MuPAD-Combinat

http://mupad-combinat.sf.net/

Coordonnateur : Nicolas M. Thiéry, LaPCS, Univ. Lyon I,
membre associé de l'IGM

Pourquoi participer au développement d'un logiciel de Calcul Symbolique ?

- *La recherche en combinatoire est indissociable de l'exploration informatique ; celle-ci requiers le développement et l'utilisation d'outils de calcul algébriques souples et performants.*

- *Une fois développés, ces outils sont valorisés par leur mise à disposition de la communauté ; celle-ci en bénéficie tant pour effectuer de nouvelles recherches que pour l'enseignement.*

- *Réciproquement cette démarche augmente notre visibilité, et permet de présenter les résultats de notre recherche à d'autres domaines scientifiques en prise avec des calculs semblables.*

Pourquoi MuPAD ?

- *Le noyau et le langage du Système de Calcul Formel MuPAD (http://www.mupad.de) sont construits sur des bases logiques et réfléchies permettant un développement plus facile.*

- *La collaboration avec le groupe MuPAD repose sur des engagements mutuels satisfaisants :*
  - *compatibilité ascendante, stabilité dans le temps (de MuPAD),*
  - *support technique, ajouts de fonctionnalités,*
  - *système raisonnablement ouvert à défaut d'être libre.*

Description

MuPAD-Combinat est une bibliothèque de combinatoire algébrique pour le système de calcul formel MuPAD. Elle fournit aux chercheurs une boîte à outils extensible pour l'exploration informatique. L'objectif est de fédérer les développements logiciels en combinatoire algébrique, via une approche logiciel libre.

État de l'art (MuPAD-Combinat 1.1.0, novembre 2003)

- *Version stable ; architecture générale du projet en place.*
- *Bibliothèques pour les classes combinatoires usuelles : permutations, tableaux, arbres, objets décomposables, ...*
- *Outils pour définir de nouvelles classes combinatoires.*
- *Exemples d'algèbres (de Hopf) combinatoires : fonctions symétriques et généralisations, algèbres de Hecke, Loday-Ronco, Steenrod, Weyl, polynômes de Schubert, invariants de groupes de permutations, ...*
- *Outils pour définir de nouvelles algèbres combinatoires.*
- *Bibliothèque pour les automates à poids.*

Quelques chiffres

- *Né en décembre 2001 (fusion de ACE (IGM), mu-EC (IGM), CS (Loria Nancy, LaBRI Bordeaux, LRI Orsay), PerMuVAR (LaPCS) et bibliothèque combinat de MuPAD).*
- *Bibliothèque officielle de combinatoire de MuPAD depuis janvier 2002.*
- *3 développeurs, environ 20 contributeurs dont 3 principaux.*
- *Laboratoires impliqués: IGM (Marne-la-vallée), LaPCS (UCB Lyon I), LIFAR (Rouen), ...*
- *50000 lignes de code de haut niveau, 15000 lignes de C++, 80000 lignes de C importées de Symmetrica.*
- *450 pages de documentation, 9500 lignes de tests, 500 messages / an sur la liste de diffusion.*

Example: Free Symmetric Functions

Implementing Free Symmetric Functions

```
/**************************************************************
   Free Littlewood-Richardson rule:
   The product of two tableaux t1 and t2 is defined as the sum
   of the tableaux appearing in the shuffle of t1 with the
   plactic class of t2 shifted by the size of t1.
**************************************************************/

// Let us define some Shortcuts
alias(tableaux=combinat::tableaux):
alias(words=combinat::words):

placticClasstableaux :=
// Compute the plactic class of a standard tableaux
proc(tab : tableaux)
begin
    map(tableaux::list(tableaux::shape(tab)),
        z -> tableaux::invSchensted(tab, z));
end_proc:

isWordStandardTableaux :=
// Check if a word is the natural row-reading
// of a standard tableaux
proc(w : words)
begin
    testtype(tableaux::fromWord(w), tableaux::typeStandard);
end_proc:
```

```
FreeLR :=
proc (t1 : tableaux::typeStandard, t2 : tableaux::typeStandard)
    local m1, t2shifted, k, res;
begin
    m1        := tableaux::toWord(t1);
    k         := nops(m1);
    t2shifted := map(t2, map, z -> z+k);
    res       := [ ];
    for i in placticClassTab(t2shifted) do
        res := append(res,
                      op(select(words::shuffle::list(m1, i),
                                isWordStandardTableaux)));
    end_for;
    map(res, z -> tableaux::insertWord([], z));
end_proc:

domain FreeSymmetricFunctions(R)
    inherits Dom::FreeModule(R, tableaux);
    category Cat::AlgebraWithBasis(R);

    printTerm := tableaux::prettyPrint;
    one := dom::term([]);
    mult2Basis := (t1,t2) ->
      dom::plus(dom::term(t) $ t in FreeLR(t1,t2));
end_domain:
```

Using Free Symmetric Functions

```
// Instantiating the algebra of Free Symmetric Functions over the field of expressions
>> FS := FreeSymmetricFunctions(Dom::ExpressionField()):

// Some sample computations
>> x := FS([[1]])^2;
                                   +---+
                                   | 2 |   +---+---+
                                   +---+ + | 1 | 2 |
                                   | 1 |   +---+---+
                                   +---+
                                                              Time: 20 msec

>> y := a*FS([[2],[1]]) - b*FS([[1,2]]);
                                  +---+
                                  | 2 |       +---+---+
                                a +---+ + (-b) | 1 | 2 |
                                  | 1 |       +---+---+
                                  +---+
                                                              Time: 30 msec

>> x * y;

                                                                    +---+                            +---+
     +---+---+        +---+                                         | 3 |            +---+           | 4 |
     | 3 | 4 |        | 3 |                +---+---+---+---+        +---+            | 2 |           +---+
(-b) +---+---+ + (-b) +---+---+---+ + (-b) | 1 | 2 | 3 | 4 | + (-b) | 2 |     + (-b) +---+---+---+ + a | 3 |     +
     | 1 | 2 |        | 1 | 2 | 4 |        +---+---+---+---+        +---+---+        | 1 | 3 | 4 |     +---+---+
     +---+---+        +---+---+---+                                 | 1 | 4 |        +---+---+---+     | 1 | 2 |
                                                                    +---+---+                          +---+---+

                    +---+
                    | 4 |     +---+
  +---+             +---+     | 4 |         +---+---+
  | 4 |             | 3 |     +---+         | 2 | 4 |
a +---+---+---+ + a +---+ + a | 2 |     + a +---+---+
  | 1 | 2 | 3 |     | 2 |     +---+---+     | 1 | 3 |
  +---+---+---+     +---+     | 1 | 3 |     +---+---+
                    | 1 |     +---+---+
                    +---+
                                                              Time: 120 msec
```

The morphism from Free Symmetric Functions to Symmetric Functions

```
// The algebra of symmetric functions
>> Sym := examples::SymmetricFunctions():

// Implementing the morphism:
//  -A tableau is sent to the Schur function indexed by the corresponding partition
//  - tableaux::shape computes the shape of a tableau
//  - Sym::s::term creates a Schur function from a partition
>> FS2Sym := operators::makeLinear(Sym::s::term @ tableaux::shape, Source=FS, ImageSet=Sym::s):

// Using the morphism
>> FS([[3],[1,2]])^2;

+---+       +---+
| 6 |       | 6 |             +---+           +---+                                     +---+---+   +---+
+---+       +---+             | 6 |           | 6 |                   +---+---+         | 4 | 6 |   | 4 |
| 4 |       | 4 |             +---+---+       +---+                   | 3 | 6 |         +---+---+   +---+---+
+---+---+ + +---+         + | 3 | 4 |     + | 3 |             + +---+---+---+---+ + | 3 | 5 | + | 3 | 6 |     +
| 3 | 5 |   | 3 |             +---+---+---+   +---+---+---+---+   | 1 | 2 | 4 | 5 |   +---+---+   +---+---+---+
+---+---+   +---+---+---+     | 1 | 2 | 5 |   | 1 | 2 | 4 | 5 |   +---+---+---+---+   | 1 | 2 |   | 1 | 2 | 5 |
| 1 | 2 |   | 1 | 2 | 5 |     +---+---+---+   +---+---+---+---+                       +---+---+   +---+---+---+
+---+---+   +---+---+---+

  +---+---+---+
  | 3 | 4 | 6 |
  +---+---+---+
  | 1 | 2 | 5 |
  +---+---+---+
                                                              Time: 510 msec

>> FS2Sym(%);

        s[3, 3] + s[2, 2, 2] + s[4, 2] + s[4, 1, 1] + 2 s[3, 2, 1] + s[3, 1, 1, 1] + s[2, 2, 1, 1]
                                                              Time: 20 msec

// Checking that this is indeed a morphism
>> Sym::s([2,1])^2;

        s[3, 3] + s[2, 2, 2] + s[4, 2] + s[4, 1, 1] + 2 s[3, 2, 1] + s[3, 1, 1, 1] + s[2, 2, 1, 1]
                                                              Time: 80 msec
```

Contact : Teresa.Gomez-Diaz@univ-mlv.fr, Florent.Hivert@univ-mlv.fr,
Jean-Christophe.Novelli@univ-mlv.fr, Nicolas.Thiery@univ-lyon1.fr