Help For:

Schur-Fkt - A Maple Package for the Hopf algebra of symmetric functions

Version 1.0.1 (17 xii 2007) -- designed for Maple 10

(Copyright (c) Rafal Ablamowicz * and Bertfried Fauser§, 2003 - 2007. All rights reserved.)

(*) Department of Mathematics, Box 5054
Tennessee Technological University
Cookeville, TN 38501 USA
Phone: (931) 372-3664, Fax: (931) 372-6353
E-mail: rablamowicz@tntech.edu
URL: http://math.tntech.edu/rafal/SchurFkt/

(§) University of Konstanz
Fachbereich Physik, Fach M678
Universitätsstrasse 10, D-78457 Konstanz, Germany
Phone: +49 (0)7531 693491
E-mail: Bertfried.Fauser@uni-konstanz.de
URL: http://clifford.physik.uni-konstanz.de/~fauser/

Last revised: December 17, 2007 (BF & RA)

Calling Sequence:
function(args) (if the package was loaded using with(SchurFkt); )
SchurFkt[function](args) (long form without loading the package)

Note:
SchurFkt _needs_ the package define of the Clifford/Bigebra packages, since it defined tensor
products of symmetric functions. It also from
time to time needs Clifford/Bigebra, so we advice strongly to install the library file which contains
all of these packages!

Description:

- SchurFkt provides essential operations on the Hopf algebra of (commutative) symmetric functions
  in formally infinite many variables. It provides several important bases which allow to implement
  products and coproducts by means of combinatorics of Young diagrams (Ferers diagrams,
  essentially a graphical display of partitions) and Young tableaux.

- Schur polynomials can be used to describe irreducible representations of general linear groups.
The product of these polynomials resembles the Glebsch-Gordan decomposition of a tensor
  product of two irreducible representations (irreps) into irreducibles again. The decomposition of
  irreps gives a coproduct of Schur functions. Schur functions encode a huge number of
combinatorial identities. Schur functions have a second product, called 'inner product'. This product has to do with the product of irreps of the symmetric group. Since it does not add the weight of the tableaux but combines two tableaux of the same weight into other such tableaux.

- Power sum symmetric functions play a role in enumerative combinatorics (Polya counting theory, cycle indicators), as in algebras K-theory (Adams operations). In our setting, power sum function form the primitive elements of the outer Hopf algebra of symmetric functions.

- Monomial symmetric functions play an important role in the approach to symmetric functions proposed by Rota-Stein, 94. Currently SchurFkt does not fully implement these algorithms.

- The **general goal** of SchurFkt is to provide a proof of concept for some new developments in symmetric function and invariant theory. Since Maple (TM) is considerably slower as, e.g. SCHUR by Brian G Wybourne, serious calculations may need special purpose software. However, being able to code new algorithms provides new insights into the theory and last but not least proves the authors understanding of the subject.

**Load SchurFkt in the following way:**

```maple
restart:with(SchurFkt);
```

**SchurFkt Version 1.0.1 (17 xii 2007) at your service**
(c) 2003-2007 BF&RA, no warranty, no fitness for anything!
Increase verbosity by infolevel[`function`]=val -- use online help > ?Bigebra[help]

[AlexComp, CharHook, CompNM, FLAT, Frob2part, KostkaPC, KostkaTable, LaplaceM, LaplaceM_mon, LaplaceTable, MLIN, MurNak, MurNak2, PartNM, Scalar, ScalarHM, ScalarMH, ScalarP, antipE, antipH, antipM, antipMC, antipP, antipS, branch, cinner, cmp2part, cmp2prtMult, concatM, conjpart, couter, couterE, couterH, couterM, couterON, couterP, cplethP, cplethS, e_to_h, e_to_s, evalJacobiTrudiMatrix, getSfktSeries, grAlexComp, h_to_m, h_to_s, inner, innerH, innerP, isLattice, m_to_p, maxlengthSymFkt, mset2part, outer, outerE, outerH, outerM, outerON, outerP, outerS, p_to_m, p_to_s, part2Frob, part2mset, plethP, plethS, plethSnm, s_to_h, s_to_hJT, s_to_hmat, s_to_p, s_to_x, skew, sq_coeff, truncWT, x_to_s, zee]

**Alphabetic (but Overview as the first topic) listing of available procedures in 'SchurFkt':**

- **Overview** -- The main HELP FILE for SchurFkt (this file)

- **AlexComp** -- compares two compositions/partitions w.r.t. anti-lexicographic ordering

- **antipS** -- the antipode acting on symmetric functions in the Schur polynomial basis
• **branch** -- branch transforms a symmetric function representing a group character into S-functions of another group via branching

• **CharHook** -- evaluation of a cycle indicator on a Hook Schur function

• **cinner** -- inner coproduct of symmetric function the Schur function basis

• **cmp2prtMult** -- computes the length of the orbit of compositions which project under sorting to the same partition

• **CompNM** -- produces a list of compositions of N into M parts

• **concatM** -- divided powers concatenation product (needed for Rota-Stein cliffordization)

• **conipart** -- computes a conjugate partition

• **couter** -- the outer coproduct in the Schur function basis

• **couterE** -- the outer coproduct in the elementary symmetric function basis

• **couterH** -- the outer coproduct in the complete symmetric function basis

• **couterM** -- the outer coproduct in the monomial symmetric function basis

• **couterON** - outer coproduct for the O(n) groups in the stable limit N--> infinity

• **couterP** -- the outer coproduct in the power sum basis

• **cplethP** -- plethysm coproduct in the power sums basis

• **cplethS** -- plethysm coproduct in the Schur function basis

• **FLAT** -- flattens the function T() used by SchurFkt[MLIN] (hence T() is made associative this way), mainly for internal use!

• **Frob2part** -- converts a partition in Frobenius notation into a standard list notation of partitions

• **getSfktSeries** -- produces a Schur function series (or a list of its coefficients)

• **grAlexComp** -- compares two compositions/partitions w.r.t. graded anti-lexicographic ordering

• **h_to_s** -- convert a homogenous symmetric function into a Schur function

• **inner** -- the inner product in Schur function basis

• **innerP** -- inner product in the power sum basis

• **isLattice** -- checks if a Young tableau is a lattice permutation

• **KostkaPC** -- computes the Kostka coefficient between a composition and a partition

• **KostkaTable** -- computes the Kostka matrix in any dimension

• **LaplaceM** -- the Rota-Stein Laplace pairing internally used for 'cliffordization' of the
concatenation product in the monomial basis into the outer product of monomial symmetric functions (internal use mostly)

- **LaplaceM_mon** -- Laplace pairing on monomials

- **LaplaceTable** -- tabulates the LaplaceM pairing of m-function monomials (exhibits some grading properties)

- **m_to_p** -- basis change from monomial to power sum symmetric functions

- **MLIN** -- makes the function $T()$ multilinear over the integers, mainly for internal use!

- **mset2part** -- translates a partition in multiset notation into a partition in standard format

- **MurNak** -- the Murnaghan Nakayama character of the symmetric group, uses internally a rim-hook representation of partitions to optimize the algorithm

- **MurNak2** -- MurNak2 uses a recursive algorithm and is much slower than MurNak (for comparison and educational/demonstration purpose only)

- **outer** -- outer product of two Schur functions (also known as `SchurFkt[outerS]`)

- **outerE** -- outer product in the elementary symmetric function basis (E-basis)

- **outerH** -- outer product in the complete symmetric function basis (H-basis)

- **outerM** -- outer product of monomial symmetric functions (a la Rota-Stein)

- **outerON** -- outer product for orthogonal (symplectic) characters

- **outerP** -- the outer product of symmetric functions in the power sum basis

- **outerS** -- outer product of two Schur functions (same function as `SchurFkt[outerI]` in this version of SchurFkt)

- **p_to_m** -- basis change from power sum to monomial symmetric functions

- **p_to_s** -- basis change from power sum symmetric functions to Schur functions

- **part2Frob** -- translates a standard partition (shape) into Frobenius notation

- **part2mset** -- translates a partition in standard representation into an multiset (exponential) representation

- **PartNM** -- returns a list of partitions of $N$ with parts of size at most $M$

- **plethP** -- plethysm in the power sum basis

- **plethS** -- computes the plethysm of two Schur function polynomials

- **plethSnm** -- computes the plethysm of two $s$functions of the form $s[n]$ (one part complete symmetric functions)

- **s_to_h** -- basis change from Schur function basis to complete symmetric function basis

- **s_to_p** -- basis change from Schur functions to power symmetric functions
• **s_to_x** -- translation of a Schur function into a polynomial in variables \( x_i \)

• **Scalar** -- the Schur-Hall [Redfield cup] scalar product in the Schur function basis

• **ScalarHM** -- the Schur-Hall [Redfield cup] scalar product in the monomial-complete symmetric function basis

• **ScalarMH** -- the Schur-Hall [Redfield cup] scalar product in the complete-monomial symmetric function basis

• **ScalarP** -- the Schur-Hall [Redfield cup] scalar product in the power sum basis

• **skew** -- (outer) skew of two Schur functions

• **sq_coeff** -- returns the square of the coefficients of a Schur function

• **truncWT** -- truncates an Schur function expression by its weight

• **x_to_s** -- translates a polynomial in the variables \( x_i \) into a Schur function expression

• **zee** -- the symmetry factor \( z \) associated to power symmetric functions (or cycles of the symmetric group)

### New Types in 'SchurFkt':

We use 'fkt' derived from German 'Funktion' (function) as in SchurFkt also for types. Typing is necessary to allow Maple(R) to decide about linearity of certain morphisms (procedures). Symmetric functions come with a number of standard bases, which have combinatorially different meanings and allow different algorithms to be used to perform calculations. The SchurFkt package knows currently the following types:

• Schur functions. This is the most important basis. Schur functions (sfunction for short) encode characters of irreducible representations of the symmetric and general linear groups. Schur functions (and all other bases) are indexed by integer partitions, written as index to the kernel-symbol (here 's'). We need to distinguish:

  -- **`type/sfktmonom`** -- Schur function monom, a basis element like \( s[3,2,2,1] \) with no prefactor.
  -- **`type/sfktterm`** -- A Schur function including a coefficient from the ground ring (usually integers) of type **cliscalar** like \( 4*s[4,1,1,1] \)
  -- **`type/sfktpolynom`** -- A linear combination of sfktterms like \( 2*s[2]+5*s[1,1] \).

The types used are inclusive, so a check if an expression <foo> has type sfktpolynomial yields true, if foo is an sffunction of type sfktmonom, sfktterm, or sfktpolynom! The check for an sfktterm yields true, if <foo> is a term of a form coefficient times a Schur function monom or if it is a Schur function monom, while the check for Schur function monom yields true only
for expressions like s[3,3] (irreps, basis monoms of the ring of symmetric functions).

Note: Schur functions are self dual wrt to the Schur-Hall inner product \`ScalarS\'. They form an orthonormal basis.
Note: Schur functions are not multiplicative (see below).

- Power sum symmetric functions have their origin in the invariant theory of the symmetric group. Considering polynomials in the indeterminates \{x_i\}_{i=1}^n it is obvious that the polynomials \( p_k(x) = \sum_{i=0}^n x_i^k \) are invariant under the action of the symmetric group acting on \( n \) letters (indeterminates). Furthermore, these polynomials are a complete set of invariants. Last but not least, the power sum symmetric functions are orthogonal but not normalized w.r.t. the Schur-Hall inner product \`ScalarP\'.

We distinguish in the same fashion as for the S-functions, basis monoms, terms and polynomials in the power sum symmetric functions \( p_k(x) \):

- `type/pfktonom` -- A basis monom like \( p[2,2,1] \).
- `type/pfkterm` -- A basis monom with an optional ring coefficient \( 5*p[2] \) of type \texttt{eliscalar}.
- `type/pfktpolynom` -- A linear combination of pfktterms or a pfktterm or a pfktmonom.

Note: Power sum symmetric functions are multiplicative. That is, the outer product of power sum symmetric functions is the (unordered) concatenation of power sum symmetric functions:

\[
\]

The outer product is particularly simple to compute for multiplicative bases!

- Complete symmetric functions are another special class of symmetric functions. Complete symmetric functions are the dual basis w.r.t. the Schur-Hall inner product of the monomial symmetric functions (see below). They are used to extract counting coefficients in generating functions in the Polya-Redfield theory of enumeration. The classical kernel symbol is \`h\', we distinguish:

- `type/hfktmonom` -- A basis monom like \( h[2,2,1] \).
- `type/hfktterm` -- A basis monom with an optional ring coefficient \( 5*h[2] \) of type \texttt{eliscalar}.
- `type/hfktpolynom` -- A linear combination of hfktterms or a hfktterm or a hfktmonom.

- Monomial symmetric functions are _the_ classical symmetric functions. They are obtained by
symmetrizing monomials \(x^{\alpha_1}\ldots x^{\alpha_k}\) using the symmetric group \(S_k\) acting on the indices of the indeterminates, where only distinct terms are kept (no multiplicities). One has \(m_\lambda = \sum_{\sigma \in S_k \text{ distinct}} x_{\sigma(1)}^{\lambda_1}\ldots x_{\sigma(k)}^{\lambda_k}\). Hence monomial symmetric functions appear by averaging over the symmetric group action on a single monomial. It is clear that a partition \(\lambda\) indexes such averages, while individual monomials are indexed by compositions (ordered integer decompositions). The monomial symmetric function basis is \_not\_ multiplicative. We distinguish:

```
-- `type/mfktmonom` -- A basis monom like m[2,2,1].
-- `type/mfktterm` -- A basis monom with an optional ring coefficient 5*m[2] of type cliscopal,
 but m[3]=1*m[1] is a term either.
-- `type/mfktpolynom` -- A linear combination of mfktterms or a mfktterm or a mfktmonom.
```

Note: The SchurFkt package has a second product employed on the basis of monomial symmetric functions. This is the `concatM` product which establishes the \_multiplicative\_ (unordered) concatenation product. This product is not usually considered in the theory of symmetric functions and is \_not\_ the outer product. However, the process of cliffordization described by Rota-Stein allows one to introduce the outer product in the monomial basis `outerM` as a Hopf algebra deformation of the (unordered) concatenation product `concatM`. (This is in analogy to how a Clifford algebra appears to be a deformation of the Grassmann algebra).

- Elementary symmetric functions are cousins of complete symmetric functions. The are obtained by conjugating the partitions indexing rows and columns of one part partitions and one row partitions. Elementary symmetric functions (while being symmetric functions) encode antisymmetric aspects of invariants. Rows in a Young diagram (tableau) are antisymmetrized. Elementary symmetric functions form a multiplicative basis, and we distinguish:

```
-- `type/efktmonom` -- A basis monom like e[2,2,1].
-- `type/efktterm` -- A basis monom with an optional ring coefficient 5*e[2] of type cliscopal,
-- `type/efktpolynom` -- A linear combination of efktterms or a efktterm or a efktmonom.
```

- The dual basis of the elementary symmetric functions is called forgotten functions (Doubilet functions), since they played a minor (invisible) role in the combinatorial and enumerative approach to invariants and symmetric functions. The forgotten functions share many properties with the monomial symmetric functions, the basis is \_not\_ multiplicative. We distinguish:
-- `type/ffkmonom` -- A basis monom like f[2,2,1].
-- `type/ffktterm` -- A basis monom with an optional ring coefficient 5*f[2] of type cliscalar,
-- `type/ffktpolynom` -- A linear combination of ffktterms or a ffktterm or a ffktmonom.

Note: Not much about forgotten functions is yet implemented in SchurFkt, .... nomen est
omen.

- The general type symfkt[monom|term|polynom] was created to check if a general expression
  <foo> contains any of the above specified bases {s,p,h,m,e,}. This may allow to form
  expressions with mixed basis types like e[3,2,1]+h[3,3] and alike. Some internal functions of
  SchurFkt do not need to know what kind of basis they process unless it is, say, a
  multiplicative one. In order not to trigger a "wrong type" error, type checking is done only
  against symfkt[monom|term|polynom].

  Note: The usage of symfkt-types is _dangerous_ and should be done only in internally used
  functions! Beware!

See Also: define

NOTE: SchurFkt needs the patched define which ships with the Clifford/Bigebra
packages!!

(c) Copyright June 2003 - December 2007, by Bertfried Fauser & Rafal Ablamowicz, all rights
reserved.
Last modified: December 17, 2007/BF/RA