Symbolic Computation with Symmetric Polynomials an Extension to Macsyma

Annick Valibouze

SYMBOLIC COMPUTATION WITH SYMMETRIC POLYNOMIALS AN EXTENSION TO MACSYMA

Annick Valibouze
LITP,
4, Place Jussieu, 75252 Paris Cedex 05
and "GRECO DE CALCUL FORMEL" No 60
UUCP: avb@litp.univ-p6-7.fr

Abstract

This paper presents SYM, an extension package to MACSYMA symbolic manipulation system. SYM allows to deal efficiently with symetric polynomials which appear to be useful in several symbolic algorithms such as the computation of resolvants, resultants or minimal polynomials.

The SYM package proposes various algorithms which allows to test the symetricity of polynomials, to compute orbits of symetric polynomials, to make algebraic operations on them, to realize change of basis for the symetric polynomials algebra and so on. Our algorithms are very efficient because they use a particular representation of the symetric polynomials using only one monomial associated with each orbit. This contracted representation has been possible thanks to the symetricity of the polynomials that we consider. The main resulting property is that we avoid the combinatory explosion associated to the exponential development of the symetric group of the variables of a polynomial.

Like MACSYMA, the SYM extension is written in the Lisp programming language because it uses heavily MACSYMA primitive tools. SYM has been developped under UNIX 4.3 bsd in Franzlisp.

Our presentation will include a description of the SYM package and some interesting applications built on SYM. The paper will be an english version of the joined presentation of the system.

INTRODUCTION

We present here a package of manipulations of symmetric polynomials implemented in Franzlisp. This package, called `SYM`, constitutes at present an extension of the system of symbolic computation `MACSYMA`. It performs a few manipulations on symmetric polynomials; it can also be used for direct applications. Some algorithms extend easily to functions that are symmetric with respect to sets of variables (i.e. multi-symmetric functions); these functions will be dealt with in the present paper.

1 Definitions and notations

1.1 Partitions and multi-partitions

Let us first introduce the notion of a partition, which is the basic object that allows us to represent the symmetric polynomials in the most possible contracted form. For more details, the reader is referred to [Andrews] or [Macdonald].

A *partition* is a finite or infinite sequence $I = (i_1, i_2, \ldots, i_n, \ldots)$ of non-negative integers in decreasing order: $i_1 \geq i_2 \geq \cdots i_n \geq \cdots$, and containing only a finite number of non-zero terms. We make the convention that sequences only differing by the number of zeros at the end are equal. For example $(2,1)$ and $(2,1,0,0)$ are the same partition. The non-zero i_k of I are called the *parts* of I. The number of parts is the *length* of I and the sum of the parts is the *weight* of I. We shall call *multi-partition of order* p a finite sequence I of length p, $I = (I_1, I_2, \ldots, I_p)$, where each I_k is a partition.

1.2 Generalities about symmetric functions

Let $\mathcal{A}$ be a ring, and let $D = (d_1, d_2, \ldots, d_p)$ be an element of $\mathbf{N}^p$ with $d_1 + d_2 + \cdots + d_p = n$. Let $X = (x^{(1)}, x^{(2)}, \ldots, x^{(p)})$, where each $x^{(r)}$ is an alphabet of d_r variables $x_1^{(r)}, x_2^{(r)}, \ldots, x_{d_r}^{(r)}$. Then $R_D = \mathcal{A}[X]$ is the ring of polynomials in the n variables $x_i^{(r)}$ ($i = 1, \ldots, d_r$ and $r = 1, \ldots, p$) with coefficients in $\mathcal{A}$. The product $S_{d_1} \times S_{d_2} \times \cdots \times S_{d_p}$ of the symmetric groups S_{d_i} will be denoted by S_D.

For each element σ of S_n and each finite sequence of n elements $T = (t_1, t_2, ..., t_n)$, $\sigma(T)$ is the sequence $(t_{\sigma(1)}, t_{\sigma(2)} ... t_{\sigma(n)})$. This generalizes as follows: let $T = (t^{(1)}, t^{(2)}, \ldots, t^{(p)})$ be a p-tuple of finite sequences $t^{(r)}$ of d_r element. For each element $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_p)$ of S_D we define:

$$\sigma(T) = (\sigma_1(t^{(1)}), \sigma_2(t^{(2)}), \ldots, \sigma_p(t^{(p)})).$$

$G_{S_D}(T)$ will be the *stabilizer* of T under the action of S_D (the subgroup of the elements of S_D leaving T unchanged). If $f \in R_D$, $O_{S_D}(f)$ will denote the orbit of f under S_D, i.e., the set of polynomials h of R_D such that $h(X) = f(\sigma(X))$ for an element σ of S_D.

A polynomial P of R_D is said to be *multi-symmetric* of order D if $P(X) = P(\sigma(X))$ for all $\sigma \in S_D$ (i.e. $card(O_{S_D}(P)) = 1$). This algebra of invariants will be denoted by $R_D^{S_D}$. If $p = 1$ we simply say that P is *symmetric* .

For $p = 1$ we take $D = d_1 = n$ and $X = (x_1, x_2, \ldots, x_n)$. If $U = (u_1, u_2, ..., u_n)$ is an element of $\mathbf{N}^n$, we define the monomial X^U by:

$$X^U = x_1^{u_1} x_2^{u_2} ... x_n^{u_n},$$

and if U is a D-tuple having p finite sequences of integers $u^{(1)}, \ldots, u^{(p)}$ of length $d_1, \ldots, d_p$, respectively, then:

$$X^U = (x^{(1)})^{u^{(1)}} \ldots (x^{(p)})^{u^{(p)}}.$$

With a multi-partition I we associate the *monomial form* $M_I(X)$, which is the sum of the elements of the orbit of X^I under the S_D-action:

$$M_I(X) = \sum_{\sigma \in S_D / G_{S_D}(I)} X^{\sigma(I)}.$$

Examples:

- $p = 1$ - $M_{(3,2,2)}(x, y, z) = x^3y^2z^2 + y^3x^2z^2 + z^3x^2y^2$.
- $p = 2$ - For $X = ((x, y), (a, b, c))$ we have $M_{((3,2),(1,1))}(X) = x^3y^2ab + x^3y^2ac + x^3y^2bc + y^3x^2ab + y^3x^2ac + y^3x^2bc$.

1.3 Contracted and partitioned forms

A *monomial form* $M_I(X)$ will be represented either by a monomial X^J of the orbit of X^I, called a *contracted monomial form*, or by the partition I. If we give it a coefficient, then the monomial form is represented by a *contracted term* or by a *partitioned term*, the latter being a list in which the first element is the coefficient and the rest is the partition. We can now represent a symmetric polynomial (or a multi-symmetric polynomial) by a *contracted polynomial*, the sum of the contracted terms, or by a *partitioned polynomial*, the list of the partitioned terms.

For example the contracted polynomial associated with the polynomial $3x^4 + 3y^4 - 2xy^5 - 2x^5y$, symmetric in the variables x et y, is $3x^4 - 2xy^5$ and the partitioned polynomial is [[3,4],[-2,5,1]].

1.4 Types of arguments

For the description of the function we use the following notations:

`card` is the cardinality of the set of the variables.
e_i : i^{th} elementary symmetric function
p_i : i^{th} power function
`l_ele` $= [e_1, e_2, e_3, ..., e_n]$, where the number n is important in some definitions of the function
`l_cele` $=$ [card,$e_1, e_2, e_3, ..., e_n$]
`l_pui` $= [p_1, p_2, p_3, ..., p_m]$
`l_cpui` $=$ [card,$p_1, p_2, p_3, ..., p_m$]
`sym` $< - - - >$ is a symmetric polynomial, but the representation is not specified
`fmc` $< - - - >$ contracted monomial form
`part` $< - - - >$ partition
`tc` $< - - - >$ contracted term
`tpart` $< - - - >$ partitioned term
`psym` $< - - - >$ symmetric polynomial in its extended form
`pc` $< - - - >$ symmetric polynomial in a contracted form
`multi_pc` $< - - - >$ multi-symmetric polynomial in a contracted form under S_D
`ppart` $< - - - >$ symmetric polynomial in its partitioned form
`P`$(x_1, \ldots, x_q)$ is a polynomial in the variables $x_1, \ldots, x_q$
`lvar` is a list of variables of X in the case $p = 1$. $[lvar_1, \ldots, lvar_p]$ is a list of lists of variables representing the multi-alphabet X and where the variables of $lvar_j$ represent the d_j variables of the alphabet $x^{(j)}$.

Remarks :
1- The functions of `SYM` can complete the lists, such as `l_cele`, with formal values. This values are `ei` for the i^{th} elementary symmetric function and `pi` for the i^{th} power function.
2- There exist many kinds of evaluations for the polynomials under MACSYMA : `ev, expand, rat, ratsimp`. With `SYM` the choice is possible with a flag `oper`. In each call of a function, `SYM` tests if the variable `oper` is modified. In this case, the modification is made as follows: if `oper = meval`, it uses the `ev` mode, if `oper = expand`, it uses the `expand` mode (it is more efficient

for the numeric calculations), if `oper = rat`, it uses the `rat` form and if `oper = ratsimp`, it uses the `ratsimp` form.

2 Description of the available functions

2.1 Change of representation

- `tpartpol(psym,lvar)` $\longrightarrow$ `ppart` ,
  `partpol(psym, lvar)` $\longrightarrow$ `ppart`
  give, in the lexicographic order, increasing and decreasing, respectively, the partioned polynomial associated with the polynomial `psym`. The function `tpartpol` tests if the polynomial `psym` is actually symmetric.

- `contract(psym,lvar)` $\longrightarrow$ `pc`
  `tcontract(psym,lvar)` $\longrightarrow$ `pc`
  act as `partpol` and give the contracted form.

- `cont2part(pc,lvar)` $\longrightarrow$ `ppart`
  gives the partitioned polynomial associated with the contracted form `pc`.

- `part2cont(ppart,lvar)` $\longrightarrow$ `pc`
  gives a contracted form associated with the partioned form `ppart`.

- `explose(pc,lvar)` $\longrightarrow$ `psym`
  gives a contracted form associated with the extended form `psym`.

.
.
.
.
.
Examples:

```
tpartpol(expand(x^4+y^4+z^4-2*(x*y+x*z+y*z)),[x,y,z]);

                        [[1, 4], [- 2, 1, 1]]
```

Now suppose that the polynomial $2a^3*b*x^4*y$ is the contracted form of a symmetric polynomial in $\mathbf{Z}[x,y,z]$.

```
pc : 2*a^3*b*x^4*y$

psym : explose(pc,[x,y,z]);
      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
```

If we use the function `contract` we find again the contracted form:

```
contract(psym,[x,y,z]);
                            3    4
                        2 a  b x  y
partpol(psym,[x,y,z]);
                             3
                       [[2 a  b, 4, 1]]
ppart : cont2part(pc,[x,y,z]);
                             3
                       [[2 a  b, 4, 1]]
part2cont(ppart,[x,y,z]);
                            3    4
                        2 a  b x  y
```

2.2 The partitions

- `kostka(part1,part2)` (written by P.ESPERET) gives the Kostka number associated with the partitions `part1` et `part2`.
- `treinat(part)` $\longrightarrow$ list of the partitions that are less than the the partition `part` in the natural order and that are of the same weight.
- `treillis(n)` $\longrightarrow$ list of the partitions of weight equal to n.
- `lgtreillis(n,m)`
- `lgtreillis(n,m)` $\longrightarrow$ list of the partitions of weight equal to n and of length equal to m.
- `ltreillis(n,m)`
- `lgtreillis(n,m)` $\longrightarrow$ list of the partitions of weight equal to n and the length less than or equal to m.

2.3 The orbits

- `orbit(p(`$x_1,\ldots,x_n$`),lvar)` $\longrightarrow$ O_{S_n}`(p)`
  gives the list of polynomials of the orbit O_{S_n}(p) where the n variables of `p` are in the list `lvar`. This function does not consider the possible symmetries of `p`.

- `multi_orbit(p,[lvar`$_1$`,lvar`$_2$`,...,lvar`$_p$`])` $\longrightarrow$ O_{S_D}`(p)`
  gives the orbit of `p` under S_D (see above), where the variables of the multi-alphabet X are in the lists `lvar`$_i$.

```
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]
```

2.4 Contracted product of two symmetric polynomials

The formula is in [V1] or [V2] and the proof in [V2].

- `multsym(ppart1, ppart2,card)` $\longrightarrow$ `ppart`
  The arguments are two partioned forms and the cardinality. The result is in partitioned form.

For example, take two symmetric polynomials in their contracted form. We first compute the product in the standard way and then with the function `multsym`. We are in $\mathbf{Z}[x, y]$ and the two contracted forms `pc1` and `pc2` are associated with the two symmetric polynomials `p1` and `p2`.

```
pc1 : x^2*y$
pc2 : x$

p1 : explose(pc1,[x,y]);
                                  2    2
                               x y  + x  y

p2 : explose(pc2,[x,y]);
                                 y + x
```

There is the product of the two symmetric polynomials with the standard operation of `MACSYMA`:

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

we verify below that this is the extended form of the product obtained with the function `multsym`:

```
contract(prod,[x,y]);
                     3       2  2
                  x y  + 2 x   y

ppart1 : cont2part(pc1,[x,y])$
ppart2 : cont2part(pc2,[x,y])$

part2cont(multsym(ppart1, ppart2, 2), [x,y]);
                     2     2  3
                2 x y  + x   y
```

2.5 Change of basis

The monomial forms $M_I(X)$ where I varies in the set of partitions of length $\leq n$ are an $\mathcal{A}$-base of the free $\mathcal{A}$-module $R_D^{S_D}$. The Schur functions also form an $\mathcal{A}$-base of the free $\mathcal{A}$-module. We have the following algebra bases: the elementary symmetric functions ($\mathcal{A}$-base), the power functions ($\mathbf{Q}$-base if $\mathcal{A} = \mathbf{Z}$) and the complete functions ($\mathcal{A}$-base).

When $I = (\overbrace{1, 1, ..., 1}^{i}, 0, 0, ..., 0)$ where $0 \leq i \leq n$, the monomial form $e_i(X) = M_I(X)$ is also called the i^{th} *elementary symmetric function* over X, with the convention $e_0 = 1$ and $e_i = 0$ for $i > n$. When $I = (i)$, the monomial form $p_i(X) = M_I(X) = \sum_{x \in X} x^i$ is called the i^{th} *power function* over X, $(p_0 = n)$. The i^{th} *complete symmetric function*, $h_i(X)$, is the sum of the monomial forms $M_I(X)$ where the weight of I is i, ($h_0 = 1$ and $h_r = 0$ if $r < 0$).
Let $\mathcal{M}$ be a matrix and $I = (i_1, i_2, \ldots, i_n)$ a sequence of $\mathbf{Z}^n$; let $\mathcal{M}_I$ be the minor of $\mathcal{M}$ constructed with the lines $1, 2, \ldots, n$ and the columms $i_1+1, i_2+2, \ldots, i_n+n$, with the convention $\mathcal{M}_I = 0$ if there exists r such that $i_r + r \leq 0$.
Let $S = (h_{i-j})_{i \geq 1, j \geq 1}$ be an infinite matrix:

$$\begin{pmatrix} h_0 & h_1 & h_2 & h_3 & \cdots \\ 0 & h_0 & h_1 & h_2 & \cdots \\ 0 & 0 & h_0 & h_1 & \cdots \\ 0 & 0 & 0 & h_0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \ldots \end{pmatrix}$$

where the h_i are the complete functions. We call *Schur function of index I* the minor S_I.

- `elem(l_cele,sym,lvar)` $\longrightarrow$ `P(e1,..., eq)`
  decomposes the symmetric polynomial `sym` into the elementary symmetric functions with the algorithm in [V1].

- `multi_elem([l_cele`$_1$`,...,l_cele`$_p$`],multi_pc,[lvar`$_1$`,...,lvar`$_p$`])` $\longrightarrow$ `P(l_cele`$_1$`,...,l_cele`
  We have the multi-symmetric polynomial `multi_pc` in its contracted form. This function decomposes successively in each package `l_cele`$_j$ of elementary symmetric functions of the alphabet $x^{(j)}$ (see the section 1.2).

- `pui(l_cpui,sym,lvar)` $\longrightarrow$ `P(p1, ...  ,pq)`
  decomposes the symmetric polynomial `sym` into the power functions with the algorithm in [V1].

- `multi_pui([l_cpui`$_1$`,...,l_cpui`$_p$`],multi_pc,[lvar`$_1$`,...,lvar`$_p$`])` $\longrightarrow$ `P(l_cpui`$_1$`,...,l_cpui`$_p$
  see `multi_elem`.

In this examples the symmetric polynomials are in contracted form.

```
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
```

If the cardinality is 3 we have:

```
elem([3],x**4 - 2*y*z,[x,y,z]);
                       4          2                  2
                     e1  - 4 e2 e1  + 4 e3 e1 + 2 e2  - 2 e2
```

If the cardinality is 3 and $e_1 = 7$ we have:

```
elem([3,7],x^4-2*x*y,[x,y]);
                                2
                   28 e3 + 2 e2  - 198 e2 + 2401
```

For the power functions we know that if the cardinality of the alphabet X is n then the i^{th} power function, $i > n$, depends algebraically on the $p_1, p_2, \ldots, p_n$. For this reason, when the function `pui` completes the list `l_cpui` with formal values and the degree of the polynomial `sym` is greater than n, the i^{th} power functions for $i > n$ do not appear. For this computation the function `pui` uses the function `puireduc` (see below).

For following formulas for the change basis we refer to [Macdonald] and [Lascoux, Schützenberger].

- `ele2pui(m,l_cele)` $\longrightarrow$ `l_cpui`
  gives the first `m` power functions as functions of the elementary symmetric functions with the [Girard]-Newton formulas.

- `pui2ele(n,l_cpui)` $\longrightarrow$ `l_cele`
  gives the elementary symmetric functions when we know the power functions. If the flag `pui2ele` is `girard`, the result is the first n elementary symmetric functions and if its is `close`, the result is the n^{th} elementary symmetric function.

In the following example we find the first 3 elementary symmetric functions when the power functions are generic.

```
pui2ele(3,[]);
                                   2                       3
                                 p1     p2  p3   p1 p2   p1
                [3, p1, --- - --, -- - ----- + ---]
                                  2     2   3      2      6
```

Now the cardinality of the alphabet X is 3 and the first power function is equal to 2. We remark that the 4^{th} elementary symmetric function is zero, because the cardinality is 3. We compute the first three power functions below.

```
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 ]
```

In the next example, since the cardinality is 2, the 3^{th} elementary symmetric function is zero:

```
ele2pui(3,[2]);
                                 2          3
                      [2, e1, e1  - 2 e2, e1  - 3 e1 e2]
```

- `puireduc(n,l_cpui)` $\longrightarrow$ `[card,`$p_1, p_2, p_3, ..., p_n$`]`
  gives the first `m` power functions when the first `n` are known. The cardinality is the first element of `l_cpui`.

In this example `card =2` and we search the first three power functions. We can give numerical values to the first two power functions in the list `l_cpui`.

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

- `ele2comp(m , l_cele)` $\longrightarrow$ `l_ccomp`
  gives the first `m` complete functions as functions of the elementary symmetric functions.

- `pui2comp(n, l_cpui)` $\longrightarrow$ `l_ccomp`
  gives the first `m` complete functions as functions of the power functions.

- `comp2ele(n, l_ccomp)` $\longrightarrow$ `l_cele`
  gives the first `m` elementary symmetric functions as functions of the complete functions.

- `comp2pui(n, l_ccomp)` $\longrightarrow$ `l_cpui`
  gives the first `m` power functions as functions of the complete functions.

- `mon2schur(liste)` $\longrightarrow$ `pc`
  compute the Schur functions as functions of the monomial forms. The list appearing as argument is a p-uple I of intergers, it represents S_I, the Schur function of index I (see Section 1.2).

- `schur2comp(P,[h`i_1`,...,h`i_q`]))` $\longrightarrow$ `list of lists`
  The polynomial `P` is $\mathcal{A}[h_1, \ldots, h_q]$, where the h_i are the complete functions. This function expresses `P` as a function of the Schur functions, denoted by $\mathtt{S}_I$ in `MACSYMA`. It is imperative to express the complete functions by means of a letter `h` "concatenated" with an integer (ex: `h2` or `h5`).

We first verify that the Schur function $S_{(1,1,1)}$ is equal to the third elementary symmetric function and that $S_{(3)}$ is equal to the third complete function (this is a general result).

```
 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
```

Let us see on two example, how with a circular set of operations we can go back to the initial lists `[3,p1,p2,p3]` and `[[3,h1,h2,h3]`.

```
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]
```

In the next example we show how to express a Schur function through the monomial forms (see the label `c48`), and then through the complete functions (`c50`), the elementary symmetric functions (`c51`) and the power functions (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
```

In the last instruction we have obtained the polynomials $h_1h_2 - h_3$ and h_3 on the basis of the Schur functions.

2.6 Direct images

In this section, we apply the previous functions to the transformations of polynomial equations.

The direct image intruces in [G,L,V] or in [V2] can represente the The resultant (function `resulsym`), the resolvents, such as Galois or Lagrange's (see [V2] chapter 9 p.91), or more generally minimal polynomials, can be seen as *direct images* ([G,L,V] and [V2]). Suppose that $\mathcal{A}$ is a field k. Let f be a function in R_D and let $P_1, P_2, \ldots, P_p$, p polynomials of degrees $d_1, d_2, \ldots, d_p$, respectively. Associate with each P_j the set $(a_1^{(j)}, \ldots, a_{d_j}^{(j)})$ of its d_j roots in an algebraic closure K of k in an arbitrary order ($1 \leq j \leq p$), and choose an evaluation map $E_a : R_D \longrightarrow K$ which is an algebra homomorphism which sends the variable $x_i^{(j)}$ to $a_i^{(j)}$. The *direct image* $f_*(P)$ is the univariate polynomial whose roots are the images under the map E_a of the elements of the f-orbite $O_{S_D}(f)$, i.e.:

$$f_*(P)(x) = \prod_{g \in O_{S_D}(f)} (x - E_a(g)).$$

- `resulsym(p,q,x)` $\longrightarrow$ `resultant(p,q,x)`
  Computes the resultant of the two polynomials, p and q, using changes of basis on sym-

metric functions. The computation is not symmetric in `p` and `q`. The computing time is best when the degree of `p` is less than the degree of `q`.

- `direct` $([P_1, P_2, \ldots, P_p]$,y,f,$[lvar_1, lvar_2, \ldots, lvar_p]) \longrightarrow f_*(P_1, P_2, \ldots, P_p)(y)$
  where the lists $lvar_i$ representing X (see p.3) allow us to find the type of the function `f`.

We now compute the direct image in two different ways. The first one uses, at the top level, the previous functions in order to obtain the elementary symmetric functions of the roots of the polynomial given by the function `direct` (which is the second way). We can change the flag `direct`. If it is `puissances` (the default value) the function `direct` uses the function `multi_pui`. If it is elementary, the function `direct` uses the function `multi_elem` (generally less efficient).

```
l :  pui_direct(multi_orbit(a*x+b*y,[[x,y],[a,b]]),[[x,y],[a,b]]);
                                           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 ]

direct([z^2  - e1* z + e2, z^2  - f1* z + f2], z, b*v + a*u, [[u, v], [a, b]]);
                    2                         2           2
                   z  - e1 f1 z - 4 e2 f2 + e1  f2 + e2 f1
```

The coefficients of this polynomial in z are equal (up to a sign) to the elementary symmetric functions obtained before.

- `somrac(l_ele,k)` $\longrightarrow$ `P(e1,...,en)(x)`
  gives the polynomial whose roots are the sums k by k of the roots of p. The polynomial p is represented by the elementary symmetric functions of these roots, listed in `l_ele`. Here the list `l_ele` cannot be completed by formal values. If the flag `somrac` is `pui` (défault value), the function `somrac` uses the function `pui`, and if it is `elem` then it uses the function `elem`.

- `prodrac(l_ele,k)` is the same function, but here we transform the polynomial using a product instead of the sum.

We remark that these functions are special cases of direct images.
For example, take the polynomial $x^4 - x^3 - 25x^2 + 25x$:

```
somrac([1,-25,-25,0],2);
                    6      5       4       3        2
                   x  - 3 x  - 47 x  + 99 x  + 550 x  - 600 x
```

2.7 Power function on a particular alphabet

Let k be a field.

- `pui_direct(`$[f_1, \ldots, f_q]$`,`$[lvar_1, \ldots, lvar_p]$`)`
  Hypotheses : each f_i is a polynomial in $k[X]$ (see the definition of X in the first section), and each symmetric function on the alphabet $A = (f_1, f_2, \ldots, f_q)$ is multisymmetric under S_D in R_D (i.e. it is in $R_D^{S_D}$). This is the case when $f = f_1$, the function defined in subsection 2.6, and the alphabet represents the orbit under S_D, a product of symmetric groups. The function `pui_direct` computes the first q power functions on the alphabet A. As these functions are multi-symmetric in R_D, the function `pui_direct` gives the power function in a contracted form in $R_D^{S_D}$ (see subsection 1.2).

```
 pui_direct ([b*y + a*x, a*y + b*x],[[x,y],[a,b]]);
                                          2  2
           [a x, 4 a b x y + a  x ]

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

References

[Andrews] 1976, George E. Andrews, *The theory of partitions*, Encyclopedia of Mathematics and its Applications, Vol. 2, Section Number Theory, Addison Wesley.

[Girard], 1629, Invention Nouvelle en Algèbre, Amsterdam.

[G,L,V] 1988, M. Giusti, D. Lazard, A. Valibouze, *Symmetric polynomials and elimination*, Notes informelles de Calcul Formel IX, Prépublication du Centre de Mathématiques de l'Ecole Polytechnique, M810.0987.

[G,L,V] 1988, M. Giusti, D. Lazard, A. Valibouze, *Algebraic transformation of polynomial equations, symmetric polynomials and elimination*, Proceedings of ISSAC-88 (Roma, Italy), Springer-Verlag.

[Lascoux], Alain, 1984-1985, *La résultante de deux polynômes*, Séminaire d'Algèbre M.P. Malliavin.

[Lascoux, Shützenberger] 1985, A. Lascoux, M.P. Schützenberger, *Formulaire raisonné de fonctions symétriques*, L.I.T.P, U.E.R. MATHS, Paris 7, L.A. 248.

[Macdonald], I.G., 1979, *Symmetric functions and Hall polynomials*, Clarendon Press, Oxford.

[V1] 1897, A. Valibouze, *Fonctions symétriques et changements de bases*, Proceedings of the European Conference on Computer Algebra EUROCAL '87 (Leipzig, RDA), Springer-Verlag.

[V2] 1988, A. Valibouze, *Manipulations de fonctions symétriques*, Thèse de l'Université Paris VI.