Using ACE, an Algebraic Combinatorics Environment for MAPLE

Jean-Yves Thibon* and Sébastien Veigneau*
Université de Marne-la-Vallée, Institut Gaspard Monge
2, rue de la Butte Verte, 93166 Noisy-le-Grand CEDEX, France.
URL: `http://www-igm.univ-mlv.fr/~veigneau/HTML/ACE_PAGE.html`
Emails: `jyt@univ-mlv.fr`, `sv@univ-mlv.fr`

Abstract

We present some examples using the ACE library devoted to Algebraic Combinatorics computations. This environment, divided into packages, provides a way to manipulate classical objects such as permutations, tableaux, symmetric functions and different algebras related to the symmetric group. One also finds more recent mathematical objects like noncommutative symmetric functions or Schubert polynomials.

1 Introduction

The first version of the `A`lgebraic `C`ombinatorics `E`nvironment (`ACE`) is composed of various packages dealing with specific objects: `SG` (*symmetric group*), `SGA` (*symmetric group algebra*), `NCA` (*nilCoxeter algebra*), `IDCA` (*idCoxeter algebra*), `HEKA` (*generic Hecke algebra*), `HEQA` (*Hecke algebra*), `SYMF` (*symmetric functions*), `GL` (*linear group*), `NCSF` (*noncommutative symmetric functions*), `TAB` (*tableaux*) and `SP` (*Schubert polynomials*). Version 2.0, which will be available in July 1996, will contain new packages about classical groups, Fock space representations of quantum affine algebras, hyperoctahedral groups, and an application of Schubert polynomials to computations in the ring of polynomials in several variables.

The `SG` library provides functions to manipulate permutations. Basic operations on permutations have been programmed as for instance, conversions between the multiple representations (diagram, code, permutation matrix, ...) of the elements of the symmetric group. Furthermore, we have for example the `Bruhat` function that realizes the comparison between two permutations according to the Bruhat order, or the `Interval` function that computes the list of the Lehmer codes of all permutations that are smaller than a given one, according to this order. Moreover, one can associate to this interval the generating function of the lengths of permutations (`Betti`).

`SGA`, `NCA`, `IDCA` and `HEKA` libraries provide different algebras related to the symmetric group $\mathfrak{S}_n$. They are all generated by elements satisfying braid relations:

$$\begin{aligned} s_i s_j &= s_j s_i \ , \qquad |i-j|>1 \ , \\ s_i s_{i+1} s_i &= s_{i+1} s_i s_{i+1} \ . \end{aligned}$$

*Supported by Procope and the EC network "Algebraic Combinatorics".

However, the square of a simple generator is different according to the algebra: $s_i^2 = 1$ (`SGA`), $s_i^2 = 0$ (`NCA`), $s_i^2 = -s_i$ (`IDCA`) and $s_i^2 = (q_1 + q_2)s_i - q_1 q_2$ (`HEKA`). The first one is the usual algebra of the symmetric group, the last one, the Hecke algebra (with two parameters q_1, q_2). Each of these algebras has a linear basis of elements which are products of simple generators. We also give the Yang-Baxter basis. Special elements of this basis are the idempotents used in the description of irreducible representations of the symmetric group or of the Hecke algebra.

Other functions related to representations of Hecke algebras are given in `HEQA`, in particular matrices, idempotents, characteristic polynomials and inversion (when possible).

Each of these algebras operates on the ring of polynomials in variables x_i: `SgaOnPol`, `NcaOnPol`, `IdcaOnPol` and `HekaOnPol`. The action is also possible in the basis of Schubert polynomials: `SgaOnX`, `NcaOnX`, `IdcaOnX` and `HekaOnX`.

The `SYMF` library provides functions to handle symmetric functions. In particular, we have reprogrammed almost all functions of the first version of the `SF` package of J.Stembridge [15] using specific algorithms for the changes of basis to accelerate computations.

The `GL` library provides functions to manipulate characters of the linear group $GL(n)$. It essentially amounts to consider symmetric functions as polynomials in a fixed number of variables. For instance, given the order of the linear group, the `s2x_n` function converts an expression from the Schur function basis to the basis of monomials (using an algorithm based on Schubert polynomials).

The `NCSF` library [16] provides functions to work with (integral) noncommutative symmetric functions. Changes of basis, ordinary product, internal product and the transformations of alphabet $A \to A/(1-q)$ and $A \to A(1-q)$ have been programmed. These transformations are related to interesting idempotents of the symmetric group algebra.

The package can also deal with quasi-symmetric functions. The basic operations and the dual bases of all usual bases of noncommutative symmetric functions are implemented.

The purpose of the `TAB` library is to manipulate tableaux, contretableaux and more generally words, that can be transformed into tableaux by the Robinson-Schensted correspondence. This library also gives, in the algebra of the symmetric group, the orthogonal idempotents corresponding to standard tableaux. The `PermOnWord` function is a non-trivial action of the symmetric group on words, or tableaux, which is compatible with the Knuth/plactic congruence.

The `SP` library provides functions to compute in the ring of polynomials as a vector space with basis Schubert polynomials or double Schubert polynomials indexed by permutations. `TableX(n)` and `TableXX(n)` contain these two bases corresponding to $\mathfrak{S}_n$. The multiplicative structure in these bases is derived from Monk's formulae.

The ring of polynomials in $x_1, \ldots, x_n$ is a free module over the ring of symmetric polynomials in $x_1, \ldots, x_n$. It has for linear basis the Schubert polynomials indexed by permutations of $\mathfrak{S}_n$. A modification of Monk's formula together with algorithms in the package `SYMF` gives the multiplicative structure.

Schubert polynomials are important in geometry ; their classes ("Schubert cycles") in the cohomology ring of the flag manifold $\mathcal{F}(\mathbb{C}^n)$ (*i.e.* the quotient of $\mathbb{Z}[x_1, \ldots, x_n]$ by the ideal generated by symmetric polynomials in $x_1, \ldots, x_n$ without constant term) are obtained by calling `Flag(n)`.

The `TEX` library provides an environment of functions that produce output which is suitable for a TeX processor. It knows how to format almost all the types defined in *Maple*, while it knows how to translate all standard internal *Maple* functions or objects such as integrals, limits, sums, products, and even arrays and tables.

2 Generating functions for Schubert polynomials ($\mathfrak{S}_n$ and B_n)

Schubert polynomials for the symmetric group can be expressed as certain coefficients in the algebra of divided differences [10] or nilCoxeter algebra [3]. The generating function of Fomin and Kirillov is:

$$\begin{aligned} \mathfrak{S}(\mathbf{x}\,;\mathbf{y}) &= C_1(x_1)\,C_2(x_2)\ldots C_{n-1}(x_{n-1}) && (1)\\ &= \sum_{\mu\in\mathfrak{S}_n} \mathbb{X}_\mu(\mathbf{x},\mathbf{y})\ \partial_\mu\ , && (2)\end{aligned}$$

in which $C_i(x) = \Big(1+(x-y_{n-i})\partial_{n-1}\Big)\Big(1+(x-y_{n-i-1})\partial_{n-2}\Big)\ldots\Big(1+(x-y_1)\partial_i\Big)$ and the x_i and y_i commute with the ∂_μ.

Here is a program computing this generating function. The function `C` computes a $C_i(x_i)$ by evaluating the product of all `(1 + (x - y_alphabet[j-i+1])*Transpo(j, n))` in the nilCoxeter algebra (`NcaMult`):

```
C:=proc(n, i, x, y_alphabet)
  local
    j, # variable for loop...
    r; # the result...
  r:=1;
  for j from i to n-1 do
    r := NcaMult((1 + (x - y_alphabet[j-i+1])*Transpo(j, n)), r)
  od;
  RETURN(r)
end;
```

The call `Transpo(j, n)` returns the elementary transposition exchanging j and $j+1$ in $\mathfrak{S}_n$. Then, the generating function of double Schubert polynomials corresponds to the following instructions:

```
An_Schubertfg:=proc(x_alphabet, y_alphabet)
  local
    i, # variable for loop...
    n, # degree of the symmetric group...
    r; # the result...
  n:=nops(x_alphabet);
  r:=1;
  for i from 1 to n-1 do
    r := NcaMult(r, C(n, i, x_alphabet[i], y_alphabet))
  od;
  RETURN(r)
end;
```

For instance, one has for $\mathfrak{S}_3$:

```
ACE> An_Schubertfg([x1, x2, x3], [y1, y2, y3]);

       (x1 - y1) [2, 1, 3] + [1, 2, 3] + (x2 - y1 + x1 - y2) [1, 3, 2]
```

```
+ (x1 - y2) (x1 - y1) [3, 1, 2] - (x1 - y1) (- x2 + y1) [2, 3, 1]

- (x1 - y2) (x1 - y1) (- x2 + y1) [3, 2, 1]
```

B_n can be canonically embedded into $\mathfrak{S}_{2n}$. More explicitly, it amounts to identify the generators $u_1, \ldots, u_{n-1}$ of the B_n-nilCoxeter algebra with:

$$\begin{array}{ccc} u_{n-1} & = & \partial_1 \partial_{2n-1} \ , \\ u_{n-2} & = & \partial_2 \partial_{2n-2} \ , \\ \vdots & \vdots & \vdots \\ u_1 & = & \partial_{n-1} \partial_{n+1} \ , \end{array}$$

and the extra generator u_0 to ∂_n.

The generating function of B_n-Schubert polynomials of Fomin and Kirillov is:

$$\mathrm{FK}(\mathbf{x}) = B(x_1)\, B(x_2) \ldots B(x_n)\, C'_1(-x_1)\, C'_2(-x_2) \ldots C'_{n-1}(-x_{n-1}) \quad (3)$$

$$= \sum_{\nu \in B_n} \mathfrak{X}_\nu(\mathbf{x})\ u_\nu \quad (4)$$

$$= \sum \mathfrak{X}_\nu(\mathbf{x})\ \partial_\mu \ , \quad (5)$$

μ being the image of ν in $\mathfrak{S}_{2n}$, $B(x) = (1+xu_{n-1}) \ldots (1+xu_1)\,(1+xu_0)\,(1+xu_1) \ldots (1+xu_{n-1})$ and $C'_i(x) = (1+xu_{n-1})\,(1+xu_{n-2}) \ldots (1+xu_i)$. First of all, one has the following function that realizes the embedding of B_n into $\mathfrak{S}_{2n}$, for a simple transposition:

```
TranspoBn:=proc(i, n)
  if (i=0) then
    RETURN(Transpo(n, 2*n))
  else
    RETURN(MultPerm(Transpo(n-i, 2*n), Transpo(n+i, 2*n)))
  fi
end;
```

Now, the call `B(n, x)` stands for $B(x)$ while `Bn_C(n, i, x)` computes $C'_i(x)$:

```
B:=proc(n, x)
  local
    j, # variable for loop...
    r; # the result...
  r:=1;
  for j from n-1 by -1 to 0 do
    r := NcaMult(r, (1 + x*TranspoBn(j, n)))
  od;
  for j from 1 to n-1 do
    r := NcaMult(r, (1 + x*TranspoBn(j, n)))
  od;
  RETURN(r)
end;
```

```
Bn_C:=proc(n, i, x)
  local
    j, # variable for loop...
    r; # the result...
  r:=1;
  for j from i to n-1 do r := NcaMult((1 + x*TranspoBn(j, n)), r) od;
  RETURN(r)
end;

Bn_Schubertfg:=proc(x_alphabet)
  local
    i, # variable for loop...
    n, # B(n)...
    r; # the result...
  n:=nops(x_alphabet);
  r:=1;
  for i from 1 to n do
    r := NcaMult(r, B(n, x_alphabet[i]))
  od;
  for i from 1 to n-1 do
    r := NcaMult(r, Bn_C(n, i, -x_alphabet[i]))
  od;
  RETURN(r)
end;
```

For instance, one obtains for $n = 2$:

```
ACE> map(factor, Bn_Schubertfg([x1, x2]));

      [1, 2, 3, 4] + (x2 + x1) [1, 3, 2, 4] + (x1 + 2 x2) [2, 1, 4, 3]

                  2
      + (x2 + x1)  [2, 4, 1, 3] + x2 (x2 + x1) [3, 1, 4, 2]

                                                    2
      + x1 x2 (x2 + x1) [3, 4, 1, 2] + x2  (x2 + x1) [4, 2, 3, 1]

              2
      + x1 x2  (x2 + x1) [4, 3, 2, 1]
```

Their are other families of B_n-Schubert polynomials. Fomin and Kirillov have defined a second one and shown the link with a third one due to Biley and Haiman. Pragacz and Ratajski [14] describe still another one related to geometrical considerations, and a fifth one is due to Fulton.

3 Yang-Baxter basis

Each of the algebras (SGA, NCA, IDCA, HEKA) has a natural linear basis indexed by permutations. To each of these algebras correspond a solution of the Yang-Baxter, from which one can construct a basis Y_μ defined inductively. In the case of the divided differences algebra, the recursion is, for μ a permutation and s_i a simple transposition such that $l(\mu s_i) > l(\mu)$:

$$Y_{\mu s_i} = Y_\mu (1 + (x_{\mu(i+1)} - x_{\mu(i)})\ \partial_i)\ ,$$

the product being in the nilCoxeter algebra. For instance:

$$Y_{3241} = \Big(1 + (x_3 - x_2)\partial_2\Big)\Big(1 + (x_3 - x_1)\partial_1\Big)\Big(1 + (x_2 - x_1)\partial_2\Big)\Big(1 + (x_4 - x_1)\partial_3\Big)\ .$$

```
ACE> NcaYang([3, 2, 4, 1]);

                       (- x3 + x1) (- x4 + x1) [2, 1, 4, 3] + [1, 2, 3, 4]

 - (- x3 + x2) (- x3 + x1) (- x4 + x1) [3, 1, 4, 2] + (x3 - x1) [1, 3, 2, 4]

                      - (- x3 + x1) (- x2 + x1) (- x4 + x1) [2, 3, 4, 1]

                                  + (- x3 + x1) (- x4 + x1) [1, 3, 4, 2]

            + (- x3 + x2) (- x3 + x1) (- x2 + x1) (- x4 + x1) [3, 2, 4, 1]

                      - (- x3 + x2) (- x3 + x1) (- x2 + x1) [3, 2, 1, 4]

           + (- x3 + x1) (- x2 + x1) [2, 3, 1, 4] + (x4 - x1) [1, 2, 4, 3]

           + (- x3 + x2) (- x3 + x1) [3, 1, 2, 4] + (x3 - x1) [2, 1, 3, 4]
```

Note that the specialization $\mathfrak{S}(x_3, x_2, x_4, x_1\,; x_1, x_2, x_3, x_4)$ coincides with the element `NcaYang([3, 2, 4, 1])` [7].

Yang-Baxter elements can be used in several ways to obtain idempotents [1] in the group algebra of the symmetric group:

```
ACE> r := SgaYang([4, 3, 2, 1]) / 288:
ACE> r := subs(x1=-1, x2=0, x3=1, x4=2, r);

      r := 1/24 [2, 1, 4, 3] + 1/24 [1, 2, 3, 4] + 1/24 [4, 2, 3, 1]

         + 1/24 [3, 1, 4, 2] + 1/24 [3, 4, 1, 2] + 1/24 [2, 4, 1, 3]

         + 1/24 [4, 3, 2, 1] + 1/24 [1, 3, 2, 4] + 1/24 [2, 4, 3, 1]

         + 1/24 [3, 4, 2, 1] + 1/24 [4, 3, 1, 2] + 1/24 [4, 2, 1, 3]

         + 1/24 [1, 4, 3, 2] + 1/24 [4, 1, 3, 2] + 1/24 [4, 1, 2, 3]

         + 1/24 [1, 4, 2, 3] + 1/24 [2, 3, 4, 1] + 1/24 [1, 3, 4, 2]

         + 1/24 [3, 2, 4, 1] + 1/24 [3, 2, 1, 4] + 1/24 [2, 3, 1, 4]

         + 1/24 [3, 1, 2, 4] + 1/24 [2, 1, 3, 4] + 1/24 [1, 2, 4, 3]
```

```
ACE> SgaMult(r, r) - r;
                                        0
```

4 Interpolation

Schubert polynomials or double Schubert polynomials are a linear basis of the ring of polynomials in the variables $x_1, \ldots, x_n$ (possibly with coefficients involving extra variables $y_1,\ldots,y_n$). The two functions x2X and x2XX give a decomposition into the two Schubert bases:

```
ACE> x2X(x3^2);

        X[2, 3, 1] - X[1, 3, 4, 2] - X[1, 4, 2, 3] + X[1, 2, 5, 3, 4]

ACE> x2XX(x3^2, collect);

          XX[1, 2, 5, 3, 4] - XX[1, 4, 2, 3] + (- y2 - y3) XX[1, 3, 2]

                                                                 2
  - XX[1, 3, 4, 2]  + (y4 + y3) XX[1, 2, 4, 3] + XX[2, 3, 1] + y3  XX[1]

ACE> Tox(");                              # we develop the previous expression.

                                           2
                                         x3
```

The Newton interpolation formula projects onto the space generated by double Schubert polynomials indexed by permutations in $\mathfrak{S}_n$:

$$f(\mathbf{x}) \simeq \sum_{\nu} \mathbb{X}_\nu(\mathbf{x},\mathbf{y})\ \partial_\nu\Big(f(\mathbf{y})\Big)\ , \tag{6}$$

in which divided differences act on the $\mathbf{y}$ variables. We provide the function NewtonInterp which realizes this projection for a given n:

```
ACE> Tox(x2XX(x3^2) - NewtonInterp(x3^2, 3));

        - x1 x2 + x1 y2 + y1 x2 - y1 y2 + x1 y1 + y3 x1 - y3 y1

                                              2      2      2      2
          + y2 x2 + y3 x2 - y3 y2 + x3   - y2   - y3   - y1

ACE> Tox(x2XX(x3^2) - NewtonInterp(x3^2, 5));

                                        0
```

The Newton formula allows to express permutations (as operators on polynomials) as linear combinations of divided differences. Indeed, substituting $\mathbf{x} = \mathbf{y}$ and $\mathbf{y} = \mathbf{x}^\mu$, one has:

$$f \longrightarrow f(\mathbf{x}^\mu) = \sum \mathbb{X}_\nu(\mathbf{x}^\mu,\mathbf{x})\ \partial_\nu\Big(f(\mathbf{x})\Big)\ ,$$

i.e.

$$\mu = \sum \mathbb{X}_\nu(\mathbf{x}^\mu,\mathbf{x})\ \partial_\nu\ .$$

References

[1] I.V.Cherednik, *A new interpretation of Gelfand-Tzetlin bases*, Duke. Math. J. **54** (1987), 563–577.

[2] S.Fomin and A.N.Kirillov, *Combinatorial B_n-analogues of Schubert polynomials*, preprint, 1994.

[3] S.Fomin and A.N.Kirillov, *The Yang-Baxter equation, symmetric functions, and Schubert polynomials*, Proceedings of the 5-*th* International Conference on Formal Power Series and Algebraic Combinatorics, Firenze 1993, 215–229.

[4] I.M.Gelfand, D.Krob, A.Lascoux, B.Leclerc, V.S.Retakh and J.-Y.-Thibon, *Noncommutative symmetric functions*, Adv. in Math. **112** (1995), 218–348.

[5] A.Kerber, *Algebraic combinatorics via finite group actions*, Mannheim:BI-Wissenschaftsverlag, 1991.

[6] D.Krob, B.Leclerc and J.-Y.Thibon, *Noncommutative symmetric functions II: Transformations of alphabets*, preprint, 1995.

[7] A.Lascoux, B.Leclerc and J.-Y.Thibon, Flag Varieties and the Yang-Baxter Equation, preprint, 1995.

[8] A.Lascoux and M.P.Schützenberger, *Le monoïde plaxique*, Quaderni de la Ricerca Scientifica, **109** (1981), 129–156.

[9] A.Lascoux and M.P.Schützenberger, *Polynômes de Schubert*, C.R. Acad. Sci. Paris **294** (1982), 447–450.

[10] A.Lascoux and M.P.Schützenberger, *Structure de Hopf de l'anneau de cohomologie et de l'anneau de Grothendieck d'une variété de drapeaux*, C.R. Acad. Sci. Paris **295** (1982), 629–633.

[11] A.Lascoux and M.P.Schützenberger, *Symmetry and Flag manifolds*, Springer L.N. **996** (1983), 118–144.

[12] I.G.Macdonald, *Notes on Schubert polynomials*, Publ. LACIM **6**, UQAM, Montréal, 1991.

[13] I.G.Macdonald, *Symmetric functions and Hall polynomials*, 2_{nd} ed., Oxford: Clarendon Press, 1995.

[14] P.Pragacz and J.Ratajski, *Formulas for Lagrangian and orthogonal degeneraci loci; The $\tilde{Q}$-Polynomials Approach*, to appear in Compositio Math.

[15] J.Stembridge, *SF, a maple package for symmetric functions*, University of Michigan, 1993.

[16] B.C.V.Ung, *NCSF, a Maple Package for Noncommutative Symmetric Functions*, Publ. IGM **95-16**, Université de Marne-la-Vallée, 1995, to appear in The Maple Technical Newsletter.

[17] B.C.V.Ung and S.Veigneau, *ACE, un environnement de calcul en Combinatoire Algébrique*, Proceedings of the 7-*th* Conference "Formal Power Series and Algebraic Combinatorics", Université de Marne-la-Vallée 1995, 557–562.

[18] S.Veigneau, *ACE, an Algebraic Combinatorics Environment for the computer algebra system MAPLE*, *User's Reference Manual*, *Version 1.0*, Publ. IGM **95-14**, Université de Marne-la-Vallée, 1995.