ACE

an Algebraic Combinatorics Environment

for the computer algebra system MAPLE

User's Reference Manual
Version 3.0

Sébastien Veigneau

Université de Marne-la-Vallée

Institut Gaspard Monge

June 2, 1998

Université de
Marne-la-Vallée
Institut Gaspard Monge

ACE

an Algebraic Combinatorics Environment
for the computer algebra system MAPLE

User's Reference Manual

Version 3.0

ACE

Version 3.0

ACE is a joint effort of a number of people from the University of Marne-la-Vallée. The project has been led since 1994 by Sébastien Veigneau (University of Marne-la-Vallée) who is grateful to all people having made up the version 3.0 of ACE. All the development has been done under supervision of Alain Lascoux (University of Marne-la-Vallée, CNRS) and Jean-Yves Thibon (University of Marne-la-Vallée). Many thanks also to Vincent Prosper for providing the necessary TeX and Perl knowledge to produce this documentation. The following persons have mainly contributed to the ACE project:

Sébastien Veigneau

Institut Gaspard Monge,
Université de Marne-la-Vallée,
Cité Descartes,
5, boulevard Descartes,
Champs-sur-Marne,
77454 Marne-la-Vallée, Cedex 2, France.
Email: `Sebastien.Veigneau@univ-mlv.fr`
Url: `http://phalanstere.univ-mlv.fr/~sv`

Alain Lascoux

Institut Gaspard Monge,
Université de Marne-la-Vallée,
Cité Descartes,
5, boulevard Descartes,
Champs-sur-Marne,
77454 Marne-la-Vallée, Cedex 2, France.
Email: `Alain.Lascoux@univ-mlv.fr`
Url: `http://phalanstere.univ-mlv.fr/~al`

Jean-Yves Thibon

Institut Gaspard Monge,
Université de Marne-la-Vallée,
Cité Descartes,
5, boulevard Descartes,
Champs-sur-Marne,
77454 Marne-la-Vallée, Cedex 2, France.
Email: Jean-Yves.Thibon@univ-mlv.fr
Url: http://www-igm.univ-mlv.fr/~jyt

Vincent Prosper

Institut Gaspard Monge,
Université de Marne-la-Vallée,
Cité Descartes,
5, boulevard Descartes,
Champs-sur-Marne,
77454 Marne-la-Vallée, Cedex 2, France.
Email: Vincent.Prosper@univ-mlv.fr
Url: http://phalanstere.univ-mlv.fr/~vince

Bun-Chan-Vorac Ung

Institut Gaspard Monge,
Université de Marne-la-Vallée,
Cité Descartes,
5, boulevard Descartes,
Champs-sur-Marne,
77454 Marne-la-Vallée, Cedex 2, France.
Email: Ung@univ-mlv.fr
Url: http://phalanstere.univ-mlv.fr/~ung

Copyright notice

> Doing business without advertising is
> like winking at a girl in the dark.
> You know what you are doing,
> but nobody else does.
>
> *New York Herald Tribune, October 30, 1956.*


```
          __/__/ __/_/ __/_/__/
        __/ __/ __/    __/
     __/_/__/ __/    __/_/
   __/    __/ __/    __/
 __/     __/ __/_/ __/_/__/
```

Université de
Marne-la-Vallée
Institut Gaspard Monge

New features of release 3.0

Il ne faut pas dire que la science est utile
parce qu'elle nous permet de construire des machines.
Il faut dire : les machines sont utiles parce qu'elles
nous permettent de faire de la science.

H. Poincaré.

Users already familiar with `ACE` may be interested in the differences between versions 2.0 and 3.0. New users of `ACE` may skip this part and proceed with the next chapter. In the previous documentation, new features of release 2.0 with respect to version 1.0 of `ACE` were identified by a ✘ in the margin. This is not the case for this documentation because many changes have been made. New features of release 3.0 include: ✘

- **New package** `COMP`: this package provides functions to work with compositions. Available functions deal with comparisons, major indexes, conjugates, lists of compositions.
- **New package** `FREE`: this package provides functions to work with the free algebra, that is the algebra of words. It is also possible to work in the plactic monoid, *i.e.* the free algebra of words modulo the plactic relations.
- **New package** `PART`: this package provides functions to work with partitions. Available functions deal with border, conjugate, diagonals, Frobenius notation, hooks, etc. of a partition, and also with lists of (skew) partitions.
- **New package** `SFA`: this is an extension of the `SYMF` package which gives a way to handle symmetric functions viewed as operators acting on the ring of polynomials with real coefficients. It corresponds to an implementation of the Λ-ring theory.
- **New possibilities** in the `SYMF` package: it is from now possible to add new bases of the ring of symmetric functions, by the mean of two new functions `SfAddBasis` and `SfDualBasis`.
- **Many function names** have been changed, many functions have disappeared, either becoming internal functions or being included as options of other more general ones. Function names have been strictly normalized.
- **New functions and algorithms** have been included in already existing packages in order to improve their efficiency. This is in particular the case for the `SG` (symmetric group), the `TAB` (tableaux) and `SYMF` (symmetric functions) packages.

- **Heavy modifications** in the coding of the elements of the different algebras implemented in `ACE`. Elements are now coded as sums of indexed names instead of sums of lists for some compatibility reasons with *Maple V.4*.
- **Heavy internal modifications** in the `CG` (classical groups) package have been performed in order to get it more efficient.
- **ACE installation** has been completely rebuilt in order to use new possibilities with respect to the development of packages, in particular in *Maple V.4*.
- **A test protocol** has been included in `ACE` in order to have the possibility to test whether the installation has been successful or not. It corresponds to the automatic production of test files (including more than 1000 tests) that you should read to test a specific package in `ACE`.
- **Removed package** `SHRI`: this package has been removed from the `ACE` system.
- **Removed package** `FOCK`: this package has been removed from the core of the `ACE` system. However, it is still available as a satellite package using `ACE` functionalities.
- **Some bugs** have been fixed.
- **A new documentation** is included in `ACE` 3.0, available both in postscript format and in the HTML hypertext language.
- **The full system** is available on CD-ROM, including related papers (articles, theses), a mathematical manual explaining many algorithms used in `ACE` and how to experiment with `ACE`, the HTML and postscript documentations. It also contains all sources, all installed (compiled) versions of `ACE` 3.0, together with the old versions of `ACE`.
- **Informations** related to the `ACE` CD-ROM can be obtained from:

  `http://phalanstere.univ-mlv.fr/~ace`

  or by email at:

  `ACE@univ-mlv.fr`.

  In any case, you can contact Alain Lascoux directly by email at:

  `Alain.Lascoux@univ-mlv.fr`.

Preface

ACE is a *Maple* library devoted to Algebraic Combinatorics computations. The project started on April 1994, at the University of Marne-la-Vallée, with the development of three *Maple* packages, dealing respectively with the symmetric group, Schubert polynomials and noncommutative symmetric functions. These packages were available in July 1994. We made up our minds to develop a more general project about algebraic combinatorics in October 1994. The aim was to build an "easy to use" Algebraic Combinatorics Environment. We have chosen *Maple* because this system is widely used by combinatorists and has a convenient user's interface. We had in mind to develop an efficient toolbox of functions for research in algebraic combinatorics. The interface had to be very flexible for first experimentations, whereas it may be more interesting to use some specific programs for large scale computations: for instance, we refer the reader to the `SYMMETRICA` system that is well suited for such computations [3].

We also refer the reader to the new system μ-EC (`http://phalanstere.univ-mlv.fr/~muec`) developped by Vincent Prosper (University of Marne-la-Vallée) which includes the `SYMMETRICA` system (to compute with symmetric functions) and can be seen as an environment of computation in combinatorics for the computer algebra system `MuPAD`. The μ-EC system includes many functions available in `ACE`.

The story started in 1994. `ACE` 1.0 was divided into various packages dealing with specific objects such as noncommutative symmetric functions, Schubert polynomials, tableaux or Hecke algebra, each package having the ability to use some functions defined in the other packages. The first version of `ACE` was available for any MS-DOS or UNIX system, for *Maple V* Release 1, 2 and 3.

The next version 2.0 was available since June 1996 for any MS-DOS, UNIX and MACINTOSH system. It provided new packages and new functions in the already existing packages. Several people had been involved in the project in order to develop new programs as for instance to work with the hyperoctahedral group and with several algebras associated to the hyperoctahedral groups B_n, together with their realizations as algebras of operators on the ring of polynomials. Moreover, a package has been developed to work with the representation theory of classical Lie groups (symplectic and orthogonal groups), whereas a new package was devoted to computations in the polynomial ring $\mathbb{Z}[x_1, \ldots, x_n]$ regarded as a free module of rank $n!$ over the ring $\mathbf{Sym}[x_1, \ldots, x_n]$ of symmetric polynomials, and another one was devoted to calculations in the Fock space representations of the quantum affine algebras $U_q(\widehat{sl}_n)$.

The third version of `ACE`, that is `ACE` 3.0, has been released at the beginning of 1998. Many changes have been made in order to get a coherent system devoted to computations in algebraic combinatorics. Many function names have been changed, many functions have disappeared, either becoming internal functions or being included as options of other more general ones. Function names have been strictly normalized.

`ACE` 3.0 has thus less functions but more functionalities. In particular, several packages have been included such as two packages to handle partitions and compositions of integers, one package to work in the free algebra, one package dealing with symmetric functions on different alphabets.

The ACE system, version 3.0, is now available for *Maple V* Release 1, 2, 3 and 4, for any MS-DOS, UNIX or MACINTOSH system.

This documentation describes how this new version of ACE can be installed and how the new user can make his first steps towards an efficient use of the environment. It gives a complete documentation about all mathematical available functions.

I hope this system will be useful for further developments in computer science in relation with algebraic combinatorics. I would like to thank you for your interest in ACE, "*quant à moi, je vais désormais voler vers d'autres horizons...*".

Sébastien Veigneau,
Université de Marne-la-Vallée, June 2, 1998.

```
############################################################
# Algebraic Combinatorics Environment,  3.0,  June 2, 1998 #
############################################################
# Alain Lascoux, Sebastien Veigneau & Co                   #
# Institut Gaspard Monge, Universite de Marne-la-Vallee,   #
# Cite Descartes, 5 Boulevard Descartes, Champs-sur-Marne, #
# 77454, Marne-la-Vallee, Cedex 2, FRANCE.               #
# Email: ACE@univ-mlv.fr                               #
# URL: http://phalanstere.univ-mlv.fr/~ace           #
####################################################
# ACE  Algebraic Combinatorics Environment         #
#                                                  #
# SFA  Symmetric Functions on different Alphabets #
# FM   Free Module over symmetric polynomials   #
# NCSF NonCommutative Symmetric  Functions    #
# CLG  Characters of Linear Groups          #
# SP   Schubert Polynomials               #
# SYMF SYMmetric Functions              #  __/__/ __/_/_/ __/_/__/
# TAB  TABleaux                       #  __/ __/ __/     __/
# FREE FREE algebra                 #  __/_/__/ __/     __/_/
# BNA  hyperoctahedral Algebras   #  __/   __/ __/     __/
# SGA  Symmetric Group Algebra  #  __/    __/ __/_/_/ __/_/__/
# NCA  NilCoxeter Algebra     #
# IDCA IDCoxeter Algebra     #                _/_/_/    _/_/_/
# HEKA HEcKe Algebra         #                 _/_/    _/  _/
#                              #           _/_/_/ _/ _/_/_/
# BN   hyperoctahedral groups    #
# SG   Symmetric Groups         #
# CG   Classical Groups       #
#                           #
# PART PARTitions         #
# COMP COMPositions      #
#                         #
# ERR  ERRors              #
# TEX  TEX output         #
# TYP  TYPe checking    #
#######################
```

Table of contents

1

Overview

The Algebraic Combinatorics Environment includes various packages providing functions to work with classical structures in algebraic combinatorics (partitions, compositions, permutations, words, tableaux, divided differences, symmetric functions), to compute in several algebras related to the symmetric group and to the hyperoctahedral group. It also gives functions to handle characters and representations of several classical groups, and to manipulate more recent objects like Schubert polynomials, non commutative symmetric functions or quasi-symmetric functions:

- `BN`: hyperoctahedral group.
- `BNA`: algebras related to hyperoctahedral groups.
- `CG`: classical Lie groups.
- `CLG`: characters of the linear groups.
- `COMP`: compositions.
- `FM`: polynomial ring in several variables.
- `FREE`: free algebra of words and free algebra modulo the plactic relations.
- `HEKA`: Hecke algebra.
- `IDCA`: idCoxeter algebra.
- `NCA`: nilCoxeter algebra.
- `NCSF`: noncommutative symmetric functions and quasi-symmetric functions.
- `PART`: partitions.
- `SFA`: symmetric functions on different alphabets.
- `SG`: symmetric group.
- `SGA`: symmetric group algebra.
- `SP`: Schubert polynomials.
- `SYMF`: symmetric functions.
- `TAB`: tableaux.

We give here a brief presentation of these libraries. The background needed to understand all the examples is presented in the next chapter, and is mainly detailed in the mathematical manual available with the `ACE` system.

1.1 The BN library

This package gives functions to work with the hyperoctahedral group, in particular with its Bruhat order. It provides functions to handle the elements of this group, represented by codes. It also gives functions to make conversions between a code and a signed permutation (`Bar`) or a palindromic permutation (`Palin`). For instance, the `BnCode2Interval` function computes the list of all codes of permutations that are smaller than a given one according to the Bruhat order:

```
ACE> r := BnCode2Interval([1,1,3]);

 r := [[1, 1, 3], [1, 1, 2], [1, 2, 1], [1, 1, 1], [1, 0, 3], [1, 0, 2], [1, 2, 0],

       [1, 1, 0], [0, 1, 3], [0, 1, 2], [0, 2, 1], [0, 1, 1], [0, 0, 3], [0, 0, 2],

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

ACE> map(BnCode2Bar, r);

        [[-3, 2, -1], [3, 2, -1], [-2, 3, -1], [2, 3, -1], [-3, -1, 2], [3, -1, 2],

             [-2, -1, 3], [2, -1, 3], [-3, 2, 1], [3, 2, 1], [-2, 3, 1], [2, 3, 1],

               [-3, 1, 2], [3, 1, 2], [-1, 3, 2], [1, 3, 2], [-2, 1, 3], [2, 1, 3],

               [-1, 2, 3], [1, 2, 3]]
```

in which the `BnCode2Bar` converts a code of B_n into a signed permutation.

1.2 The BNA library

This package provides functions to work with several algebras associated to the hyperoctahedral groups B_n, together with their realizations as algebras of operators on the ring of polynomials. It includes the B_n-group algebra together with the B_n-nilCoxeter and B_n-idCoxeter algebra. Bases of these algebras are indexed by codes of B_n. For example, the `BnNcaOnPol` function realizes the action of an element of the B_n-nilCoxeter algebra on a given expression:

```
ACE> BnNcaOnPol(B[1,0,1], x1^3*x2^2*x3);

                                                   2
                                                 x1   x2 x3
```

One has also the `BnIdcaYang` function which calculates a Yang-Baxter element of the B_n-idCoxeter algebra.

The set `BnIdcaYang(code)`, for all codes in B_n is a linear basis of the B_n-idCoxeter algebra, as a free module with coefficients in the x_i's. This basis is defined by recursion:

```
ACE> BnIdcaYang([0,1]);

                         (- x1 x3 + x2 x4) B[0, 1]
               B[0, 0] - -------------------------
                                   x1 x3
```

1.3 The CG library

This package gives functions to work with the representation theory of the classical Lie groups, *i.e.* the symplectic and orthogonal groups. It provides functions to compute products of characters, dimension of irreducible representations together with functions to handle new bases of the ring of symmetric polynomials, that are the orthogonal, named 'O' and the symplectic, named 'Sp', Schur function bases.

Using a special rule, one can for instance easily compute the product of symplectic Schur functions. This is automatically recognized in the ToSp function which expresses any symmetric function in the symplectic Schur function basis:

```
ACE> ToSp(Sp[2,1]^3);

    12 Sp[2, 2, 1, 1, 1] + 15 Sp[2, 2, 2, 1] + 20 Sp[1, 1, 1]  +  43 Sp[2, 1]

         + Sp[2, 2, 2, 1, 1, 1]  +  12 Sp[5, 2]  + 6 Sp[5]  +  42 Sp[2, 2, 1]

         + 18 Sp[4, 1, 1, 1] +  12 Sp[5, 1, 1]  +  3 Sp[6, 1]  +  30 Sp[4, 1]

         + 9 Sp[4, 3, 1, 1] + 8 Sp[4, 3, 2] + 48 Sp[3, 1, 1] + 24 Sp[3, 3, 1]

         + 36 Sp[4, 2, 1]  +  15 Sp[4, 3]   +   24 Sp[3, 2, 2]   +   20 Sp[3]

         + 9 Sp[4, 2, 2, 1] + 4 Sp[4, 4, 1]  +  4 Sp[5, 2, 2]  +  42 Sp[3, 2]

         + 6 Sp[5, 3, 1] + 2 Sp[5, 4] + 6 Sp[1, 1, 1, 1, 1]  +  2 Sp[6, 2, 1]

         + Sp[6, 3] + 6 Sp[5, 2, 1, 1] +  3 Sp[2, 1, 1, 1, 1, 1]  +  12 Sp[1]

         + Sp[4, 1, 1, 1, 1, 1]  +  2 Sp[3, 2, 1, 1, 1, 1]  +  Sp[6, 1, 1, 1]

         + 6 Sp[4, 2, 1, 1, 1] + 2 Sp[5, 1, 1, 1, 1]  +  12 Sp[3, 1, 1, 1, 1]

         + 8 Sp[3, 3, 2, 1]   +   30 Sp[2, 1, 1, 1]   +   6 Sp[3, 2, 2, 1, 1]

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

         + 36 Sp[3, 2, 1, 1] + 2 Sp[3, 3, 3]
```

Moreover, the ToO function expresses any symmetric function in the orthogonal Schur function basis:

```
ACE> ToO(s[2,1]*O[3]);

       O[1, 1] + O[5, 1] + 3 O[3, 1] + 2 O[4] + 2 O[2] + O[4, 2] + O[2, 2]

                  + O[4, 1, 1] + O[2, 1, 1] + O[3, 2, 1]
```

1.4 The CLG library

This library provides functions to manipulate characters of the linear group $GL(n)$. It essentially amounts to consider symmetric functions in a fixed number of variables. For instance, given the order of the linear group, say 3, the `Tox_n` function converts any symmetric function to the basis of monomials:

```
ACE> CLG_n(3);
                                       3

ACE> Tox_n(s[2,1]*h2);

       4          3   2        2   3         4        2   3        3   2         4
     x3  x2 + 2 x3  x2  + 2 x3  x2  + x1 x3  + 2 x1  x3  + 2 x1  x3  + x3 x2

                4     4          3   2        2   3         4        3
          + x3 x1  + x2  x1 + 2 x2  x1  + 2 x2  x1  + x2 x1  + 4 x3  x2 x1

               2   2          2      2          3             2   2              3
          + 6 x3  x2  x1 + 6 x3  x2 x1  + 4 x3 x2  x1 + 6 x3 x2  x1  + 4 x3 x2 x1
```

At the level of the Schur basis, it amounts to consider only partitions having at most 3 parts:

```
ACE> Tos_n(s[2,1]^2);

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

In the basis of product of complete symmetric functions, it amounts to reduce all products of complete functions having more than 3 terms:

```
ACE> Toh_n(h2^2*h1^2);

                                 3                  2
                               h2  + 2 h3 h2 h1 - h3  - 2 h4 h2 + h6
```

1.5 The COMP library

This package provides functions to work with compositions. Available functions give tools to compare, conjugate, generate, etc., compositions.

For instance, the ListCompo function generates compositions of a given weight:

```
ACE> ListCompo(4);

        [[4], [3, 1], [2, 2], [2, 1, 1], [1, 3], [1, 2, 1], [1, 1, 2], [1, 1, 1, 1]]
```

It accepts many options which make it very attractive to generate compositions having certain properties:

```
ACE> ListCompo(2, 'allowzeros', 'lg'=3);

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

One has also the ListCompoFiner function which generates all compositions finer than a given composition, that is all compositions obtained by decomposing the parts of the initial composition. The result is a list of compositions, as shown below:

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

          [[2, 1, 3], [1, 1, 1, 3], [2, 1, 2, 1], [1, 1, 1, 2, 1], [2, 1, 1, 2],

                [1, 1, 1, 1, 2], [2, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1]]
```

1.6 The FM library

This package provides functions to compute in the polynomial ring in n variables seen as a free module of rank $n!$ over the ring of symmetric polynomials in the same variables. We choose as basis of this module Schubert polynomials $\mathbf{X}_\mu$ indexed by permutations μ in the symmetric group $\mathfrak{S}_n$ of degree n. For instance, we can express any polynomial in 4 variables in terms of the Schubert polynomials $\mathbf{X}_\mu, \mu \in \mathfrak{S}_4$ with symmetric functions as coefficients expressed in the basis of Schur functions. The x2XfixScal realizes this operation by computing scalar products:

```
ACE> FM_n(4);

                                              4
ACE> x2XfixScal(q^2*x2^3*x4);

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

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

                  2                             2
               - q  s[1, 1] X[3, 1, 2, 4] - q  s[1, 1] X[2, 3, 1, 4]

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

                  2
               + q  X[2, 4, 3, 1]
```

Furthermore, we also have generalized the Monk formula [70] to describe the multiplicative structure of this module, as illustrated by the following example:

```
ACE> ToXfix(x3*X[3,1,4,2]);

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

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

The package is also able to convert any expression given in the basis of Schubert polynomials indexed by permutations in $\mathfrak{S}_\infty$ in terms of Schubert polynomials in $\mathfrak{S}_n$, with coefficients in $\mathbf{Sym}\,[x_1, \ldots, x_n]$. The following example illustrates this computation:

```
ACE> X2Xfix(X[3,2,1] + q*X[3,2,1] + z*X[4,5,1,3,2], 'collect');

                    (1 + q) X[3, 2, 1, 4] + z s[1, 1, 1] X[3, 4, 1, 2]

                       - z s[1, 1, 1, 1] X[2, 4, 1, 3]
```

1.7 The FREE library

This package provides functions to work with the free algebra, that is the algebra of words. A general element is expressed as a linear combination of `w[word]`, where *word* is a sequence of integers, with arbitrary coefficients.

The `Sf2Free` function converts a symmetric polynomial in n variables into an element of the free algebra. The correspondance amounts to develop each Schur function as a sum of tableaux (seen as special words) of a given shape. Thus, the symmetric function is first expressed in the basis of Schur functions and then each Schur function is converted:

```
ACE> Sf2Free(s[2,1], 3);

           w[2, 1, 1] + w[2, 1, 2] + w[3, 1, 1] + w[3, 1, 3]

         + w[2, 1, 3] + w[3, 2, 2] + w[3, 1, 2] + w[3, 2, 3]
```

The `Pol2Free` function converts a polynomial into an element of the free algebra. The polynomial is first expressed in the basis of products of elementary symmetric functions on an alphabet flag (dual basis of the usual basis of monomials). Then, this product is seen as the noncommutative concatenation of elementary symmetric functions which are seen as sums of increasing words:

```
ACE> Pol2Free((x1+x2)*x1 + x2^2);

                    w[2, 2] + w[1, 2] + w[1, 1]
```

The `FreeShuffle` function computes the shuffle product of two elements of the free algebra:

```
ACE> FreeShuffle(w[1,2,1,1] + w[2,1,1,1], w[1,2,1]);

       4 w[2, 1, 1, 1, 1, 2, 1]  + 6 w[2, 1, 1, 1, 2, 1, 1]

     + 4 w[2, 1, 2, 1, 1, 1, 1]  + 4 w[1, 2, 1, 1, 1, 2, 1]

     + 8 w[1, 2, 1, 1, 2, 1, 1] + 10 w[1, 2, 1, 2, 1, 1, 1]

     + 8 w[1, 2, 2, 1, 1, 1, 1] +  2 w[1, 1, 2, 1, 1, 2, 1]

     + 6 w[1, 1, 2, 1, 2, 1, 1] + 12 w[1, 1, 2, 2, 1, 1, 1]

     + 6 w[2, 1, 1, 2, 1, 1, 1]
```

It is also possible to work in the plactic monoid, *i.e.* the free algebra of words modulo the plactic relations:

```
ACE> Free2Plax(w[1,2,1] + w[2,1,1] + z*w[1,3,2]);

                              2 w[2, 1, 1] + z w[3, 1, 2]

ACE> Free2PlaxClass(w[1,2,1]);

                                w[2, 1, 1] + w[1, 2, 1]

ACE> Free2PlaxClass(w[2,1,1]);

                                w[2, 1, 1] + w[1, 2, 1]

ACE> Free2PlaxClass(w[1,3,2]);

                                w[3, 1, 2] + w[1, 3, 2]
```

The `Free2NilFree` is the inverse plaxification map which transforms a word into a reduced decomposition. The inverse plaxification map has the property to send a plactic class onto a nil plactic class:

```
ACE> r := Free2PlaxClass(w[1,2,1,2]);

                r := w[2, 1, 1, 2] + w[1, 2, 1, 2] + w[1, 2, 2, 1]

ACE> Free2NilFree(r);

                     w[2, 1, 2, 2] + w[1, 1, 2, 1] + w[1, 2, 1, 2]
```

The `NilFree2Free` function transforms each word seen as a reduced decomposition into a word, according to the plaxification map:

```
ACE> NilFree2Free(w[2, 1, 2, 2] + w[1, 1, 2, 1] + w[1, 2, 1, 2]);

                     w[2, 1, 1, 2] + w[1, 2, 2, 1] + w[1, 2, 1, 2]
```

1.8 The HEKA library

This package provides functions to work with the generic Hecke algebra of the symmetric group. This algebra is generated by simple operators τ_i's satisfying braid relations and the Hecke relation: $\tau_i^2 = (q_1 + q_2)\tau_i - q_1 q_2$.

For instance, we have included the `HekaYang` function that computes the Yang-Baxter basis of this algebra (related to the Yang-Baxter equation), seen as a free module with coefficients in the x_i's, q_1, q_2:

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

                    (- x2 + x1) A[2, 1, 3]
                  - ---------------------- + A[1, 2, 3]
                         x1 (q1 + q2)

                                   2
       (- x3 x2 + x1 x2 + x3 x1 - x1 ) A[2, 3, 1]   (x3 - x1) A[1, 3, 2]
     - ------------------------------------------ + --------------------
                              2   2                     x1 (q1 + q2)
                       (q1 + q2)  x1
```

1.9 The IDCA library

It provides functions to work in the idCoxeter algebra (or 0-Hecke algebra) associated with the symmetric group. This is the algebra generated by operators π_i satisfying the braid relations:

$$\begin{aligned}\pi_i\pi_j &= \pi_j\pi_i & &\text{if } |i-j|>1,\\ \pi_i\pi_{i+1}\pi_i &= \pi_{i+1}\pi_i\pi_{i+1} & &1\le i\le n-2,\end{aligned}$$

together with the extra relations: $\pi_i^2 = -\pi_i$. This algebra corresponds to the Hecke algebra for which we set $q = 0$. It is related to *Grothendieck polynomials* and the K-theory of the flag manifold. We can distinguish special elements of this algebra such as the *Demazure* elements: `Idca2Demazure(A[`*perm*`])` where *perm* is a permutation, is defined as the sum of all permutations below *perm* for the Bruhat order, as illustrated by the following lines:

```
ACE> r := Idca2Demazure(A[2,3,1]);

              r := A[2, 3, 1] + A[2, 1, 3] + A[1, 3, 2] + A[1, 2, 3]

ACE> expand( IdcaOnPol(r, x1^2*x2) );

                    2                  2        2        2
                  x2  x3 + x3 x2 x1 + x2  x1 + x1  x3 + x1  x2
```

1.10 The NCA library

It provides functions to work in the nilCoxeter algebra corresponding to the symmetric group. This algebra is realized as the algebra of divided differences acting on polynomials.

The `NcaOnX` function realizes the action of an element of the nilCoxeter algebra on an expression in the $\mathbf{X}$ Schubert basis. A simple divided difference ∂_i acts on a Schubert polynomial $\mathbf{X}_{perm}$ by sending it to 0 if $perm(i) < perm(i+1)$, or to $\mathbf{X}_\nu$ where ν is obtained by transposing $perm(i)$ and $perm(i+1)$, if $perm(i) > perm(i+1)$.

The following example illustrates this action:

```
ACE> NcaOnX(q^4*A[1,3,2] - q^3*A[2,1], z*X[3,2,1] - X[2,1,3]);

                                   4                 3               3
                               z q  X[3, 1, 2] - z q  X[2, 3, 1] + q  X[1, 2]
```

1.11 The NCSF library

This package provides functions to work with (integral) noncommutative symmetric functions together with quasi-symmetric functions. For noncommutative symmetric functions, changes of basis, ordinary product, internal product and the transformations of alphabet $\mathrm{A} \to \mathrm{A}/(1-q)$ and $\mathrm{A} \to \mathrm{A}(1-q)$ have been programmed. These transformations are related to interesting idempotents of the symmetric group algebra. For instance:

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

corresponds to a Lie idempotent of the descent algebra. This can be checked as follows:

```
ACE> r := NcsfTransAlpha(Ps[3], 'direct', 'q'):
ACE> r := map(simplify, ToR(r*(1-q^3)/3, 'collect'));

                     R[2, 1] q       R[1, 2]
          r := - 1/3 --------- - 1/3 ------- + 1/3 R[1, 1, 1] + 1/3 R[3]
                       q + 1          q + 1

ACE> map(simplify, ToR(NcsfInternal(r, r), 'collect'));

                              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
```

The NCSF package gives a function to compute the transition matrices between the different bases of noncommutative symmetric functions. For instance:

```
ACE> NcsfMat(3, 'R', 'S');

                                  [  1    0    0  0 ]
                                  [                 ]
                                  [ -1    1    0  0 ]
                                  [                 ]
                                  [ -1    0    1  0 ]
                                  [                 ]
                                  [  1   -1   -1  1 ]
```

The package can also handle quasi-symmetric functions. It provides the usual changes of basis (quasi-ribbon Schur functions, quasi-monomial functions, . . .).

Because symmetric functions are quasi-symmetric functions, we also provide functions to give the expression of any symmetric function in terms of quasi-symmetric functions. For instance:

```
ACE> Sf2Qsf(s[2,1]*p3);

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

        - 2 F[1, 2, 2, 1] - F[2, 1, 2, 1] - F[1, 2, 1, 2] + F[2, 1, 1, 1, 1]

        + F[1, 2, 1, 1, 1] + F[1, 1, 1, 2, 1] + F[1, 1, 1, 1, 2]
```

1.12 The PART library

This package provides functions to work with partitions. Available functions deal with border, conjugate, diagonals, Frobenius notation, hooks, etc. of a partition, and also with lists of (skew) partitions.

For instance, we can count the number of partitions of 5 with at most 3 parts and with the first part lesser than 3 and the second one lesser than 2:

```
ACE> ListPart(5, 'maxpart'=[3,2], 'maxlg'=3);

                              [[3, 2], [3, 1, 1], [2, 2, 1]]
```

In fact, we are also able to count the number of partitions whose diagram is contained into the diagram of a given partition:

```
ACE> ListPartIn([40,30,15,10,8,7,5,3,1], 'nb');

                                         9631414
```

We also give a function to construct the order matrix on all partitions of n. The considered order is the natural order: $I \geq J$ if $\sum_I \leq \sum_J$ for the lexicographic order, in which $\sum_I = [I_1, I_1+I_2, I_1+I_2+I_3, \ldots]$. The inverse matrix is known as the Moebius matrix:

```
ACE> ListPart(5);

      [[5], [4, 1], [3, 2], [3, 1, 1], [2, 2, 1], [2, 1, 1, 1], [1, 1, 1, 1, 1]]
```

```
ACE> PartOrderMat(5);

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

The package also includes several functions to change the coding of partitions.

1.13 The SFA library

Because symmetric functions can also be considered as operators acting on the ring of polynomials with coefficients in $\mathbb{R}$, the **SFA** package has been included in **ACE**.

In the **SFA** package—*Symmetric Functions on different Alphabets*, we give the possibility of doing more formal computations (with respect to the **SYMF** package for instance) by implementing the structure of Λ-ring of symmetric functions viewed as such operators. In other words, the objects, which are symmetric functions, contain their own alphabets, which are polynomials with coefficients in $\mathbb{R}$.

For instance, the **SFAExpand** function expands, whenever possible, any expression in **SFA**. It is a direct implementation of the Λ-ring structure. Expansion is made considering the shape of the alphabet of each symmetric function appearing in the expression, and also products of symmetric functions:

```
ACE> SfAExpand( p[4,2](7+A1) );

             49 + 7 p[2](A1) + 7 p[4](A1) + p[4, 2](A1)
```

Expansion is inductively propagated, thus enabling chains of plethysms:

```
ACE> SfAExpand( (p[3]/3 + p[2,1]/2 + p[1,1,1]/6)(p[2](6*A1^2) + k) );

          3         2                2
   1/6 k  + 1/2 k  + 1/3 k + 3 k  p[2, 2](A1) + 3 k p[2, 2](A1)

     + 3 k p[4, 4](A1) + 18 k p[2, 2, 2, 2](A1) + 2 p[6, 6](A1)
```

```
               + 18 p[4, 4, 2, 2](A1) + 36 p[2, 2, 2, 2, 2, 2](A1)
```

The `SfAVars` function specifies which variables are considered for the `SfAExpand` function. Thus, after having specified that the x_i's and the t variables must be considered:

```
ACE> SfAVars({ {x}, t });

                                      {t, {x}}
```

we can develop a Schur function as follows:

```
ACE> SfAExpand( s[3](x1+x2-t) );

                     2                 2     3         2     2         3
                - t x2  - t x1 x2 - t x1  + x2  + x1 x2  + x1  x2 + x1
```

1.14 The SG library

It 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 instance a function that gives a way to realize the comparison between two permutations according to the *Bruhat order*, or the `Perm2Interval` function that computes the list of the Lehmer codes of all permutations that are smaller than a given one, according to this order:

```
ACE> r := Perm2Interval([3,4,1,2]);

      r := [[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]]
```

```
ACE> map(Code2Perm, r);

        [[3, 4, 1, 2], [3, 2, 1, 4], [3, 1, 4, 2], [3, 1, 2, 4], [2, 4, 1, 3],

         [2, 3, 1, 4], [2, 1, 4, 3], [2, 1, 3, 4], [1, 4, 3, 2], [1, 4, 2, 3],

                   [1, 3, 4, 2], [1, 3, 2, 4], [1, 2, 4, 3], [1, 2, 3, 4]]
```

One can naturally associate to this interval the generating function of the lengths of permutations. For instance, the following expression:

```
ACE> Perm2Betti([3,4,1,2], 'q');

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

corresponds, in geometric terms, to the Poincaré polynomial:

$$\sum q^k \dim \mathrm{H}^{2k}(X_{3412}, \mathbb{Q}) \quad ,$$

of the Schubert variety X_{3412} of the manifold $\mathcal{F}_4$ of complete flags of $\mathbb{C}^4$.

One can also list or count permutations satisfying certain properties. For instance, there are 103 vexillary permutations in $\mathfrak{S}_5$:

```
ACE> ListPerm(5, 'vexillary', 'nb');

                                      103
```

among which, 15 have 7 inversions and for instance 9 have 8 inversions:

```
ACE> ListPerm(5, 'vexillary', 'nb', 'level');

                   [1, 4, 6, 11, 16, 18, 18, 15, 9, 4, 1]
```

1.15 The SGA library

It provides functions to work in the symmetric group algebra $\mathbb{K}[\mathfrak{S}_n]$ where $\mathbb{K}$ is a ring of coefficients. For instance, we have the `SgaOnPol` function that realizes the action of an element of this algebra on a polynomial, or `SgaMult` which multiplies two elements of the algebra:

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

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

One has also the possibility to generate a central idempotent of the symmetric group algebra corresponding to a given partition, as an element of this algebra. The `Part2SgaCIdemp` function is devoted to this purpose:

```
ACE> r := Part2SgaCIdemp([2,1,1]);

  r := 3/8 A[1, 2, 3, 4] - 1/8 A[1, 2, 4, 3] + 1/8 A[4, 1, 2, 3] - 1/8 A[1, 3, 2, 4]

     - 1/8 A[1, 4, 3, 2] - 1/8 A[3, 4, 1, 2] + 1/8 A[3, 1, 4, 2] + 1/8 A[4, 3, 1, 2]

     - 1/8 A[2, 1, 3, 4] + 1/8 A[2, 4, 1, 3] - 1/8 A[2, 1, 4, 3] + 1/8 A[2, 3, 4, 1]

     - 1/8 A[4, 2, 3, 1] - 1/8 A[3, 2, 1, 4] + 1/8 A[3, 4, 2, 1] - 1/8 A[4, 3, 2, 1]

ACE> SgaMinus(r, SgaMult(r, r));

                                            0
```

1.16 The SP library

It provides functions to compute in the linear basis of Schubert polynomials. The *Monk* formula describing the multiplicative structure is used in many algorithms, for instance to express a Schubert polynomial in the basis of monomials: this is done by the `Tox` function which uses the so-called *transition* algorithm. We can compute the complete table of all Schubert polynomials of a given symmetric group using the `TableX` function as shown in the following example for $\mathfrak{S}_3$:

```
ACE> TableX(3);

                                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
                                  ])
```

The `TableY` function realizes the same calculation but works with codes using another algorithm. The same combinatorics may be used to define Schubert polynomials in two sets of variables (*double Schubert* polynomials). The product of a double Schubert polynomial and a single variable is realized as follows (it extends the formula known in geometry for the intersection of a Schubert variety with a hyperplane):

```
ACE> ToXX(x4*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]
```

Double Schubert polynomials are polynomials in two sets of variables $\{x_i\}, \{y_j\}$, which are the universal coefficients in the Newton interpolation of functions in several variables. For many applications, one needs to specialize the second set of variables to a permutation of the first one. The following example computes the table of double Schubert polynomials specialized to $y_1 = x_3, y_2 = x_1, y_3 = x_2$:

```
ACE> TableXX(3, [3,1,2]);
                              table([
                                  [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
                               ])
```

The cohomology ring of the flag manifold $\mathcal{F}_n$ is isomorphic to $\mathbb{Z}[x_1, \ldots, x_n]/I^+$ where I^+ is the ideal generated by the symmetric polynomials in $x_1, \ldots, x_n$ without constant term. A geometrical linear basis of the cohomology ring is constituted by Schubert cycles, which are the images of Schubert polynomials in $\mathbb{Z}[x_1, \ldots, x_n]/I^+$. Thus Schubert polynomials allow an efficient "Schubert calculus" on flag manifolds.

1.17 The SYMF library

It provides functions to handle symmetric functions. Several bases are available but one can add new ones declared for instance as dual to another already existing basis. This package uses specific algorithms for the changes of basis to accelerate certain computations.

Newton, *Wronski*, *Muir*, *Pieri* formulae and the *Littlewood-Richardson* rule constitute the heart of all algorithms for the classical changes of basis. For instance, iterating the *Pieri* formula on the Schur function corresponding to the empty partition, gives the expression of a product of elementary symmetric functions as a linear combination of Schur functions indexed by partitions:

```
ACE> Tos(h3^2*h2);

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

                                + 3 s[6, 2] + s[6, 1, 1] + 2 s[7, 1] + s[8]

ACE> Tos(h3*s[3,2,2,1]);

            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]
```

The product of a Schur function and a monomial symmetric function is given by the *Muir* formula:

```
ACE> Tos(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]

           + 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]
```

The general changes of basis divide calculations into elementary operations easily computed using these formulae. The package provides also some internal functions to compute specific plethysms. These special functions are called by the `SfPlethysm` function:

```
ACE> SfPlethysm(e4, e4, 'e', 'e', 's');

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

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

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

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

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

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

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

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

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

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

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

The SYMF package gives a function to compute the transition matrices between the different bases of the ring of symmetric functions. For instance:

```
ACE> SfMat(5, 's', 'm');

                                [ 1  1  1  1  1  1  1 ]
                                [                     ]
                                [ 0  1  1  2  2  3  4 ]
                                [                     ]
                                [ 0  0  1  1  2  3  5 ]
                                [                     ]
                                [ 0  0  0  1  1  3  6 ]
                                [                     ]
                                [ 0  0  0  0  1  2  5 ]
                                [                     ]
                                [ 0  0  0  0  0  1  4 ]
                                [                     ]
                                [ 0  0  0  0  0  0  1 ]

ACE> ListPart(5);

       [[5], [4, 1], [3, 2], [3, 1, 1], [2, 2, 1], [2, 1, 1, 1], [1, 1, 1, 1, 1]]
```

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

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

1.18 The TAB library

Its purpose is to manipulate tableaux, contretableaux and more generally words, that are transformed into tableaux by an algorithm due to Schensted.

This library also gives, in the algebra of the symmetric group, the orthogonal idempotents corresponding to standard tableaux:

```
ACE> r := Word2Tab(w[3,1,2]);

                               r := [[3], [1, 2]]

ACE> StdTab2SgaIdemp(r);

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

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

We have for instance, the `ListStdTab` function that computes the list of all standard tableaux filled with n letters. The `Part2ListStdTab` function generates all standard tableaux of a given shape. This is done by generating all Yamanouchi words of a given weight:

```
ACE> Part2ListStdTab([4,2,1]);

               [[[3], [2, 5], [1, 4, 6, 7]], [[3], [2, 6], [1, 4, 5, 7]],

                [[3], [2, 7], [1, 4, 5, 6]], [[4], [2, 7], [1, 3, 5, 6]],

                [[5], [2, 7], [1, 3, 4, 6]], [[6], [2, 7], [1, 3, 4, 5]],

                [[7], [2, 6], [1, 3, 4, 5]], [[7], [3, 6], [1, 2, 4, 5]],

                [[7], [4, 6], [1, 2, 3, 5]], [[7], [5, 6], [1, 2, 3, 4]],

                [[6], [3, 7], [1, 2, 4, 5]], [[6], [4, 7], [1, 2, 3, 5]],
```

```
[[6], [5, 7], [1, 2, 3, 4]], [[5], [3, 7], [1, 2, 4, 6]],

[[5], [4, 7], [1, 2, 3, 6]], [[4], [3, 7], [1, 2, 5, 6]],

[[4], [2, 6], [1, 3, 5, 7]], [[5], [2, 6], [1, 3, 4, 7]],

[[6], [2, 5], [1, 3, 4, 7]], [[7], [2, 5], [1, 3, 4, 6]],

[[7], [3, 5], [1, 2, 4, 6]], [[7], [4, 5], [1, 2, 3, 6]],

[[6], [3, 5], [1, 2, 4, 7]], [[6], [4, 5], [1, 2, 3, 7]],

[[5], [3, 6], [1, 2, 4, 7]], [[5], [4, 6], [1, 2, 3, 7]],

[[4], [3, 6], [1, 2, 5, 7]], [[4], [2, 5], [1, 3, 6, 7]],

[[5], [2, 4], [1, 3, 6, 7]], [[6], [2, 4], [1, 3, 5, 7]],

[[7], [2, 4], [1, 3, 5, 6]], [[7], [3, 4], [1, 2, 5, 6]],

[[6], [3, 4], [1, 2, 5, 7]], [[5], [3, 4], [1, 2, 6, 7]],

[[4], [3, 5], [1, 2, 6, 7]]]
```

We have also the `ListTab` function which returns the list of all tableaux of a given shape and a given weight:

```
ACE> map(Tab2Mat, ListTab([3,2,2], [1,0,0,1,2,1,0,1,1]));

       [ 8  9    ]  [ 8  9    ]  [ 8  9    ]  [ 6  9    ]  [ 6  9    ]  [ 5  9    ]
       [         ]  [         ]  [         ]  [         ]  [         ]  [         ]
      [[ 5  6    ], [ 4  6    ], [ 5  5    ], [ 5  8    ], [ 4  8    ], [ 4  8    ],
       [         ]  [         ]  [         ]  [         ]  [         ]  [         ]
       [ 1  4  5 ]  [ 1  5  5 ]  [ 1  4  6 ]  [ 1  4  5 ]  [ 1  5  5 ]  [ 1  5  6 ]

       [ 6  9    ]  [ 5  9    ]  [ 6  8    ]  [ 5  8    ]
       [         ]  [         ]  [         ]  [         ]
       [ 5  5    ], [ 4  6    ], [ 5  5    ], [ 4  6    ]]
       [         ]  [         ]  [         ]  [         ]
       [ 1  4  8 ]  [ 1  5  8 ]  [ 1  4  9 ]  [ 1  5  9 ]
```

The `PermOnTab` function plactically permutes rows in a product of rows (not necessarily a tableau, but the image of a tableau under a row-permutation).

A simple plactic row-transposition transforms a two-row contre-tableau into the two-row tableau which is obtained from it by Schensted algorithm, or conversely. This transformation is also given by the "jeu de taquin".

```
ACE> PermOnTab([2,1], [[1,3,4,6], [2,5]]);

                              [[3, 6], [1, 2, 4, 5]]
```

1.19 The TEX library

This package 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 convert all standard internal *Maple* functions or objects such as integrals, limits, sums, products, and even arrays and tables. A special format devoted to the objects manipulated in ACE is included by default. For instance:

```
ACE> Tex(s[4,2,2,1,1,1]=SfJtMat([4,2,2,1,1,1]), 'TEX_MODE'='MATHS');
$s_{4, 2, 2, 1, 1, 1}=\pmatrix{S_{4}& S_{5}& S_{6}& S_{7}& S_{8}& S_{9}\cr\cr S_{1} &
S_{2}& S_{3}& S_{4}& S_{5}& S_{6}\cr\cr 1& S_{1}& S_{2}& S_{3}& S_{4}& S_{5}\cr\cr 0&
0& 1& S_{1}& S_{2}& S_{3}\cr\cr 0& 0& 0& 1& S_{1}& S_{2}\cr\cr 0& 0& 0& 0& 1 & S_{1}}$
```

which can be included in a TEX file to produce the following output:

$$s_{4,2,2,1,1,1} = \begin{pmatrix} S_4 & S_5 & S_6 & S_7 & S_8 & S_9 \\ S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \\ 1 & S_1 & S_2 & S_3 & S_4 & S_5 \\ 0 & 0 & 1 & S_1 & S_2 & S_3 \\ 0 & 0 & 0 & 1 & S_1 & S_2 \\ 0 & 0 & 0 & 0 & 1 & S_1 \end{pmatrix}$$

One can also compute for example the plactic class corresponding to a given word as explained previously:

```
ACE> Tex(Free2PlaxClass(w[2,3,1,3,2,1,1]), 'TEX_MODE'='MATHS');
$3\,2\,1\,2\,1\,3\,1+3\,2\,1\,2\,3\,1\,1+3\,2\,2\,1\,3\,1\,1
+3\,2\,2\,3\,1\,1\,1+2\,1\,3\,3\,2\,1\,1+2\,1\,3\,3\,1\,2\,1
+2\,3\,1\,3\,1\,2\,1+2\,3\,3\,1\,1\,2\,1+2\,3\,1\,3\,2\,1\,1
+2\,3\,3\,1\,2\,1\,1+2\,3\,2\,3\,1\,1\,1+2\,3\,3\,2\,1\,1\,1
+2\,3\,2\,1\,3\,1\,1+3\,2\,2\,1\,1\,3\,1+2\,1\,3\,2\,1\,3\,1
+2\,1\,3\,2\,3\,1\,1+2\,1\,3\,1\,3\,2\,1+2\,3\,1\,1\,3\,2\,1
+2\,3\,1\,2\,3\,1\,1+2\,3\,1\,2\,1\,3\,1+2\,3\,2\,1\,1\,3\,1$
```

and obtain the following TEX output: $3\,2\,1\,2\,1\,3\,1+3\,2\,1\,2\,3\,1\,1+3\,2\,2\,1\,3\,1\,1+3\,2\,2\,3\,1\,1\,1+2\,1\,3\,3\,2\,1\,1+$

2133121+2313121+2331121+2313211+2331211+2323111+2332111+2321311+ 3221131+2132131+2132311+2131321+2311321+2312311+2312131+2321131.

Let us see another example about tableaux. By running the next command, we obtain the TEX output corresponding to the list of tableaux (seen as lists of lists), of a given shape and a given evaluation, that is:

```
ACE> Tex(ListTab([4,2,1], [1$7]), 'TEX_MODE'='MATHS');
$[[[7], [5, 6], [1, 2, 3, 4]], [[7], [4, 6], [1, 2, 3, 5]], [
[7], [3, 6], [1, 2, 4, 5]], [[7], [2, 6], [1, 3, 4, 5]], [[7
], [4, 5], [1, 2, 3, 6]], [[7], [3, 5], [1, 2, 4, 6]], [[7],
[2, 5], [1, 3, 4, 6]], [[7], [3, 4], [1, 2, 5, 6]], [[7], [2
, 4], [1, 3, 5, 6]], [[6], [5, 7], [1, 2, 3, 4]], [[6], [4,
7], [1, 2, 3, 5]], [[6], [3, 7], [1, 2, 4, 5]], [[6], [2, 7]
, [1, 3, 4, 5]], [[5], [4, 7], [1, 2, 3, 6]], [[5], [3, 7],
[1, 2, 4, 6]], [[5], [2, 7], [1, 3, 4, 6]], [[4], [3, 7], [1
, 2, 5, 6]], [[4], [2, 7], [1, 3, 5, 6]], [[3], [2, 7], [1,
4, 5, 6]], [[6], [4, 5], [1, 2, 3, 7]], [[6], [3, 5], [1, 2,
4, 7]], [[6], [2, 5], [1, 3, 4, 7]], [[6], [3, 4], [1, 2, 5,
7]], [[6], [2, 4], [1, 3, 5, 7]], [[5], [4, 6], [1, 2, 3, 7]
], [[5], [3, 6], [1, 2, 4, 7]], [[5], [2, 6], [1, 3, 4, 7]],
[[4], [3, 6], [1, 2, 5, 7]], [[4], [2, 6], [1, 3, 5, 7]], [[
3], [2, 6], [1, 4, 5, 7]], [[5], [3, 4], [1, 2, 6, 7]], [[5]
, [2, 4], [1, 3, 6, 7]], [[4], [3, 5], [1, 2, 6, 7]], [[4],
[2, 5], [1, 3, 6, 7]], [[3], [2, 5], [1, 4, 6, 7]]]$
```

But, this output is not very attractive. However, we can transform each tableau into the corresponding word and convert the list of words into a sum to obtain by the next command:

```
ACE> Tex(convert(map(Tab2Word, ListTab([4,2,1], [1$7])), `+`), 'TEX_MODE'='MATHS');
```

the sum: $7461235 + 7241356 + 6571234 + 6471235 + 7351246 + 7251346 + 7341256 + 7261345+7451236+7361245+6351247+6251347+3271456+6451237+4371256+ 4271356+5371246+5471236+5271346+6371245+6271345+7561234+4351267+ 4251367+3251467+5341267+5241367+3261457+4261357+5261347+4361257+ 6241357 + 5461237 + 5361247 + 6341257$.

If we want a two-dimensional representation of tableaux, we can tranform each tableau into the corresponding matrix and print 5 tableaux per row by the next command:

```
ACE> lt := map(Tab2Mat, ListTab([4,2,1], [1$7])):
ACE> for i from 1 by 5 to nops(lt) do
       Tex([seq(op(j, lt), j=i..i+4)], 'TEX_MODE'='MATHS');
       TexStr(``);
     od:
```

Thus, we can obtain the next output:

$$\begin{pmatrix}7&&&\\5&6&&\\1&2&3&4\end{pmatrix},\begin{pmatrix}7&&&\\4&6&&\\1&2&3&5\end{pmatrix},\begin{pmatrix}7&&&\\3&6&&\\1&2&4&5\end{pmatrix},\begin{pmatrix}7&&&\\2&6&&\\1&3&4&5\end{pmatrix},\begin{pmatrix}7&&&\\4&5&&\\1&2&3&6\end{pmatrix}$$

$$\begin{pmatrix}7&&&\\3&5&&\\1&2&4&6\end{pmatrix},\begin{pmatrix}7&&&\\2&5&&\\1&3&4&6\end{pmatrix},\begin{pmatrix}7&&&\\3&4&&\\1&2&5&6\end{pmatrix},\begin{pmatrix}7&&&\\2&4&&\\1&3&5&6\end{pmatrix},\begin{pmatrix}6&&&\\5&7&&\\1&2&3&4\end{pmatrix}$$

$$\begin{pmatrix}6&&&\\4&7&&\\1&2&3&5\end{pmatrix},\begin{pmatrix}6&&&\\3&7&&\\1&2&4&5\end{pmatrix},\begin{pmatrix}6&&&\\2&7&&\\1&3&4&5\end{pmatrix},\begin{pmatrix}5&&&\\4&7&&\\1&2&3&6\end{pmatrix},\begin{pmatrix}5&&&\\3&7&&\\1&2&4&6\end{pmatrix}$$

$$\begin{pmatrix}5&&&\\2&7&&\\1&3&4&6\end{pmatrix},\begin{pmatrix}4&&&\\3&7&&\\1&2&5&6\end{pmatrix},\begin{pmatrix}4&&&\\2&7&&\\1&3&5&6\end{pmatrix},\begin{pmatrix}3&&&\\2&7&&\\1&4&5&6\end{pmatrix},\begin{pmatrix}6&&&\\4&5&&\\1&2&3&7\end{pmatrix}$$

$$\begin{pmatrix}6&&&\\3&5&&\\1&2&4&7\end{pmatrix},\begin{pmatrix}6&&&\\2&5&&\\1&3&4&7\end{pmatrix},\begin{pmatrix}6&&&\\3&4&&\\1&2&5&7\end{pmatrix},\begin{pmatrix}6&&&\\2&4&&\\1&3&5&7\end{pmatrix},\begin{pmatrix}5&&&\\4&6&&\\1&2&3&7\end{pmatrix}$$

$$\begin{pmatrix}5&&&\\3&6&&\\1&2&4&7\end{pmatrix},\begin{pmatrix}5&&&\\2&6&&\\1&3&4&7\end{pmatrix},\begin{pmatrix}4&&&\\3&6&&\\1&2&5&7\end{pmatrix},\begin{pmatrix}4&&&\\2&6&&\\1&3&5&7\end{pmatrix},\begin{pmatrix}3&&&\\2&6&&\\1&4&5&7\end{pmatrix}$$

$$\begin{pmatrix}5&&&\\3&4&&\\1&2&6&7\end{pmatrix},\begin{pmatrix}5&&&\\2&4&&\\1&3&6&7\end{pmatrix},\begin{pmatrix}4&&&\\3&5&&\\1&2&6&7\end{pmatrix},\begin{pmatrix}4&&&\\2&5&&\\1&3&6&7\end{pmatrix},\begin{pmatrix}3&&&\\2&5&&\\1&4&6&7\end{pmatrix}$$

But this is not so attractive too because tableaux are not represented as matrices in your articles. The best thing would be to write TEX macros to dump tableaux:

```
TabMacros:=proc()
  TexStr(`\\def\\boxit#1#2{\\setbox1=\\hbox{\\kern#1{#2}\\kern#1}`);
  TexStr(`\\dimen1=\\ht1 \\advance\\dimen1 by #1 \\dimen2=\\dp1 \\advance\\dimen2 by #1`);
  TexStr(`\\setbox1=\\hbox{\\vrule height\\dimen1 depth\\dimen2\\box1\\vrule}`);
  TexStr(`\\setbox1=\\vbox{\\hrule\\box1\\hrule}`);
  TexStr(`\\advance\\dimen1 by .4pt \\ht1=\\dimen1 `);
  TexStr(`\\advance\\dimen2 by .4pt \\dp1=\\dimen2  \\box1\\relax}`);
  TexStr(`\\catcode``\\@=12`);
  TexStr(`\\newdimen\\squaresize \\squaresize=14pt`);
  TexStr(`\\newdimen\\thickness \\thickness=0.5pt`);
  TexStr(`\\def\\square#1{\\hbox{\\vrule width \\thickness`);
  TexStr(`\\vbox to \\squaresize{\\hrule height \\thickness\\vss`);
  TexStr(`\\hbox to \\squaresize{\\hss#1\\hss}`);
  TexStr(`\\vss\\hrule height\\thickness}`);
  TexStr(`\\unskip\\vrule width \\thickness}\\kern-\\thickness}`);
  TexStr(`\\def\\vsquare#1{\\vbox{\\square{$#1$}}\\kern-\\thickness}`);
  TexStr(`\\def\\blank{\\omit\\hskip\\squaresize}`);
  TexStr(`\\def\\tableau#1{`);
  TexStr(`\\vbox{\\smallskip\\offinterlineskip`);
  TexStr(`\\halign{&\\vsquare{##}\\cr #1}}}`);
end:
```

and then to have a function to dump a tableau:

```
`TeX/UserdefinedFun/Tab`:=proc()
  local
    tab,  # the argument...
    i, j; # variables for loops...
  tab:=args[1];
  TexCat(`\\tableau{`);
  for i from 1 to nops(tab) do
    for j from 1 to nops(op(i, tab))-1 do
      Texer(op(j, op(i, tab)));
      TexCat(` & `);
    od;
    Texer(op(nops(op(i, tab)), op(i, tab)));
    TexCat(`\\cr`);
  od;
  TexCat(`}\\quad`);
end:
```

Then, we have just to dump the macros by:

```
ACE> TabMacros();
```

and each tableau by:

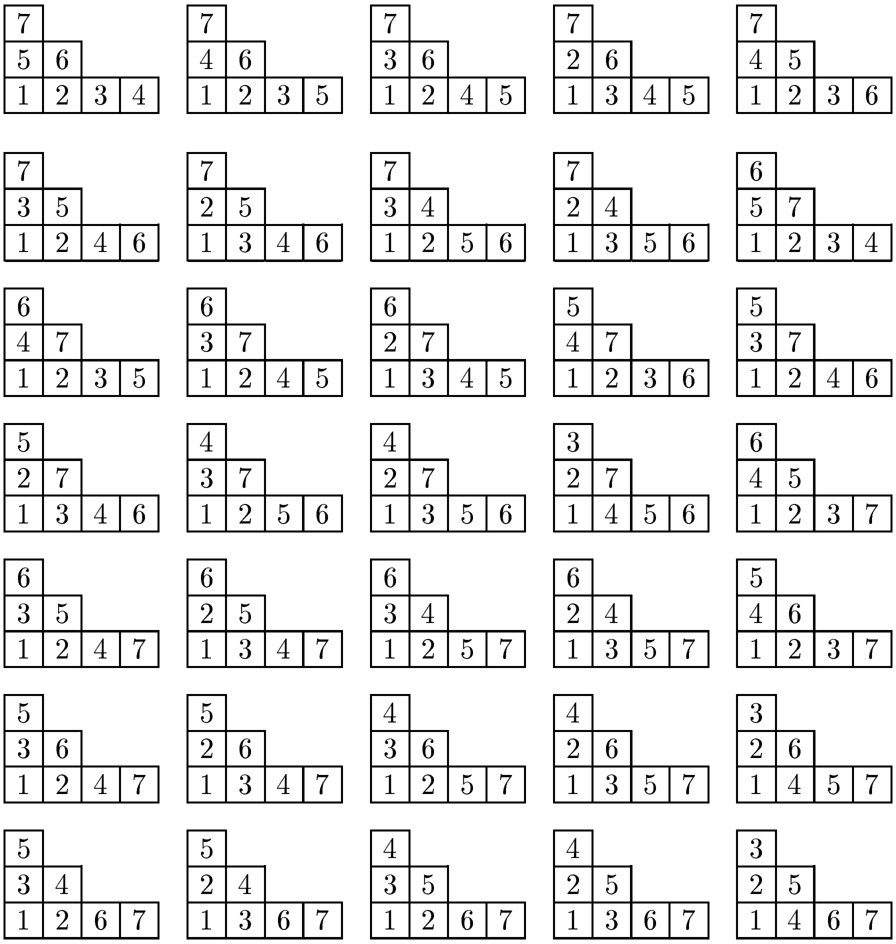
```
ACE> lt := ListTab([4,2,1], [1$7]):
ACE> for i from 1 by 5 to nops(lt) do
       TexStr(`$$`);
       for j from 0 to 4 do
         Tex(Tab(op(i+j, lt)));
       od;
       TexStr(`$$`);
     od:
```

to obtain the following output:

```
\def\boxit#1#2{\setbox1=\hbox{\kern#1{#2}\kern#1}
\dimen1=\ht1 \advance\dimen1 by #1 \dimen2=\dp1 \advance\dimen2 by #1
\setbox1=\hbox{\vrule height\dimen1 depth\dimen2\box1\vrule}
\setbox1=\vbox{\hrule\box1\hrule}
\advance\dimen1 by .4pt \ht1=\dimen1
\advance\dimen2 by .4pt \dp1=\dimen2  \box1\relax}
\catcode`\@=12
\newdimen\squaresize \squaresize=14pt
\newdimen\thickness \thickness=0.5pt
\def\square#1{\hbox{\vrule width \thickness
\vbox to \squaresize{\hrule height \thickness\vss
\hbox to \squaresize{\hss#1\hss}
\vss\hrule height\thickness}
\unskip\vrule width \thickness}\kern-\thickness}
\def\vsquare#1{\vbox{\square{$#1$}}\kern-\thickness}
\def\blank{\omit\hskip\squaresize}
\def\tableau#1{
\vbox{\smallskip\offinterlineskip
\halign{&\vsquare{##}\cr #1}}}
$$
\tableau{7\cr5 & 6\cr1 & 2 & 3 & 4\cr}\quad \tableau{7\cr4 & 6\cr1 & 2 & 3 & 5\cr}\quad
\tableau{7\cr3 & 6\cr1 & 2 & 4 & 5\cr}\quad \tableau{7\cr2 & 6\cr1 & 3 & 4 & 5\cr}\quad
\tableau{7\cr4 & 5\cr1 & 2 & 3 & 6\cr}\quad
$$
$$
\tableau{7\cr3 & 5\cr1 & 2 & 4 & 6\cr}\quad \tableau{7\cr2 & 5\cr1 & 3 & 4 & 6\cr}\quad
\tableau{7\cr3 & 4\cr1 & 2 & 5 & 6\cr}\quad \tableau{7\cr2 & 4\cr1 & 3 & 5 & 6\cr}\quad
\tableau{6\cr5 & 7\cr1 & 2 & 3 & 4\cr}\quad
$$
$$
\tableau{6\cr4 & 7\cr1 & 2 & 3 & 5\cr}\quad \tableau{6\cr3 & 7\cr1 & 2 & 4 & 5\cr}\quad
\tableau{6\cr2 & 7\cr1 & 3 & 4 & 5\cr}\quad \tableau{5\cr4 & 7\cr1 & 2 & 3 & 6\cr}\quad
\tableau{5\cr3 & 7\cr1 & 2 & 4 & 6\cr}\quad
$$
$$
\tableau{5\cr2 & 7\cr1 & 3 & 4 & 6\cr}\quad \tableau{4\cr3 & 7\cr1 & 2 & 5 & 6\cr}\quad
\tableau{4\cr2 & 7\cr1 & 3 & 5 & 6\cr}\quad \tableau{3\cr2 & 7\cr1 & 4 & 5 & 6\cr}\quad
```

```
\tableau{6\cr4 & 5\cr1 & 2 & 3 & 7\cr}\quad
$$
$$
\tableau{6\cr3 & 5\cr1 & 2 & 4 & 7\cr}\quad \tableau{6\cr2 & 5\cr1 & 3 & 4 & 7\cr}\quad
\tableau{6\cr3 & 4\cr1 & 2 & 5 & 7\cr}\quad \tableau{6\cr2 & 4\cr1 & 3 & 5 & 7\cr}\quad
\tableau{5\cr4 & 6\cr1 & 2 & 3 & 7\cr}\quad
$$
$$
\tableau{5\cr3 & 6\cr1 & 2 & 4 & 7\cr}\quad \tableau{5\cr2 & 6\cr1 & 3 & 4 & 7\cr}\quad
\tableau{4\cr3 & 6\cr1 & 2 & 5 & 7\cr}\quad \tableau{4\cr2 & 6\cr1 & 3 & 5 & 7\cr}\quad
\tableau{3\cr2 & 6\cr1 & 4 & 5 & 7\cr}\quad
$$
$$
\tableau{5\cr3 & 4\cr1 & 2 & 6 & 7\cr}\quad \tableau{5\cr2 & 4\cr1 & 3 & 6 & 7\cr}\quad
\tableau{4\cr3 & 5\cr1 & 2 & 6 & 7\cr}\quad \tableau{4\cr2 & 5\cr1 & 3 & 6 & 7\cr}\quad
\tableau{3\cr2 & 5\cr1 & 4 & 6 & 7\cr}\quad
$$
```

which gives:



that seems to be more convenient!