h_4(X)h_1(Y)^4+h_2(X)^2h_1(Y)^4-2h_4(X)h_2(Y)^2$

```
> evalpl((s[[4,3,1],[2,2]])[X+Y],s,s);
```

$s_3(X)s_1(Y)+s_{2,2}(X)+3s_{2,1}(X)s_1(Y)+s_{1,1,1}(X)s_1(Y)+s_1(X)s_3(Y)+3s_1(X)s_{2,1}(Y)+s_1(X)s_{1,1,1}(Y)+s_{2,2}(Y)+s_{3,1}(X)+s_{2,1,1}(X)+2s_2(X)s_2(Y)+2s_2(X)s_{1,1}(Y)+2s_{1,1}(X)s_2(Y)+2s_{1,1}(X)s_{1,1}(Y)+s_{3,1}(Y)+s_{2,1,1}(Y)$

```
> evalpl((s[[2,2],[1]])[X*Y]-(e2+m[1,1])[X+Y],s,s);
```

$s_3(X)s_{2,1}(Y) + s_{2,1}(X)s_3(Y) + s_{2,1}(X)s_{2,1}(Y) + s_{2,1}(X)s_{1,1,1}(Y) + s_{1,1,1}(X)s_{2,1}(Y) - 2s_{1,1}(Y) - 2s_1(X)s_1(Y) - 2s_{1,1}(X)$

Spécialisation d'alphabet

```
> specialise(t,[U,4],[V,2]);
```

$s_7(U)s_{5,2}(V)+s_{6,1}(U)s_{6,1}(V)+s_{6,1}(U)s_{5,2}(V)+s_{6,1}(U)s_{4,3}(V)+2s_{5,2}(U)s_{5,2}(V)+s_{5,2}(U)s_{4,3}(V)+s_{5,1,1}(U)s_{4,3}(V)+s_{4,3}(U)s_{4,3}(V)+s_{5,2}(U)s_7(V)+s_{4,2,1}(U)s_{6,1}(V)+s_{5,2}(U)s_{6,1}(V)+s_{5,1,1}(U)s_{6,1}(V)+s_{4,3}(U)s_{6,1}(V) + 2s_{4,2,1}(U)s_{5,2}(V) + s_{3,3,1}(U)s_{5,2}(V) + s_{5,1,1}(U)s_{5,2}(V) + s_{4,3}(U)s_{5,2}(V) + s_{4,1,1,1}(U)s_{5,2}(V)+s_{3,2,2}(U)s_{5,2}(V)+s_{3,3,1}(U)s_{4,3}(V)+s_{3,2,1,1}(U)s_{4,3}(V)+s_{3,2,2}(U)s_{4,3}(V)+2s_{4,2,1}(U)s_{4,3}(V)$

A.5 Schur gauches / skew Schur

A.5.1 Evaluation plethystique et Schur positivité

Calcule l'évaluation plethystique de s_mu[X+Y] en fonctions Schur et fonctions Schur gauches

```
> addSkew:=proc(mu) local sq,res;
>    res:=add(s[op(nu)](X)s[[op(mu)],[op(nu)]](Y),nu in subPar(mu));
>    sq:=seq(s[[op(nu)],[op(nu)]](Y)=1,nu in subPar(mu)),s[](X)=1,s[](Y)=1;
>    sq:=sq,seq(s[[op(nu)],[]](Y)=s[op(nu)](Y),nu in subPar(mu));
>    subs(sq,res);
> end:
```

Vérifie la Schur positivité d'une expression

```
> spositive:=proc(s1) local res,l,i,pos;
>    pos:=true;
>    res:=Tos(s1);
>    res:=subs(s[]=1,res);
>    l:=op(res);
>    for i in l do
>       if member(-1,[op(i)]) then pos:=false; fi;
>    od;
>    pos;
> end:
```

A.5.2 Décomposition

Décompose une fonction Schur gauche en deux Schur gauches

```
> decompose:=proc(skew) local mu,nu,imu,inu,res,i,j,snu,smu;
>    mu:=op(1,skew); imu:=nops(mu);
>    nu:=op(2,skew); inu:=nops(nu);
>    if not imu>inu then
>       ERROR("Skew Schur non disconnected or non compact"); fi;
>    i:=0;
>    for j from 1 by 1 to inu while i=0 do
>       if nu[j]=mu[j+1] then i:=j; fi;
>    od;
>    if i=0 or (inu>i and mu[i+1]-nu[i+1]=0) then
>       ERROR("Skew Schur non disconnected or non compact");
>    else
>       # Construction de la concatenation
>       smu:=seq(j=mu[j]-1,j=1..i);
>       snu:=seq(j=nu[j]-1,j=1..i);
>       res:=s[subsop(smu,mu),subsop(snu,nu)];
>       # Construction de la quasi-concatenation
>       smu:=seq(j=mu[j+1],j=i+1..imu-1),imu=NULL;
>       snu:=seq(j=nu[j+1],j=i..inu-1),inu=NULL;
>       res:=res+s[subsop(smu,mu),subsop(snu,nu)];
>    fi;
> end:
```

A.5.3 Exemples

Schur positivité

```
> Tos(s[[5,4,3,2,1],[2,2,1]]-s[[4,4,3,3,1],[2,2,1]]): subs(s[]=1,%);
> spositive(s[[5,4,3,2,1],[2,2,1]]-s[[4,4,3,3,1],[2,2,1]]);
```

$s_{5,2,1,1,1} + s_{3,3,3,1} + 2s_{5,2,2,1} + s_{4,4,1,1} + 2s_{5,3,1,1} + 2s_{4,4,2} + 3s_{4,3,2,1} + 2s_{4,3,3} + s_{4,2,2,2} + s_{4,3,1,1,1} + 2s_{5,3,2} + s_{3,3,2,1,1} + s_{4,2,2,1,1} + s_{5,4,1}$

$$true$$

```
> Tos(s[[5,4,4,2],[3,2]]-s[[4,4,3,3,1],[2,2,1]]): subs(s[]=1,%);
> spositive(s[[5,4,4,2],[3,2]]-s[[4,4,3,3,1],[2,2,1]]);
```

$s_{5,4,1} + s_{5,3,2} + s_{5,3,1,1} + s_{5,2,2,1} + 2s_{4,4,2} + s_{4,3,3} - s_{4,3,1,1,1} - s_{4,2,2,1,1} - s_{3,3,3,1} - 2s_{3,3,2,2} - s_{3,3,2,1,1} - s_{3,2,2,2,1}$

$$false$$

Evaluation plethystique

```
> addSkew([3]);
> addSkew([2,1]);
```

$$s_3(X) + s_2(X)s_{[3],[2]}(Y) + s_1(X)s_{[3],[1]}(Y) + s_3(Y)$$

$$s_{2,1}(X) + s_2(X)s_{[2,1],[2]}(Y) + s_{1,1}(X)s_{[2,1],[1,1]}(Y) + s_1(X)s_{[2,1],[1]}(Y) + s_{2,1}(Y)$$

Décomposition

```
> Tos(s[[5,3,3,2,1,1],[2,2,2]]);
> d1:=decompose(s[[5,3,3,2,1,1],[2,2,2]]);
> Tos(d1);
```

$s_{5,2,2} + s_{5,2,1,1} + s_{5,1,1,1,1} + s_{4,3,2} + s_{4,3,1,1} + 2s_{4,2,2,1} + 2s_{4,2,1,1,1} + s_{4,1,1,1,1,1} + s_{3,3,2,1} + s_{3,3,1,1,1} + s_{3,2,2,2} + s_{3,2,2,1,1} + s_{3,2,1,1,1,1}$

$$d1 := s_{[4,2,2,2,1,1],[1,1,1]} + s_{[5,3,3,1,1],[2,2]}$$

$s_{5,2,2}+s_{5,2,1,1}+s_{5,1,1,1,1}+s_{4,3,2}+s_{4,3,1,1}+2s_{4,2,2,1}+2s_{4,2,1,1,1}+s_{4,1,1,1,1,1}+s_{3,3,2,1}+s_{3,3,1,1,1}+s_{3,2,2,2}+s_{3,2,2,1,1}+s_{3,2,1,1,1,1}$

```
> Tos(s[[5,4,3,2,1],[4,3,2,1]]);
> d2:=decompose(s[[5,4,3,2,1],[4,3,2,1]]);
> Tos(d2);
```

$$s_5 + 4\,s_{4,1} + 5\,s_{3,2} + 6\,s_{3,1,1} + 5\,s_{2,2,1} + 4\,s_{2,1,1,1} + s_{1,1,1,1,1}$$

$$d2 := s_{[4,4,3,2,1],[3,3,2,1]} + s_{[5,3,2,1],[3,2,1]}$$

$$s_5 + 4\,s_{4,1} + 5\,s_{3,2} + 6\,s_{3,1,1} + 5\,s_{2,2,1} + 4\,s_{2,1,1,1} + s_{1,1,1,1,1}$$

A.6 Autres

A.6.1 Calcul des coefficients de Littlewood-Richardson

Calcul de c_{mu,nu}^lambda, les coefficients de Littlewood-Richardson de forme lambda\mu et de poids nu (mu,nu et lambda sont trois partitions)

```
> coefLR:=proc(lambda,mu,nu) local t,i,k;
>    t:=tos(skew(s[op(mu)],s[op(lambda)]));
>    if op(0,t)='+' then t:=[op(t)];
>    else t:=[t]; fi;
>    for i in t do
>       if op(0,i)='*' and op(op(2,i))=op(nu) then
>          return op(1,i);
>       elif op(i)=op(nu) then
>          return 1;
>       fi;
>    od;
>    return 0;
> end:
```

A.6.2 Affichage des partitions

```
> with(plots): with(plottools):
> carre:=proc(i,j)
>    polygon([[i,j],[i+1,j],[i+1,j+1],[i,j+1]],color=white,linestyle=plain);
> end:

> plot_partition:=proc(lambda) local i,j,p,s;
>    p:=NULL;
>    for i from 1 by 1 to nops(lambda) do
>       for j from 1 by 1 to op(i,lambda) do
>          p:=p,carre(j,i);
>       od;
>    od;
>    s:=20*max(nops(lambda),op(1,lambda)),20*max(nops(lambda),op(1,lambda));
>    display([p],axes=none,scaling=constrained,size=[s]);
> end:
```

```
> plot_partitionGauche:=proc(lambda,mu) local k,i,j,p,s,nu;
>    p:=NULL; nu:=op(mu);
>    for k from 1 by 1 to nops(lambda)-nops(mu) do
>       nu:=nu,0;
>    od;
>    for i from 1 by 1 to nops(lambda) do
>       for j from op(i,[nu])+1 by 1 to op(i,lambda) do
>          p:=p,carre(j,i);
>       od;
>    od;
>    s:=20*max(nops(lambda),op(1,lambda)),20*max(nops(lambda),op(1,lambda));
>    display([p],axes=none,scaling=constrained,size=[s]);
> end:
```

A.6.3 Exemples

Calculs de coefficients de LR

```
> Tos(s[[7,4,3,3,2],[4,2,2,1]]);
```

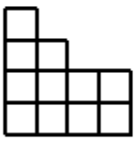
$s_{7,2,1} + s_{7,1,1,1} + 2s_{6,3,1} + 2s_{6,2,2} + 4s_{6,2,1,1} + s_{6,1,1,1,1} + s_{5,4,1} + 3s_{5,3,2} + 5s_{5,3,1,1} + 5s_{5,2,2,1} + 3s_{5,2,1,1,1} + s_{4,4,2} + 2s_{4,4,1,1} + s_{4,3,3} + 5s_{4,3,2,1} + 3s_{4,3,1,1,1} + 2s_{4,2,2,2} + 3s_{4,2,2,1,1} + s_{3,3,3,1} + s_{3,3,2,2} + 2s_{3,3,2,1,1} + s_{3,2,2,2,1}$

```
> coefLR([7,4,3,3,2],[5,3,1,1],[4,2,2,1]);
> coefLR([7,4,3,3,2],[5,3,2],[4,2,2,1]);
> coefLR([7,4,3,3,2],[3,3,3,1],[4,2,2,1]);
> coefLR([7,4,3,3,2],[3,3,3],[4,2,2,1]);
```

5
3
1
0

Affichage de partitions et forme gauches

```
> plot_partition([4,4,2,1]);
```



```
> plot_partitionGauche([4,4,2,1],[2,1]);
```