Extension SYM de MACSYMA
MANUEL DE L'UTILISATEUR

Annick Valibouze
LITP (tour 45-55)
4, Place Jussieu
75252 Paris Cedex 05
Unité associée au CNRS No 248
et
GDR DE CALCUL FORMEL MEDICIS
e-mail : avb@sysal.ibp.fr

19 juillet 1993

INTRODUCTION
Cette documentation concerne un module de manipulation de fonctions symétriques implanté en CommonLisp. Ce module, nommé `SYM`, se présente actuellement comme une extension du système de calcul formel `MACSYMA`.

Parmi ses applications, signalons des calculs de résolvantes.

1 Définitions et notations

Nous considèrerons un anneau $\mathcal{A}$.

1.1 Polynômes symétriques

Nous nous donnons un entier $n > 0$.

Action de groupe

Soit S_n le groupe symétrique de degré n.

Soit E un ensemble quelconque. Notons E^n l'ensemble des n-uplets d'éléments de E. L'action de S_n sur tout n-uplet $\underline{a} = (a_1, \ldots, a_n)$ de E^n est définie comme suit :

$$\begin{array}{lll} S_n \times & E^n & \longrightarrow E^n \\ \sigma \ \times & \underline{a} & \longrightarrow \sigma\underline{a} = (a_{\sigma(1)}, \ldots, a_{\sigma(n)}). \end{array}$$

Soit $\mathcal{A}[\underline{x}]$ l'anneau des polynômes en les n variables de $\underline{x} = (x_1, \ldots, x_n)$ à coefficients dans $\mathcal{A}$. L'action de S_n sur $\mathcal{A}(\underline{x})$ est définie comme suit :

$$\begin{array}{lll} S_n & \times \mathcal{A}[\underline{x}] & \longrightarrow \mathcal{A}[\underline{x}] \\ \sigma & \times f & \longrightarrow \sigma f \ : \ \sigma f(\underline{x}) = f(\sigma\underline{x}). \end{array}$$

Remarque. L'action de S_n s'étend à des ensembles E^c ou $\mathcal{A}[x_1, \ldots, x_c]$ avec $c \leq n$ par simple plongement de ces ensembles dans E^n et $\mathcal{A}[\underline{x}]$ respectivement.

Soit f une fonction de $\mathcal{A}[\underline{x}]$ (resp. $\underline{a}$ un n-uplet). Nous noterons $S_n f$ (resp. $S_n\underline{a}$) l'ensemble $\{\sigma f \mid \sigma \in S_n\}$ (resp. $\{\sigma\underline{a} \mid \sigma \in S_n\}$) appelé l'*orbite* de f (resp. $\underline{a}$) sous l'action de S_n.

Partitions, polynômes symétriques

Soit m un entier naturel. Une *partition* de m, est une suite décroissante d'entiers naturels $i_1 \geq i_2 \geq \ldots$.. appelées *parts* dont la somme est égale à m. Cette somme est le *poids.* de la partition et sa *longueur* est le nombre de ses parts non nulles.

Soit p un polynôme de n variables. Ce polynôme est dit *symétrique* s'il reste invariant sous l'action de S_n, c'est-à-dire si $S_n p = \{p\}$.

Un polynôme symétrique peut-être représenté par des partitions :

Soient $I = (i_1, \ldots, i_n)$ (i.e. $i_1 \geq i_2 \geq \ldots \geq i_n$) une partition de longueur inférieure ou égale à n et $\underline{x}^I$ le monôme

$$\underline{x}^I = x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}.$$

Une *forme monomiale* $M_I(\underline{x})$ sur $\underline{x}$ indexée par I est la somme des monômes de l'orbite de $\underline{x}^I$ sous l'action de S_n :

$$M_I(\underline{x}) = \sum_{J \in S_n I} \underline{x}^J.$$

Les formes monomiales constituent naturellement une base d'espace vectoriel sur l'anneau des polynômes symétriques : tout polynôme symétrique se représente comme une combinaison linéaire finie de formes monomiales.

Représentations en machine

Soit p un polynôme symétrique sur un anneau $\mathcal{A}$. Supposons que p soit donné sur la base des formes monomiales.

Dans la *représentation contractée* de p on remplace toute forme monomiale $M_I(x)$ par x^I, ou par tout monôme de $S_n x^I$, son orbite. Dans sa *représentation partitionnée*, on remplace tout *terme monomial* $cM_I(x)$, où c est un coefficient sur $\mathcal{A}$, par la liste $[c, i_1, \ldots, i_n]$ et on réunit le tout dans une liste.

Exemple. le polynôme contracté associé à $3x^4 + 3y^4 - 2xy^5 - 2x^5y$ est $3x^4 - 2xy^5$ et le polynôme partitionné est [[3,4],[-2,5,1]].

1.2 Polynômes multi-symétriques

Nous généralisons ici le cas symétrique. Nous nous donnons un entier $r > 0$ et un r-uplet d'entiers $D = (d_1, \ldots, d_r)$.

Action de groupe

Soient $E_1, E_2, \ldots, E_r$ r ensembles quelconques. Notons $D(E)$ le produit $E_1^{d_1} \times E_2^{d_2} \times \ldots \times E_r^{d_r}$

L'action du produit de groupes symétriques $S_D = S_{d_1} \times S_{d_2} \times \ldots \times S_{d_r}$ sur $D(E)$ est définie naturellement comme suit :

$$\begin{array}{lll} S_D \times & D(E) & \longrightarrow D(E) \\ \sigma \ \times & A & \longrightarrow \sigma A = (\sigma_1 \underline{a}_1, \ldots, \sigma_r \underline{a}_r), \end{array}$$

où $\sigma = (\sigma_1, \ldots, \sigma_r)$ avec $\sigma_i \in S_{d_i}$. L'orbite de A sous l'action de S_D sera notée $S_D A$.

De même si $X = (\underline{x}_1, \ldots, \underline{x}_r)$ est un r-uplet tel que la i-ième composante $\underline{x}_i$ est un d_i-uplet de variables, on notera $\mathcal{A}[X]$ l'ensemble des polynômes en les variables de X et à coefficients dans $\mathcal{A}$. L'action de S_D sur $\mathcal{A}[X]$ est définie comme suit :

$$\begin{array}{lll} S_D & \times \mathcal{A}[X] & \longrightarrow \mathcal{A}[X] \\ \sigma & \times f & \longrightarrow \sigma f \ : \ \sigma f(X) = f(\sigma X). \end{array}$$

L'orbite de f sous l'action de S_D sera notée $S_D f$.

Multi-partitions, Polynômes multi-symétriques

Une *multi-partition*, I, d'ordre r est un r-uplet de partitions. Nous appellerons la *multi-longueur* de I la liste des longueur des partitions qui constituent I.

Soit p un polynôme de $d_1 + \cdots + d_r$ variables. Ce polynôme est dit *multi-symétrique* s'il reste invariant par S_D, c'est-a-dire si $S_D p = \{p\}$.

Un polynôme multi-symétrique peut-être représenté par des multi-partitions :

Soient X comme ci-dessus et $I = (I_1, \ldots, I_r)$ un r-uplet de $\mathbf{N}^{d_1} \times \ldots \times \mathbf{N}^{d_r}$. on notera naturellement par X^I le monôme

$$X^I = \underline{x}_1^{I_1} \ldots \underline{x}_r^{I_r}.$$

Si I est une multi-partition de multi-longueur inférieure à D, nous définissons la *multi-forme monomiale* $M_I(X)$ sur X indexée par I par la somme des monôme de l'orbite de X^I sous l'action de S_D :

$$M_I(X) = \sum_{J \in S_D I} X^J.$$

Il est naturel de représenter un polynôme multi-symétrique par une combinaison linéaire de multi-forme monomiales sur l'anneau des coefficients du polynôme.

Répresentation machine

Considérons maintenant un polynôme multi-symétrique à coefficients dans un anneau $\mathcal{A}$ en les variables de X. Ce polynôme s'exprime naturellement comme une somme finie de $cM_I(X)$ où $c \in \mathcal{A}$ et $M_I(X)$ est une multi-forme monomiale sur X. Une *forme contractée* associée à un polynôme multi-symétrique consiste à remplacer dans ce polynôme chaque $M_I(X)$ par le monôme X^I ou bien par un autre monôme de son orbite $S_D X^I$.

2 Initialisation et mode opératoire

Le fichier sym.mac comporte toutes les commandes d'auto-chargement. C'est éventuellement dans ce fichier qu'il faut modifier des noms de chemins si l'utilisation de `SYM` doit se faire d'un autre directory que celui où il est installé.

le fichier `compile.lsp` comporte la liste des fichiers à compiler. Pour l'exécuter on peut le charger sous `LISP` ou sous `MACSYMA` par la commande `load(‘‘compile.lsp’’)`. Une fois les fichiers de `SYM` compilés on charge le module sous `MACSYMA` avec le fichier `sym.mac` :

```
load(‘‘sym.mac’’);
```

Sauf en cas de précision, les fonctions appelées complètent les listes avec des valeurs formelles. Pour la ième fonction symétrique élémentaire, ce sera `ei`, pour la ième fonction puissance, ce sera `pi`, et pour la ième fonction complète ce sera `hi`.

Il existe plusieurs modes d'évaluation de polynômes sous MACSYMA : meval, expand, rat, ratsimp. SYM permet le choix du mode opératoire. A chaque appel d'une fonction, SYM teste si le drapeau de la variable `oper` a été modifié. Dans ce cas, le mode opératoire est modifié ainsi :

Si `oper = meval` (par défaut), on fait les opérations avec `meval` .

Si `oper = expand`, elles sont faites avec `expand`.

Si `oper = rat`, elles sont faites avec `rat`.

Si `oper = ratsimp`, elles sont faites avec `ratsimp`.

Le mode `meval` est avantageux en numérique. Avec des valeurs formelles le mode `rat` est souvent préférable.

Pour mettre le drapeau d'`oper` à `expand`, par exemple, il suffit d'écrire :

```
oper : expand;
```

3 Descriptif des fonctions disponibles

Prenons q la valeur minimale entre le degré du polynôme `sym` et le cardinal `card`. On met dans `lvar` les variables du polynôme considéré le distinguant ainsi des paramètres éventuels.

3.1 Combinatoire

- `CARD_ORBIT(partition,n)` $\longrightarrow$ `entier`
  `partition` est une partition donnée sous la forme $[a_1, m_1, ..., a_q, m_q]$ où m_i est la multiplicité de a_i dans la partition. La fonction calcule le cardinal de l'orbite de la partition sous l'action du groupe symétrique de degré `n`.

- `MULTINOMIAL(r,part)` $\longrightarrow$ `entier`
  où `r` est le poids de la partition `part`. Cette fonction ramène le coefficient multinomial associé : si les parts de la partitions `part` sont $i_1, i_2, ..., i_k$, le résultat de `MULTINOMIAL` est $r!/(i_1!i_2!...i_k!)$.

- `CARD_STAB(l,egal)` $\longrightarrow$ `entier`
  `l` est une liste d'objets ordonnés et `egal` est le test d'égalité entre eux. Soit n la longueur de la liste `l`. Cette fonction calcule le cardinal du stabilisateur de `l` sous l'action du groupe symétrique d'ordre n.

- `PERMUT(l)` $\longrightarrow$ liste
  ramène la liste des permutations de la liste `l`.

```
CARD_ORBIT([5,2,1,3],6);
                                   60
CARD_STAB([a, a, c, b, b], eq);
                                    4
CARD_STAB([1,1,2,3,3], "=");
                                    4
```

3.2 Sur les polynômes d'une variable

- `ELE2POLYNOME(cele,z)` $\longrightarrow$ polynôme
  donne le polynôme en `z` dont les fonctions symétriques élémentaires des racines sont dans la liste `cele`. `cele`$=[n, e_1, ..., e_n]$ où n est le degré du polynôme et e_i la i-ième fonction symétrique élémentaire.

- `POLYNOME2ELE(polynôme,z)` $\longrightarrow$ `cele`
  donne la liste `cele`$=[n, e_1, ..., e_n]$ où n est le degré du polynôme de la variable `z` et e_i la i-ième fonction symétrique élémentaire des ses racines.

```
ELE2POLYNOME([2,e1,e2],z);
                                   2
                                  Z  - E1 Z + E2

POLYNOME2ELE(X^7-14*X^5  + 56*X^3  - 56*X + 22,X);

                     [7, 0, - 14, 0, 56, 0, - 56, - 22]

ELE2POLYNOME( [7, 0, - 14, 0, 56, 0, - 56, - 22],X);

                         7       5       3
                        X  - 14 X  + 56 X  - 56 X + 22
```

3.3 Changements de représentations

Un polynôme symétrique peut être donné sous plusieurs formes : éetendue, contractée, partitionnée. Les fonctions décrites ci-dessous permettent de passer d'une forme à une autre. Certaines réalisent de plus un test de symétrie.

Dans tout les cas les variables du polynôme sont contenues dans la liste `lvar`.

- `TPARTPOL(polynôme,lvar)` $\longrightarrow$ `ppart ,`
  `PARTPOL(polynôme, lvar)` $\longrightarrow$ `ppart`
  ramènent, ordonné dans l'ordre lexicographique croissant et décroissant respectivement, le polynôme partitionné associé au polynôme donné sous sa forme étendue. Si le polynôme n'est pas symétrique, la fonction `TPARTPOL` déclenche une erreur.

- `TCONTRACT(polynôme,lvar)` $\longrightarrow$ `pc`
  `CONTRACT(polynôme,lvar)` $\longrightarrow$ `pc`

agissent respectivement comme TPARTPOL etPARTPOL en restituant la représention contractée.

- `CONT2PART(pc,lvar)` $\longrightarrow$ `ppart`
  rend le polynôme partitionné `ppart` d'un polynôme symétrique donné sous une forme contractée `pc`.

- `PART2CONT(ppart,lvar)` $\longrightarrow$ `pc`
  rend une forme contractée `pc` d'un polynôme symétrique donné sous sa forme partitionné `ppart`.

- `EXPLOSE(pc,lvar)` $\longrightarrow$ `psym`
  rend la forme étendue `psym` d'un polynôme symétrique donné sous une forme contractée `pc`.

```
TPARTPOL(x^3-y,[x,y]);

                         manque des monomes
```

Soit le polynôme symétrique de $\mathbf{Z}[x,y,z]$ dont une forme contractée est $2a^3bx^4y$. Nous allons lui imposer divers changements de represéntations.

```
lvar : [x,y,z]$
pc : 2*a^3*b*x^4*y;
                                 3    4
                              2 a  b x  y
psym : EXPLOSE(pc,lvar);
                        3      4      3      4      3    4
                     2 a  b y z  + 2 a  b x z  + 2 a  b y  z
          3    4        3      4      3    4
   + 2 a  b x  z + 2 a  b x y  + 2 a  b x  y
```

Et si on lui applique la fonction CONTRACT on retrouve une forme contractée :

```
CONTRACT(psym,lvar);
                                  3    4
                               2 a  b x  y
TCONTRACT(psym,lvar);
                                  3    4
                               2 a  b x  y
TPARTPOL(psym,lvar);
                                   3
                              [[2 a  b, 4, 1]]
PARTPOL(psym,lvar);
```

```
                                  3
                             [[2 a  b, 4, 1]]
ppart : CONT2PART(pc,lvar);
                                  3
                             [[2 a  b, 4, 1]]
PART2CONT(ppart,lvar);
                                 3    4
                              2 a  b x  y
```

3.4 Les fonctions liées aux partitions

- `KOSTKA(part1,part2)` ⟶ entier
  (écrit par P.ESPERET) ramène le nombre de Kostka associé aux partitions `part1` et `part2`.

- `TREINAT(part)` ⟶ liste des partitions inférieures pour l'ordre naturel à la partition `part` et de même poids.

- `TREILLIS(n)` ⟶ liste des partitions de poids n.

- `LGTREILLIS(n,m)` ⟶ liste des partitions de poids n et de longueur m.

- `LTREILLIS(n,m)` ⟶ liste des partitions de poids n et de longueur inférieure ou égale à m.

```
KOSTKA([3,3,3],[2,2,2,1,1,1]);
                                  6
LGTREILLIS(4,2);
                           [[3, 1], [2, 2]]
LTREILLIS(4,2);
                         [[4, 0], [3, 1], [2, 2]]
TREILLIS(4);
                [[4], [3, 1], [2, 2], [2, 1, 1], [1, 1, 1, 1]]
TREINAT([5]);
                               [[5]]
TREINAT([1,1,1,1,1]);
                        [[5], [4, 1], [3, 2], [3, 1, 1],
                   [2, 2, 1], [2, 1, 1, 1], [1, 1, 1, 1, 1]]
TREINAT([3,2]);
                        [[5], [4, 1], [3, 2]]
```

3.5 Les calculs d'orbites

- ORBIT(polynôme,lvar) $\longrightarrow$ S_n(polynôme)
  ramène la liste des polynômes de l'orbite du polynôme sous l'action du groupe symétrique S_n. Les n variables du polynôme variables sont contenues dans la liste lvar.

- MULTI_ORBIT(polynôme,[lvar$_1$,lvar$_2$,...,lvar$_r$]) $\longrightarrow$ S_D(polynôme)
  les variables de polynôme sont dans les listes de variables lvar$_1$,lvar$_2$,...,lvar$_r$ sur lesquelles on fait agir respectivement les groupes symétriques $S_{d_1}, S_{d_2}, \ldots, S_{d_r}$. Cette fonction ramène l'orbite du polynôme sous l'action du produit S_D de ces groupes symétriques.

```
ORBIT(a*x+b*y,[x,y]);
                              [a y + b x, b y + a x]
ORBIT(2*x+x**2,[x,y,z]);
                                   2         2         2
                                 [z  + 2 z, y  + 2 y, x  + 2 x]
MULTI_ORBIT(a*x+b*y,[[x,y],[a,b]]);
                                   [b y + a x, a y + b x]
MULTI_ORBIT(x+y+2*a,[[x,y],[a,b,c]]);

                  [y + x + 2 c, y + x + 2 b, y + x + 2 a]
```

3.6 Le produit contracté de deux polynômes symétriques

- MULTSYM(ppart1, ppart2,n) $\longrightarrow$ pc
  calcule le produit de deux polynômes symétriques de n variables dont part1 et part2 sont les formes partitionnées associées.

Soient deux polynômes symétriques p1 et p2. On va calculer le produit des polynômes par une méthode classique du produit de deux polynômes quelconques, puis on réalisera ce produit avec MULTSYM. On se place dans $\mathbf{Z}[x, y]$.

```
p1 : x*y^2  + x^2*y$
p2 : y+x$
prod : expand(p1*p2);
                              3      2  2     3
                           x y  + 2 x  y  + x  y
```

Constatons c'est la forme étendue du produit obtenu avec MULTSYM :

```
PARTPOL(prod,[x,y]);
                                    [[1, 3, 1], [2, 2, 2]]
ppart1 : PARTPOL(p1,[x,y]);
                                      [[1, 2, 1]]
ppart2 : PARTPOL(p2,[x,y]);
                                      [[1, 1, 0]]
MULTSYM(ppart1, ppart2, 2);
                                    [[1, 3, 1], [2, 2, 2]]
```

3.7 Les changements de bases

De manière générale, la liste `lvar` represénte la liste des variables des polynômes.

- `ELEM(cele,sym,lvar)` $\longrightarrow$ `P(e1,..., eq)`
  décompose le polynôme symétrique `sym` en les fonctions symétriques élémentaires contenues dans `cele` (3 drapeaux possibles).

- `multi_elem([cele`$_1$`, ..., cele`$_r$`],multi_pc, [lvar`$_1$`, ...,lvar`$_r$`])` $\longrightarrow$ `P(cele`$_1$`, ...,cele`$_r$`)`
  On a un polynôme `multi_pc` multi-symétrique sous l'action de S_D donné par sa forme multi-contractée. On le décompose successivement en chacun des groupes `cele`$_j$ de fonctions symétriques élémentaires de l'ensemble $\underline{x}_j$. La liste de variables `lvar`$_j$ permet de lire le polynôme. On y trouve les variables de $\underline{x}_j$ intervenant dans l'expression du polynôme multi-contracté.

- `PUI(cpui,sym,lvar)` $\longrightarrow$ `P(p1, ...  ,pq)`
  décompose un polynôme symétrique en les fonctions puissances (3 drapeaux possibles).

- `multi_pui([cpui`$_1$`, ..., cpui`$_p$`],multi_pc, [lvar`$_1$`, ...,lvar`$_p$`])` $\longrightarrow$ `P(cpui`$_1$`, ...,cpui`$_p$` )`
  agit comme `MULTI_ELEM` en décomposant le polynôme multi-symétrique sur les fonctions puissances.

Si le polynôme symétrique est sous une forme contractée le drapeau `elem` doit être à 1 (sa valeur par défaut).

Considérons le polynôme symétrique $x^4+y^4+z^4+t^4-2*(x*(y+z+t)+y*(z+t)+z*t)$ dont une forme contractée est $x^4-2*y*z$.

```
elem:1$

elem([],x**4 - 2*y*z, [x,y,z]);
```

```
     4         2                    2
   e1  - 4 e2 e1  + 4 e3 e1 + 2 e2  - 2 e2 - 4 e4
```

Supposons maintenant que le nombre de variables du polynôme symétrique est 3 (i.e. $t = 0$). Cette valeur 3 doit être signalée en tête de la liste `cele` comme suit :

```
elem([3],x**4 - 2*y*z,[x,y,z]);
```

```
     4         2                    2
   e1  - 4 e2 e1  + 4 e3 e1 + 2 e2  - 2 e2
```

Si de plus la première fonction symétrique élementaire a une valeur particulière, $e_1 = 7$, on la met en deuxième élément de la liste `cele` :

```
ELEM([3,7],x^4-2*x*y,[x,y]);
```

```
                  2
   28 e3 + 2 e2  - 198 e2 + 2401
```

Le i+1-ième élément de la liste cele doit être la i-ième fonction symétrique élémentaire.

Si le polynôme symétrique est sous une forme étendue le drapeau `elem` doit être à 2 et s'il est sous une forme partitionnée le drapeau `elem` doit être à 3 :

```
elem :2$
ELEM([3,f1,f2,f3],x^4+y^4+z^4 - 2*(x*y + x*z + y*z),[x,y,z]);

                     4         2                    2
                   f1  - 4 f2 f1  + 4 f3 f1 + 2 f2  - 2 f2

elem:3$
ELEM(([],[[1, 2, 1]],[]);
                            E1 E2 - 3 E3
```

Il en est de même pour le drapeau `pui` associé à la fonction `PUI`.

Pour la fonction `PUI`, si des valeurs formelles doivent être rajoutées à `cpui` on tient compte du cardinal de l'alphabet, s'il est fournit, pour calculer les fonctions puissances en fonction des premières. On se sert alors de la fonction `PUIREDUC` (voir plus loin).

```
MULTI_ELEM([[2,e1,e2],[2,f1,f2]],a*x+a^2+x^3,[[x,y],[a,b]]);

                              2                          3
                - 2 f2 + f1   + e1 f1 - 3 e1 e2 + e1

MULTI_PUI([[2,p1,p2],[2,t1,t2]],a*x+a^2+x^3,[[x,y],[a,b]]);

                                                3
                                      3 P1 P2   P1
                         T2 + P1 T1 + ------- - ---
                                         2       2
```

- `ELE2PUI(m,cele)` $\longrightarrow$ `cpui`
  réalise le passage des fonctions symétriques élémentaires aux fonctions puissances de 1 à m.

- `PUI2ELE(n,cpui)` $\longrightarrow$ `cele`
  réalise le passage des fonctions puissances aux fonctions symétriques élémentaires. Si le drapeau `pui2ele` est égal à `girard`, on récupère la liste des fonctions symétriques élémentaires de 1 à `n`, et s'il est égal à `close`, on récupère la `n`^{ième} fonction symétrique élémentaire.

Ci-dessous, on cherche les 3 premières fonctions symétriques élémentaires. On ne donne pas de valeur aux 3 premières fonctions puissances. La fonction rajoute donc des variables formelles.

```
PUI2ELE(3,[]);

                   2                         3
                 p1     p2  p3    p1 p2    p1
        [3, p1, --- - --, -- - ----- + ---]
                 2      2   3      2       6
```

Ci-dessous on cherche les 4 premières fonctions symétriques élémentaires en fonctions des fonctions puissances telles que $p_1 = 2$. Le cardinal de l'alphabet étant 3, la quatrième fonction symétrique élémentaire est donc nulles. Ensuite la fonction `ELE2PUI` calcule les 3 premiìres fonctions pussance en fonctions des 3 premiìres fonctions symétriques élémentaires.

```
PUI2ELE(4,[3,2]);
                   4 - p2  p3 - 3 p2 + 4
        [3, 2 , ------, -------------, 0]
```

```
                               2         3
ELE2PUI(3,[]);
                                 2                             3
                  [3, e1, e1  - 2 e2, 3 e3 - 3 e1 e2 + e1 ]
```

Ici, comme le cardinal est 2, la troisième fonction symétrique élémentaire est nulle :

```
ELE2PUI(3,[2]);
                                    2            3
                     [2, e1, e1  - 2 e2, e1  - 3 e1 e2]
```

- `PUIREDUC(n,cpui)` $\longrightarrow$ `[card,`$p_1, p_2, p_3, ..., p_n$`]`
  permet d'avoir les fonctions puissances jusqu'à `n` connaissant celles jusqu'à `m`. Le cardinal est précisé dans `cpui`.

Si le cardinal, n, de l'alphabet est donné, on peut exprimer toutes les fonctions puissances en fonction des n premières. On prend par exemple 2 pour cardinal et on demande les 3 premières fonctions puissances.

```
PUIREDUC(3,[2]);
                                        3
                            3 p1 p2   p1
               [2, p1, p2, ------- - ---]
                              2        2
```

- `ELE2COMP(m , cele)` $\longrightarrow$ `ccomp`
  réalise le passage des fonctions symétriques élémentaires aux fonctions symétriques complètes de 1 à m.
- `PUI2COMP(n, cpui)` $\longrightarrow$ `ccomp`
  réalise le passage des fonctions puissances aux fonctions symétriques complètes de 1 à n.
- `COMP2ELE(n, ccomp)` $\longrightarrow$ `cele`
  réalise le passage des fonctions symétriques complètes aux fonctions symétriques élémentaires de 1 à n.
- `COMP2PUI(n, ccomp)` $\longrightarrow$ `cpui`
  réalise le passage des fonctions symétriques complètes aux fonctions puissances de 1 à n.

- `MON2SCHUR(liste)` $\longrightarrow$ `pc`
  réalise le passage des formes monomiales aux fonctions de Schur. Le résultat `pc` est donc un polynôme symétrique donné sous une forme contractée.

- `SCHUR2COMP(P,[h`i_1`,...,h`i_q`]))` $\longrightarrow$ `liste de listes`
  réalise le passage des fonctions de Schur, notées S_I, aux fonctions totales. Le polynôme P est un polynôme en les fonctions totales hi_k. Il est indispensable de noter ces fonctions totales avec un h concaténé à un entier.

La fonction `MON2SCHUR` permet d'écrire une fonction de Schur sur la base des formes monomiales représentées sous leur forme contractée. La fonction de Schur est donnée par une partition. Nous allons d'abord vérifier que la fonction de Schur associé à la partition (1^3) est égale à la $3^{ième}$ fonction symétrique élémentaire et que celle associée à la partition (3) est égale à la $3^{ième}$ fonction symétrique complète (ceci découle d'un résultat général).

```
MON2SCHUR([1,1,1]);
                      x1 x2 x3
MON2SCHUR([3]);
                          2        3
             x1 x2 x3 + x1  x2 + x1

MON2SCHUR([1,2]);
                                  2
                   2 x1 x2 x3 + x1  x2
```

Voyons sur un exemple comment, en opérant circulairement sur des changements de bases, on recupére bien sur la donnée initiale.

```
a1 :  PUI2COMP(3,[3]);
                                2                  3
                        p2    p1    p3    p1 p2    p1
                [3, p1, -- + ---, -- + ----- + ---]
                        2     2     3       2      6
a2 : COMP2ELE(3, a1);
                      2                        3
                    p1     p2  p3    p1 p2    p1
               [3, p1, --- - --, -- - ----- + ---]
                     2     2   3       2      6
a3 : ELE2PUI(3,a2);
                      [3, p1, p2, p3]
```

```
a4 : COMP2PUI(3,[]);
                                   2                          3
            [3, h1, 2 h2 - h1 , 3 h3 - 3 h1 h2 + h1 ]

a5 : PUI2ELE(3,a4);
                              2                          3
              [3, h1, h1  - h2, h3 - 2 h1 h2 + h1 ]

a6 : ELE2COMP(3,a5);
                              [3, h1, h2, h3]
```

Ci-dessous on montre comment exprimer une fonction de Schur sur les bases des formes monomiales (en `c48`), des fonctions complètes (en `c50`), des fonctions symétriques élémentaires (en `c51`) et des fonctions puissances (en `c52`).

```
(c48)  MON2SCHUR([1,2]);

                                2
(d48)       2 x1 x2 x3 + x1  x2

(c49) COMP2ELE(3,[]);

                         2                          3
(d49)      [3, h1, h1  - h2, h3 - 2 h1 h2 + h1 ]

(c50) ELEM(d49,d48,[x1,x2,x3]);

(d50)            h1 h2 - h3

(c51) ELEM([],d48,[x1,x2,x3]);

(d51)            e1 e2 - e3

(c52) PUI([],d48,[x1,x2,x3]);

                3
              p1    p3
(d52)         --- - --
              3     3

(c53) SCHUR2COMP(h1*h2-h3,[h1,h2,h3]);


(d53)       s
```

```
     1, 2

(c54) SCHUR2COMP(a*h3,[h3]);

(d54)                                    s   a
                                          3
```

On a, en dernières instructions, décomposé $h_1 h_2 - h_3$ et h_3 sur la base des fonctions de Schur.

3.8 Les résolvantes

Soit p un polynôme d'un variable x et de degré n sur un anneau A. Soit $f \in A[x_1, x_2, \ldots, x_n]$ une fonction de transformation et $S_n f$ l'orbite de la fonction f sur l'action du groupe symétrique S_n. Alors la résolvante de p par f, notée $f_*(p)$, est le polynôme unitaire :

$$f_*(p)(y) = \prod_{h \in S_n f} (y - h(\alpha_1, \ldots, \alpha_n)),$$

où $\alpha_1, \ldots, \alpha_n$ sont les racines de p.

Si la fonction de transformation est

1. un polynôme d'une variable ,
2. un polynôme linéaire,
3. une polynôme linéaire dont les coefficients non nuls sont alternés,
4. une polynôme linéaire avec que des coefficients unitaires,
5. un polynôme symétrique en les variables qui apparaissent dans son expression,
6. un monôme dont le coefficient est 1,
7. une fonction associée à un groupe,
8. un polynôme quelconque,

alors chaque résolvante associée est respectivement appelée *résolvante*

1. *unitaire,*

2. *linéaire,*

3. *alternée,*

4. *somme,*

5. *symétrique,*

6. *produit,*

7. *de Lagrange,*

8. *de groupe (non encore implantée),*

9. *générale.*

Nous définissons l'*arité* d'une fonction comme le nombre de variables qui apparaissent dans son expression. En général si une fonction est d'arité k nous notons $f(x_1, \ldots, x_k)$ à la place de $f(x_1, \ldots, x_k, \ldots, x_n)$.

Donnons des exemples pour chacun de ces cas :

1. $f(x) = x^7 - x + 1$ d'arité 1,

2. $f(x_1, x_2) = x_1 + 3x_2$ d'arité 2,

3. $f(x_1, x_2, x_3, x_4) = x_1 - x_2 + 3x_3 - 3x_4$ d'arité 4,

4. $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$ d'arité 3,

5. $f(x_1, x_2, x_3) = 3x_1x_2 + 3x_2x_3 + 3x_1x_3$ d'arité 3,

6. $f(x_1, x_2) = x_1x_2$ d'arité 2,

7. $f(x_1, x_2, x_3) = \epsilon x_1 + \epsilon^2 x_2 + \epsilon^3 x_3$, où ϵ est une racine troisième de l'unité,

8. toute fonction donnée avec le groupe qui lui est associée

9. $f(x_1, x_2, x_3) = x_1 + 2x_2x_3$ d'arité 3.

Il est clair qu'une résolvante somme ou produit est également symétrique et qu'une résolvante somme ou alternée est également linéaires.

Les calculs de ces résolvantes se réalisent de deux manières. Ou bien avec la fonction `RESOLVANTE` ou bien avec un nom spécifique précédé de `RESOLVANTE_`. La liste des fonctions possibles est donc :

- `RESOLVANTE_PRODUIT_SYM(p,x)` qui calcule la liste toutes les résolvantes produit du polynôme `p` dont la variable est `x`
- `RESOLVANTE_UNITAIRE(p,q,x)` qui calcule la résolvante du polynôme `p(x)` par le polynôme $q(x)$ Soit $\prod_{p(\alpha)=0}(y-q(\alpha))$
- `RESOLVANTE_ALTERNEE1(p,x)` qui calcule la transformation de `p(x)` par la fonction $\prod_{1\leq i<j\leq 7}(x_i-x_j)$
- `RESOLVANTE_KLEIN(p,x)` qui calcule la transformation de `p(x)` par la fonction $x_1x_2+x_3$
- `RESOLVANTE_KLEIN(p,x)` qui calcule la transformation de `p(x)` par la fonction $x_1x_2x_4+x_4$
- `RESOLVANTE_VIERER(p,x)` qui calcule la transformation de `p(x)` par la fonction $x_1x_2-x_3x_4$
- `RESOLVANTE_DIEDRAL(p,x)` qui calcule la transformation de `p(x)` par la fonction $x_1x_2+x_3x_4$
- `RESOLVANTE(p,x,f,[x1,x2,...,xk])` qui calcule la transformation de `p(x)` par la fonction `f` d'arité `k` et de variables `x1,x2,...,xk` ; (L'arité d'une fonction est le nombre minimum de variables nécessaire pour l'écrire).

Il est essentiel pour l'efficacité des calculs de ne mettre dans la liste des variables de `f` que celle qui apparaissent effectivement dans son expression. Avant d'appeler la fonction `resolvante`, selon le type de résolvante calculée on peut mettre un drapeau à la variable `resolvante` afin d'utiliser l'algorithme le plus adapté. Selon les cas des résolvantes cités plus haut nous devrons mettre le drapeau de la variable `resolvante` respectivement à

1. unitaire,
2. lineaire,
3. alternee,
4. somme,
5. symetrique,
6. produit,

7. lagrange,

8. groupe (non encore implémentée),

9. generale.

On peut généraliser la notion de résolvante par la transformation de r polynôme par une fonction dépendant de r blocs de variables :

- DIRECT $([P_1, P_2, \ldots, P_r]$,y,f,$[lvar_1, lvar_2, \ldots, lvar_r]) \longrightarrow f_*(P_1, P_2, \ldots, P_r)(y)$ qui, étant donné les r polynômes $P_1, \ldots, P_r$ en la variable y de degrés respectifs $d_1, \ldots, d_r$, ramène le polynôme produit des $(y - h(a^{(1)}, \ldots, a^{(p)}))$ où $a^{(i)}$ est le d_i-uplet des racines de P_i pour i allant de 1 à r et où h parcoure toute l'orbite de la fonction f sous l'action du produit de groupes symétriques $S_{d_1} \times \cdots \times S_{d_r}$. Dans $lvar_i$ on met les variables de f qui apparaisent effectivement dans son expression. C'est à dire que $lvar_i$ peut comporter moins de d_i variables. La fonction `direct` se chargera d'en déduire l'action du groupe symétrique S_{d_i}.

Exemples.

```
 resolvante:unitaire;
 RESOLVANTE(x^7-14*x^5  + 56*x^3  - 56*X + 22,x,x^3-1,[x]);

       7      6        5         4          3
     Y  + 7 Y  - 539 Y  - 1841 Y  + 51443 Y

 2
                     + 315133 Y  + 376999 Y + 125253

resolvante : lineaire$
RESOLVANTE(x^4-1,x,x1+2*x2+3*x3,[x1,x2,x3]);
   24       20         16             12
  Y   + 80 Y   + 7520 Y   + 1107200 Y
                               8       4
                  + 49475840 Y  + 344489984 Y + 655360000

resolvante : generale$
RESOLVANTE(x^4-1,x,x1+2*x2+3*x3,[x1,x2,x3]);
   24       20         16             12
  Y   + 80 Y   + 7520 Y   + 1107200 Y
                               8       4
                  + 49475840 Y  + 344489984 Y + 655360000
```

```
RESOLVANTE(x^4-1,x,x1+2*x2+3*x3,[x1,x2,x3,x4]);
   24       20         16             12
  Y   + 80 Y   + 7520 Y   + 1107200 Y
                                8       4
                  + 49475840 Y  + 344489984 Y + 655360000

DIRECT([x^4-1],x,x1+2*x2+3*x3,[[x1,x2,x3]]);

   24       20         16             12
  Y   + 80 Y   + 7520 Y   + 1107200 Y
                                8       4
                  + 49475840 Y  + 344489984 Y + 655360000
```

Nous constatons ainsi que la fonction `DIRECT` est une généralisation de la fonction résolvante.

```
resolvante : lineaire$
RESOLVANTE(x^4-1,x,x1+2*x2,[x1,x2]);
                          12       8        4
                         Y   + 13 Y  + 611 Y  - 625
DIRECT([x^4-1],x,x1+2*x2,[[x1,x2,x3]]);
                          12       8        4
                         Y   + 13 Y  + 611 Y  - 625
```

Le résultat de ce dernier calcul eut été le même avec

```
DIRECT([x^4-1],x,x1+2*x2,[[x1,x2]]);
```

Comme de plus il est plus efficace, il est conseiller de ne pas donner les variables n'intervenant pas dans l'expression de la fonction de transformation.

```
resolvante:lineaire$
RESOLVANTE(x^4-1,x,x1+x2+x3,[x1,x2,x3]);
                                4
                               Y  - 1
resolvante:symetrique$

RESOLVANTE(x^4-1,x,x1+x2+x3,[x1,x2,x3]);
                                4
                               Y  - 1
resolvante:lineaire$
RESOLVANTE(x^4+x+1,x,x1-x2,[x1,x2]);
              12      8       6        4        2
             Y   + 8 Y  + 26 Y  - 112 Y  + 216 Y  + 229
```

```
resolvante:alternee$
RESOLVANTE(x^4+x+1,x,x1-x2,[x1,x2]);
             12      8       6        4        2
            Y   + 8 Y  + 26 Y  - 112 Y  + 216 Y  + 229
resolvante:generale$
RESOLVANTE(x^4+x+1,x,x1-x2,[x1,x2]);
             12      8       6        4        2
            Y   + 8 Y  + 26 Y  - 112 Y  + 216 Y  + 229
```

Nous calculons ci-dessous une image directe de deux manières différentes. La première met en évidence les calculs intermédiaires utilisés par la fonction DIRECT. On peut changer le drapeau de `direct`. On a `puissances` par défaut ce qui veut dire que l'on utilise la fonction MULTI_PUI. Si on fait : `direct : elementaire`, la fonction DIRECT utilise la fonction MULTI_ELEM généralement moins performante.

```
l :  PUI_DIRECT(MULTI_ORBIT(a*x+b*y,[[x,y],[a,b]]),[[x,y],[a,b]],[2,2]);

                                  2  2
               [a x, 4 a b x y + a  x ]

m: MULTI_ELEM([[2,e1,e2],[2,f1,f2]],l[1],[[x,y],[a,b]]);

                         e1 f1

n: MULTI_ELEM([[2,e1,e2],[2,f1,f2]],l[2],[[x,y],[a,b]]);

                          2             2     2   2
            8 e2 f2 - 2 e1  f2 - 2 e2 f1  + e1  f1

PUI2ELE(2,[2,m,n]);

                                     2          2
            [2, e1 f1, - 4 e2 f2 + e1  f2 + e2 f1 ]

ELE2POLYNOME(%,y);

                2                        2          2
               y  - e1 f1 y - 4 e2 f2 + e1  f2 + e2 f1

DIRECT([z^2  - e1* z + e2, z^2  - f1* z + f2], z, b*v + a*u,
            [[u, v], [a, b]]);

                2                        2          2
```

```
y  - e1 f1 y - 4 e2 f2 + e1  f2 + e2 f1
```

- `PUI_DIRECT(`$[f_1, \ldots, f_q]$`,`$[lvar_1, \ldots, lvar_p]$` , `$[d_1, d_2, ..., d_p]$`)`
  Soit f un polynome en r blocs de variables $lvar_1, \ldots, lvar_r$ Soit c_i le nombre de variables dans $lvar_i$ et S_C le produit des r groupes symétriques S_{c_i} de degrés respectifs $c_1, ..., c_r$. Ce groupe agit naturellement sur f. La liste `ORBITE` est l'orbite, notée $S_C f$, de la fonction f sous l'action de S_C. (Cette liste peut être obtenue avec la fonction `MULTI_ORBIT`). Les d_i sont des entiers tels que $c_1 \leq d_1$ $, c_2 \leq d_2$ $, \ldots, c_p \leq d_p$. Notons S_D le produit des groupes symétriques $S_{d_1} \times S_{d_2} \times \ldots \times S_{d_p}$.

  La fonction `PUI_DIRECT` ramène les N premières fonctions puissances de l'orbite $S_D f$ de la fonction f déduites des fonctions puissances de l'orbite $S_C f$ où N est le cardinal de $S_D f$.

  Le résultat est rendu sous forme multi-contractée par rapport à S_D. (i.e. on ne conserve qu'un élément par orbite sous l'action de S_D).

```
L:[[x,y],[a,b]]$

pui_direct ([b*y + a*x, a*y + b*x],L,[2,2]);

                                  2  2
            [a x, 4 a b x y + a  x ]

 pui_direct([b*y + a*x, a*y + b*x], L,[3,2]);

                          2  2      2    2         3  3
 [2 A X, 4 A B X Y + 2 A  X , 3 A  B X  Y + 2 A  X ,

      2  2  2  2      3    3        4  4
  12 A  B  X  Y  + 4 A  B X  Y + 2 A  X ,

      3  2  3  2      4    4        5  5
  10 A  B  X  Y  + 5 A  B X  Y + 2 A  X ,

      3  3  3  3       4  2  4  2      5    5        6  6
  40 A  B  X  Y  + 15 A  B  X  Y  + 6 A  B X  Y + 2 A  X ]

pui_direct ([y+x+2*c, y+x+2*b, y+x+2*a],[[x,y],[a,b,c]],[2,3]);
```

```
                         2              2
[3 x + 2 a, 6 x y + 3 x  + 4 a x + 4 a ,

      2                    3        2       2        3
     9 x  y + 12 a x y + 3 x  + 6 a x  + 12 a  x + 8 a ]
```

Evidemment, on retrouve le même résultat ainsi

```
pui_direct([y+x+2*a],[[x,y],[a]],[2,3]);
```

```
                         2              2
[3 x + 2 a, 6 x y + 3 x  + 4 a x + 4 a ,

      2                    3        2       2        3
     9 x  y + 12 a x y + 3 x  + 6 a x  + 12 a  x + 8 a ]
```

Signification des objets

`card` est le cardinal de l'ensemble des variables sur lequel on travaille.

e_i : ième fonction symétrique élémentaire.

p_i : ième fonction puissance.

h_i : ième fonction complète.

`ele` = $[e_1, e_2, e_3, ..., e_n]$, n intervenant dans le descriptif des fonctions.

`cele` = [card,$e_1, e_2, e_3, ..., e_n$]

`pui` = $[p_1, p_2, p_3, ..., p_m]$, m intervenant dans le descriptif des fonctions.

`cpui` = [card,$p_1, p_2, p_3, ..., p_m$].

`ccomp` = [card,$h_1, h_2, h_3, ..., h_m$].

`sym` : polynôme symétrique sans précision sur sa représentation.

`fmc` : forme monomiale contractée.

`part` : partition.

`tc` : terme contracté.

`tpart` : terme partitionné.

`psym` : polynôme symétrique sous sa forme étendue.

`pc` : polynôme symétrique sous une forme contractée.

`multi_pc` : polynôme multisymétrique sous une forme multicontractée sous S_D.

`ppart` : polynôme symétrique sous sa forme partitionnée.

`P(`$x_1,\ldots,x_q$`)` : polynôme en $x_1,\ldots,x_q$.

`lvar` : liste de variables.

$[lvar_1,\ldots,lvar_r]$: liste de listes de variables.

Contents