ACE, un environnement de calcul en Combinatoire Algébrique

BUN-CHAN-VORAC UNG* et SÉBASTIEN VEIGNEAU*
ung@univ-mlv.fr veigneau@univ-mlv.fr
Institut Gaspard Monge, Université de Marne-la-Vallée,
2, rue de la Butte Verte, 93166 Noisy-le-Grand CEDEX, France

(13 janvier 1995)

Résumé. On présente ACE, logiciel de calcul en Combinatoire Algébrique. Cet environnement regroupe plusieurs librairies permettant des calculs sur des objets classiques tels que partitions, permutations, tableaux ou fonctions symétriques, ainsi que d'autres plus récents comme les fonctions symétriques non commutatives ou les polynômes de Schubert.

Abstract. We present the library ACE devoted to Algebraic Combinatorics computations. This environment, divided into packages, provides a way to manipulate classical objects such as partitions, permutations, tableaux or symmetric functions, and some more recent ones such as noncommutative symmetric functions or Schubert polynomials.

1 Introduction

L'environnement ACE (Algebraic Combinatorics Environment), que l'on peut obtenir en envoyant une demande par courrier électronique à l'adresse `veigneau@univ-mlv.fr`, regroupe des librairies de fonctions *Maple* pour la Combinatoire Algébrique. On trouve notamment les librairies NCA (*nilCoxeter algebra*), NCSF (*noncommutative symmetric functions*), SG (*symmetric group*), SGA (*symmetric group algebra*), SP (*Schubert polynomials*), SYMF (*symmetric functions*) et TAB (*tableaux*). On a recherché la compatibilité avec les logiciels déjà existants, en particulier ceux de J.STEMBRIDGE (coxeter, posets, SF, weyl).

Voici un aperçu de quelques-unes de ces librairies.

*Supported by the EC network "Algebraic Combinatorics".

2 La librairie SG

Elle fournit des fonctions de manipulation des permutations. Au-delà des fonctions classiques réalisant des conversions entre les différentes représentations des permutations (diagrammes, codes, matrices de permutation, ...), on trouve par exemple la fonction `Bruhat` de comparaison de deux permutations pour l'ordre de *Bruhat*, ou encore la fonction `Interval` qui calcule l'intervalle des permutations inférieures à une permutation donnée pour cet ordre :

```
ACE> Interval([3,4,1,2]);          # code des permutations inferieures.

 [[2, 2, 0, 0], [2, 1, 0, 0], [2, 0, 1, 0], [2, 0, 0, 0], [1, 2, 0, 0],

  [1, 1, 0, 0], [1, 0, 1, 0], [1, 0, 0, 0], [0, 2, 1, 0], [0, 2, 0, 0],

  [0, 1, 1, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 0]]
```

On peut naturellement associer à cet intervalle la fonction génératrice des longueurs des permutations. Par exemple, l'expression

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

                          4       3       2
                         q  + 4 q  + 5 q  + 3 q + 1
```

correspond, en termes géométriques, au polynôme de Poincaré

$$\sum q^k \dim \mathrm{H}^{2k}(X_{3,4,1,2}, \mathbb{Q})$$

de la variété de Schubert $X_{3,4,1,2}$ de la variété $\mathcal{F}_4$ des drapeaux complets de $\mathbb{C}^4$.

3 La librairie SGA

Elle permet un calcul dans l'algèbre du groupe symétrique $\mathbb{K}[\mathfrak{S}_n]$ où $\mathbb{K}$ est un anneau de polynômes. On trouve par exemple la fonction `SgaOnPol` qui réalise l'action d'un élément de cette algèbre sur un polynôme ou encore `SgaMult` qui multiplie deux éléments de l'algèbre :

```
ACE> K3:=[1,2,3]+q*([2,1,3]+[3,1,2])+q^2*([1,3,2]+[2,3,1])+q^3*[3,2,1]:
ACE> G3:=[1,2,3]-q*([2,1,3]+[3,1,2])+q^2*[3,2,1]:
ACE> map(factor, SgaMult(expand(K3), expand(G3)));

                            2                   2
              (q + 1) (q  + q + 1) (q - 1)  [1, 2, 3]
```

4 La librairie SYMF

Cet ensemble de fonctions permet un calcul sur les fonctions symétriques. On reprogramme en fait une grande partie des fonctions de la première version de la librairie SF de J.Stembridge [6] en utilisant des algorithmes spécifiques pour les changements de bases ce qui permet d'accélérer certains calculs. On a veillé à avoir une syntaxe identique à celle de SF.

Les formules de *Newton*, *Wronski*, *Muir*, *Pieri* ou encore la règle de *Littlewood-Richardson* sont au coeur des changements de bases usuels. Par exemple, en itérant la formule de *Pieri* sur la fonction de Schur associée à la partition vide, on obtient tout naturellement l'expression d'un produit de fonctions élémentaires en termes de fonctions de Schur.

```
ACE> PieriSf(h3, s[3,2,2,1]);                       # formule de Pieri.

 s[3, 3, 2, 2, 1] + s[4, 2, 2, 2, 1] + s[4, 3, 2, 1, 1] + s[4, 3, 2, 2]

     + s[5, 2, 2, 1, 1] + s[5, 2, 2, 2] + s[5, 3, 2, 1] + s[6, 2, 2, 1]
```

La multiplication d'une fonction de Schur par une fonction monomiale repose sur une formule due à Muir :

```
ACE> MuirSf(m[2,2,1], s[4,2,2,1]);
                                    s[5, 3, 3, 1, 1, 1] + s[6, 4, 3, 1]

+ s[5, 4, 4, 1] - s[6, 3, 3, 2]  +  s[6, 4, 2, 2] + s[4, 2, 2, 2, 2, 2]

- s[6, 3, 3, 1, 1] + s[6, 4, 2, 1, 1]  + 3 s[4, 2, 2, 1, 1, 1, 1, 1, 1]

+ 2 s[4, 3, 3, 1, 1, 1, 1]  -  2 s[4, 4, 2, 1, 1, 1, 1] + s[4, 4, 4, 2]

- 2 s[6, 2, 2, 1, 1, 1, 1]  +  s[4, 4, 4, 1, 1]  -  s[6, 2, 2, 2, 1, 1]

+ s[6, 2, 2, 2, 2] -  s[4, 4, 2, 2, 1, 1]  +  s[4, 2, 2, 2, 1, 1, 1, 1]

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

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

- s[6, 3, 2, 1, 1, 1] + s[4, 3, 2, 1, 1, 1, 1, 1] - s[5, 4, 2, 1, 1, 1]

- s[4, 2, 2, 2, 2, 1, 1] + s[4, 3, 3, 2, 1, 1]
```

Les algorithmes généraux de changements de bases font largement intervenir ces formules en fractionnant les calculs en opérations “élémentaires” plus faciles à maîtriser.

L'utilisation des formules spécifiques aux fonctions symétriques nous permet de repousser sensiblement les limites des calculs réalisables et envisageables.

5 La librairie NCSF

Elle permet un calcul sur les fonctions symétriques non commutatives [2]. Les fonctions disponibles autorisent des changements de bases, réalisent le produit ordinaire et le produit intérieur ou encore permettent les transformations de l'alphabet $\mathrm{A} \rightarrow \mathrm{A}/(1-q)$ et $\mathrm{A} \rightarrow (1-q)\mathrm{A}$. Ces transformations donnent naissance à de nombreux idempotents. Par exemple,

$$\varphi_n(q) = \frac{1-q^n}{n} \Psi_n \left(\frac{\mathrm{A}}{1-q} \right)$$

correspond à un idempotent (de Lie) de l'algèbre des descentes [1], ce que l'on vérifie par le calcul suivant :

```
ACE> SpDirect(Ps[3], q):            # on  calcule  Psi_3(A/(1-q)) et on
ACE> ToR(simplify("*(1-q^3)/3));    # developpe dans la base des rubans.

                        q R[2, 1]       R[1, 2]
        1/3 R[3] - 1/3 --------- - 1/3 ------- + 1/3 R[1, 1, 1]
                          q + 1           q + 1
ACE> ToR(Internal(", "));           # Internal est le produit interieur.

                        q R[2, 1]       R[1, 2]
        1/3 R[3] - 1/3 --------- - 1/3 ------- + 1/3 R[1, 1, 1]
                          q + 1           q + 1
```

6 La librairie TAB

Elle propose de manipuler les tableaux, contre-tableaux et plus généralement les mots, qui sont transformés en tableaux grâce à un algorithme dû à Schensted.

Cette librairie donne aussi, dans l'algèbre du groupe symétrique, les idempotents orthogonaux correspondant aux tableaux standards :

```
ACE> Word2Tab([3,1,2]):
ACE> StdTab2Idemp(");

          - 1/6 [3, 2, 1] - 1/6 [1, 3, 2] + 1/3 [1, 2, 3] -

            1/6 [3, 1, 2] - 1/6 [2, 3, 1] + 1/3 [2, 1, 3]
```

Outre la fonction `ListStdTab` calculant la liste des tableaux standards sur n lettres, on trouve aussi la fonction `PermOnWord` qui réalise l'action (compatible avec les relations plaxiques) d'une permutation sur un mot non commutatif (et donc en particulier sur un tableau).

7 La librairie SP

Elle met en place les calculs sur la base des polynômes de Schubert (cf [4], [5]). La formule de Monk qui induit efficacement la structure multiplicative est au coeur de nombreux algorithmes comme par exemple le développement dans la base des monômes (que réalise la fonction `X2x`) qui passe par le calcul de *transitions*. La fonction `ToX` permet de développer un polynôme sur la base Schubert. La table complète des polynômes de Schubert se calcule à l'aide des transitions :

```
ACE> TableX(3);            # calcule la table des polynomes de Schubert
                           # simples  indices   par  les   permutations.

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

La fonction `TableY` réalise le même calcul mais travaille uniquement sur les codes des permutations en utilisant un autre algorithme.

La même combinatoire permet de définir des polynômes de Schubert sur deux ensembles de variables (polynômes de *Schubert doubles*). La multiplication d'un polynôme de Schubert double par une variable est réalisée par la fonction `Monk` (qui étend la formule connue en géométrie dans le cas d'un seul ensemble de variables) :

```
ACE> Monk(4, XX[4,1,2,5,3,7,8,6]);

       - XX[4, 1, 5, 2, 3, 7, 8, 6] - XX[5, 1, 2, 4, 3, 7, 8, 6]

       + XX[4, 1, 2, 7, 3, 5, 8, 6] + XX[4, 1, 2, 6, 3, 7, 8, 5]

                                  + y5 XX[4, 1, 2, 5, 3, 7, 8, 6]
```

Les polynômes de Schubert doubles sont les coefficients universels dans l'interpolation "de Newton" des fonctions de plusieurs variables. Leurs spécialisations (spécialisation du deuxième alphabet sur une permutation du premier alphabet) permet d'exprimer les différences divisées dans la base des permutations, ainsi que les coefficients de Yang-Baxter dans le produit des opérateurs $R(u, v)$:

```
ACE> TableXX(3, [3,1,2]);          # calcule   la   table   des
                                   # polynomes de  Schubert  doubles
                table([             # specialises a y1=x3,y2=x1,y3=x2.
                    [1, 2, 3] = 1
                    [2, 1, 3] = x1 - x3
                    [1, 3, 2] = x2 - x3
                    [2, 3, 1] = (x1 - x3) (x2 - x3)
                    [3, 1, 2] = 0
                    [3, 2, 1] = 0
                ])
```

L'anneau de cohomologie de la variété $\mathcal{F}_n$ des drapeaux complets de $\mathbb{C}^n$ est isomorphe à $\mathbb{Z}[x_1, \ldots, x_n]/I^+$ où I^+ est l'idéal engendré par les polynômes symétriques en $x_1, \ldots x_n$ sans terme constant. Les polynômes de Schubert, qui sont les représentants des cycles de Schubert, permettent un calcul efficace et direct dans cet anneau de cohomologie. En effet, la variété $\mathcal{F}_n$ est munie de n fibrés en droites $\mathrm{L}_1, \ldots, \mathrm{L}_n$ dont les premières classes de *Chern* sont égales aux classes des $x_1, \ldots, x_n$ modulo l'idéal I^+. Le fibré tangent a les mêmes classes de Chern que la somme directe $\bigoplus_{i>j} \mathrm{L}_i \mathrm{L}_j^*$, d'où $c(\mathcal{F}_n) = \prod_{i<j}(1 + x_i - x_j)$. L'exemple suivant illustre le développement de $c(\mathcal{F}_3)$ sur la base des cycles de Schubert :

```
ACE> Flag(3): ToX((1+x1-x2)*(1+x1-x3)*(1+x2-x3));

        X[1, 2, 3] + 2 X[1, 3, 2] + 6 X[2, 3, 1]
                   + 2 X[2, 1, 3] + 6 X[3, 1, 2] + 6 X[3, 2, 1]
```

Références bibliographiques

[1] G.Duchamp, D.Krob, B.Leclerc, J.-Y.Thibon, *Déformations de projecteurs de Lie*, C.R. Acad. Sci. Paris **319** (1994), 909-914.

[2] I.M.Gelfand, D.Krob, A.Lascoux, B.Leclerc, V.S.Retakh, J.-Y.Thibon, *Noncommutative symmetric functions*, Adv. in Math. (à paraître).

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

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

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

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