SpinSym

Brauer tables of spin-symmetric groups

Version 1.5.3

26 November 2026

Lukas Maas

Lukas Maas Email: maasluk@gmail.com

Copyright

© 2012 Lukas Maas

The SpinSym GAP package is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program (see the file LICENSE). If not, see https://www.gnu.org/licenses/.

Contents

1 Introduction **4**
- 1.1 The data part . . . 4
- 1.2 The functions part . . . 5
- 1.3 Installation and loading . . . 5

2 Usage and features **6**
- 2.1 Accessing the tables . . . 6
- 2.2 Character parameters . . . 7
- 2.3 Class parameters . . . 9
- 2.4 Young subgroups . . . 12
- 2.5 Class Fusions . . . 14

References **17**

Index **18**

Chapter 1

Introduction

The purpose of this GAP package is to make a collection of p-modular character tables (Brauer tables) of spin-symmetric groups (and some related groups) available in GAP, thereby extending Thomas Breuer's GAP Character Table Library [1]. The SpinSym package is based on [2] which serves as the general reference here. If you are interested in computing with SpinSym I would like to refer you to [2] for further references and a more thorough description of some of the topics below. And, of course, I would like to hear from you about more or less successful attempts in using the present functionalities.

The term 'spin-symmetric' refers to the groups

$$\tilde{S}_n = \langle\, z, t_1, \ldots, t_{n-1} \;:\; z^2 = 1,\; t_i^2 = (t_i t_{i+1})^3 = z,\; (t_j t_k)^2 = z \,\rangle$$

and

$$\hat{S}_n = \langle\, z, t_1, \ldots, t_{n-1} \;:\; z^2 = 1,\; t_i^2 = (t_i t_{i+1})^3 = 1,\; (t_j t_k)^2 = z,\; z t_i = t_i z \,\rangle$$

where the relations are imposed for all admissible i, j, k with $|j-k| > 1$. Provided $n \geq 4$, these groups are double covers of the symmetric group S_n on n letters. Although $\tilde{S}_n$ and $\hat{S}_n$ are non-isomorphic groups for $n \neq 6$, they are isoclinic and their representation theory is very similar. By *choice*, we restrict the attention to $\tilde{S}_n$. (However, if you are interested in character tables of $\hat{S}_n$ then have a look at `CharacterTableIsoclinic()` in the GAP Reference Manual.)

The natural epimorphism $\pi : \tilde{S}_n \to S_n,\; t_i \mapsto (i, i+1)$, whose kernel is generated by the central involution z, gives rise to the double cover $\tilde{A}_n = A_n^{\pi^{-1}}$ of the alternating group A_n as the preimage of A_n under π. Irreducible faithful representations of $\tilde{S}_n$ or $\tilde{A}_n$ are called spin representations and a similar 'spin' terminology is used for all related faithful objects, to set them apart from the non-faithful objects that belong essentially to S_n or A_n , respectively.

1.1 The data part

The package contains complete Brauer tables of $\tilde{S}_n$ and $\tilde{A}_n$ up to degree $n = 18$ in characteristic $p = 3, 5, 7$. Thus it includes the corresponding Brauer tables of S_n and A_n . Moreover, Brauer tables of S_n and A_n up to degree $n = 19$ in characteristic $p = 2$ are part of the package too.

Every Brauer table comes with lists of character parameters (row labels) and class parameters (column labels), see 2.2 and 2.3. I would like to mention that only some of the data is 'new', large portions date back to the work of James, Morris, Yaseen, and the Modular Atlas Project. Detailed

references are to be found in [2]. The 2-modular tables of S_n and A_n for $n = 18, 19$ were computed jointly by Jürgen Müller and the author.

Please note that some of our Brauer tables differ to some extent from those contained in the GAP Character Table Library [1] (for example, in terms of the ordering of conjugacy classes and characters or in terms of their parameters). Therefore it seemed appropriate to collect these tables in their own package - so here we are.

I'm grateful to Thomas Breuer for supporting the idea of writing this package and for converting my tables into the right GAP Character Table Library format.

1.2 The functions part

Besides Brauer tables, the package provides some related functionalities such as functions that determine class fusions of subgroup character tables and functions that compute character tables of some Young subgroups of $\tilde{S}_n$.

1.3 Installation and loading

To install this package, download the latest archive file `spinsym-X.Y.Z.tar.gz` from https://github.com/gap-packages/spinsym/releases and unpack it inside the `pkg` subdirectory of your GAP installation. It creates a subdirectory called `spinsym`. Then load the package using the `LoadPackage` command.

Example

```
gap> LoadPackage("spinsym");
```

The SpinSym package banner should appear on the screen. You may want to run a quick test of the installation:

Example

```
gap> dir:= DirectoriesPackageLibrary( "spinsym", "tst" )[1];;
gap> tst:= Filename( dir, "testall.tst" );;
gap> Test( tst );
true
```

Chapter 2

Usage and features

2.1 Accessing the tables

All Brauer tables in this package are relative to a *generic* ordinary character table obtained by one of the following constructions

`CharacterTable( "2.Sym(n)" )`, the character table of $\tilde{S}_n$,

`CharacterTable( "2.Alt(n)" )`, the character table of $\tilde{A}_n$,

`CharacterTable( "Sym(n)" )`, the character table of S_n,

`CharacterTable( "Alt(n)" )`, the character table of A_n.

Note that these are synonymous expressions for

```
CharacterTable( "DoubleCoverSymmetric", n ),
CharacterTable( "DoubleCoverAlternating", n ),
CharacterTable( "Symmetric", n ),
CharacterTable( "Alternating", n ),
```

respectively. More detailed information on these tables is to be found in [3]. In this manual, we call such a character table an (ordinary) *SpinSym table*. If `ordtbl` is an ordinary SpinSym table, the relative Brauer table in characteristic `p` can be accessed using the `mod`-operator (i.e. `ordtbl mod p;`). Such a Brauer table is called a (p-modular) *SpinSym table* in the following.

Example

```
gap> ordtbl:= CharacterTable( "2.Sym(18)" );
CharacterTable( "2.Sym(18)" )
gap> modtbl:= ordtbl mod 3;
BrauerTable( "2.Sym(18)", 3 )
gap> OrdinaryCharacterTable(modtbl)=ordtbl;
true
```

2.2 Character parameters

An ordinary SpinSym table has character parameters, that is, a list of suitable labels corresponding to the rows of `ordtbl` and therefore the irreducible ordinary characters of the underlying group. See `CharacterParameters()` in the GAP Reference Manual.

2.2.1 Parameters of ordinary characters

In the following, 'ordinary (spin) character' is used synonymously for 'irreducible ordinary (spin) character'. It is well known that there is a bijection between the set of ordinary characters of S_n and the set $P(n)$ of all partitions of n. Recall that a partition of a natural number n is a list of non-increasing positive integers (its *parts*) that sum up to n. In this way, every ordinary character χ of S_n has a label of the form `[1,c]` where `c` is a partition of n. The labels of the ordinary characters of A_n are induced by Clifford theory as follows. Either the restriction $\psi = \chi|_{A_n}$ of χ to A_n is an ordinary character of A_n, or ψ decomposes as the sum of two distinct ordinary characters ψ_1 and ψ_2.

In the first case there is another ordinary character of S_n, say ξ labelled by `[1,d]`, such that the restriction of ξ to A_n is equal to ψ. Moreover, the induced character of S_n obtained from ψ decomposes as the sum of χ and ξ. Then ψ is labelled by `[1,c]` or `[1,d]`.

In the second case, both ψ_1 and ψ_2 induce irreducibly up to χ. Then ψ_1 and ψ_2 are labelled by `[1,[c,'+']]` and `[1,[c,'-']]`.

Example

```
gap> ctS:= CharacterTable( "Sym(5)" );;
gap> CharacterParameters(ctS);
[ [ 1, [ 1, 1, 1, 1, 1 ] ], [ 1, [ 2, 1, 1, 1 ] ], [ 1, [ 2, 2, 1 ] ],
  [ 1, [ 3, 1, 1 ] ], [ 1, [ 3, 2 ] ], [ 1, [ 4, 1 ] ], [ 1, [ 5 ] ] ]
gap> ctA:= CharacterTable( "Alt(5)" );;
gap> CharacterParameters(ctA);
[ [ 1, [ 1, 1, 1, 1, 1 ] ], [ 1, [ 2, 1, 1, 1 ] ], [ 1, [ 2, 2, 1 ] ],
  [ 1, [ [ 3, 1, 1 ], '+' ] ], [ 1, [ [ 3, 1, 1 ], '-' ] ] ]
gap> chi:= Irr(ctS)[1];;
gap> psi:= RestrictedClassFunction(chi,ctA);;
gap> Position(Irr(ctA),psi);
1
gap> xi:= Irr(ctS)[7];;
gap> RestrictedClassFunction(xi,ctA) = psi;
true
gap> InducedClassFunction(psi,ctS) = chi + xi;
true
gap> chi:= Irr(ctS)[4];;
gap> psi:= RestrictedClassFunction(chi,ctA);;
gap> psi1:= Irr(ctA)[4];; psi2:= Irr(ctA)[5];;
gap> psi = psi1 + psi2;
true
gap> InducedClassFunction(psi1,ctS) = chi;
true
gap> InducedClassFunction(psi2,ctS) = chi;
true
```

If χ is an ordinary character of $\tilde{S}_n$ or $\tilde{A}_n$, then $\chi(z) = \chi(1)$ or $\chi(z) = -\chi(1)$. If $\chi(z) = \chi(1)$, then χ is obtained by inflation (along the central subgroup generated by z) from an ordinary character of S_n or

A_n, respectively, whose label is given to χ. Otherwise, if χ is a spin character, that is $\chi(z) = -\chi(1)$, then its label is described next.

The set of ordinary spin characters of $\tilde{S}_n$ is parameterized by the subset $D(n)$ of $P(n)$ of all distinct-parts partitions of n (also called bar partitions). If `c` is an even distinct-parts partition of n, then there is a unique ordinary spin character of $\tilde{S}_n$ that is labelled by `[2,c]`. In contrast, if `c` is an odd distinct-parts partition of n, then there are two distinct ordinary spin characters of $\tilde{S}_n$ that are labelled by `[2,[c,'+']]` and `[2,[c,'-']]`. Now the labels of the ordinary spin characters of $\tilde{A}_n$ follow from the labels of $\tilde{S}_n$ in the same way as those of A_n follow from the labels of S_n (see the beginning of this subsection 2.2.1).

Example

```
gap> ctS:= CharacterTable( "Sym(5)" );;
gap> ct2S:= CharacterTable( "2.Sym(5)" );;
gap> ch:= CharacterParameters(ct2S);
[ [ 1, [ 1, 1, 1, 1, 1 ] ], [ 1, [ 2, 1, 1, 1 ] ], [ 1, [ 2, 2, 1 ] ],
  [ 1, [ 3, 1, 1 ] ], [ 1, [ 3, 2 ] ], [ 1, [ 4, 1 ] ], [ 1, [ 5 ] ],
  [ 2, [ [ 3, 2 ], '+' ] ], [ 2, [ [ 3, 2 ], '-' ] ],
  [ 2, [ [ 4, 1 ], '+' ] ], [ 2, [ [ 4, 1 ], '-' ] ], [ 2, [ 5 ] ] ]
gap> pos:= Positions( List(ch, x-> x[1]), 1 );;
gap> RestrictedClassFunctions( Irr(ctS), ct2S ) = Irr(ct2S){pos}; #inflation
true
gap> ct2A:= CharacterTable( "2.Alt(5)" );;
gap> CharacterParameters(ct2A);
[ [ 1, [ 1, 1, 1, 1, 1 ] ], [ 1, [ 2, 1, 1, 1 ] ], [ 1, [ 2, 2, 1 ] ],
  [ 1, [ [ 3, 1, 1 ], '+' ] ], [ 1, [ [ 3, 1, 1 ], '-' ] ], [ 2, [ 3, 2 ] ],
  [ 2, [ 4, 1 ] ], [ 2, [ [ 5 ], '+' ] ], [ 2, [ [ 5 ], '-' ] ] ]
```

2.2.2 Parameters of modular characters

In the following, 'p-modular (spin) character' is used synonymously for 'irreducible p-modular (spin) character'. The set of p-modular characters of S_n is parameterized by the set of all p-regular partitions of n. A partition is p-regular if no part is repeated more than $p-1$ times. Now every p-modular character χ of S_n has a label of the form `[1,c]` where `c` is a p-regular partition of n.

Again, the labels for the p-modular characters of A_n follow from the labels of S_n. However, compared with Subsection 2.2.1, their format is slightly different.

If χ and ξ are distinct p-modular characters of S_n that restrict to the same p-modular character ψ of A_n , then ψ is labelled by `[1,[c,'0']]` where either χ or ξ is labelled by `[1,c]`. If χ is a p-modular character of S_n whose restriction to A_n decomposes as the sum of two distinct p-modular characters, then these are labelled by `[1,[c,'+']]` and `[1,[c,'-']]` where χ is labelled by `[1,c]`.

As in the ordinary case, the set of p-modular characters of $\tilde{S}_n$ is the union of the subset consisting of all inflated p-modular characters of S_n and the subset of spin characters characterized by negative integer values on the central element z. The analogous statement holds for $\tilde{A}_n$. The set of p-modular spin characters of $\tilde{S}_n$ is parameterized by the set of all restricted p-strict partitions of n. A partition is called p-strict if every repeated part is divisible by p, and a p-strict partition λ is restricted if $\lambda_i - \lambda_{i+1} < p$ whenever λ_i is divisible by p, and $\lambda_i - \lambda_{i+1} \leq p$ otherwise for all parts λ_i of λ (where we set $\lambda_{i+1} = 0$ if λ_i is the last part). If `c` is a restricted p-strict partition of n such that n minus the number of parts not divisible by p is even, then there is a unique p-modular spin character of $\tilde{S}_n$ that is labelled by `[2,[c,'0']]`. Its restriction to $\tilde{A}_n$ decomposes as the sum of two distinct p-modular characters which are labelled by `[2,[c,'+']]` and `[2,[c,'-']]`. If n minus the number of parts of

c that are not divisible by p is odd, then there are two distinct p-modular spin characters of $\tilde{S}_n$ that are labelled by [2,[c,'+']] and [2,[c,'-']]. Both of these characters restrict to the same irreducible p-modular spin character of $\tilde{A}_n$ which is labelled by [2,[c,'0']].

Example

```
gap> ctS:= CharacterTable( "Sym(5)" ) mod 3;;
gap> ct2S:= CharacterTable( "2.Sym(5)" ) mod 3;;
gap> ch:= CharacterParameters(ct2S);
[ [ 1, [ 5 ] ], [ 1, [ 4, 1 ] ], [ 1, [ 3, 2 ] ],
  [ 1, [ 3, 1, 1 ] ], [ 1, [ 2, 2, 1 ] ],
  [ 2, [ [ 4, 1 ], '+' ] ], [ 2, [ [ 4, 1 ], '-' ] ],
  [ 2, [ [ 3, 2 ], '0' ] ] ]
gap> pos:= Positions( List(ch, x-> x[1]), 1 );;
gap> RestrictedClassFunctions( Irr(ctS), ct2S ) = Irr(ct2S){pos}; #inflation
true
gap> ct2A:= CharacterTable( "2.Alt(5)" ) mod 3;;
gap> CharacterParameters(ct2A);
[ [ 1, [ [ 5 ], '0' ] ], [ 1, [ [ 4, 1 ], '0' ] ],
  [ 1, [ [ 3, 1, 1 ], '+' ] ], [ 1, [ [ 3, 1, 1 ], '-' ] ],
  [ 2, [ [ 4, 1 ], '0' ] ], [ 2, [ [ 3, 2 ], '+' ] ], [ 2, [ [ 3, 2 ], '-' ] ] ]
```

2.3 Class parameters

Let ct be an ordinary SpinSym table. Then ct has a list of class parameters, that is, a list of suitable labels corresponding to the columns of ct and therefore the conjugacy classes of the underlying group. See ClassParameters() in the GAP Reference Manual. If bt is a Brauer table in characteristic p relative to ct, its class parameters are inherited from ct in correspondence with the p-regular conjugacy classes of the underlying group.

Let $P(n)$ denote the set of partitions of n.

The conjugacy classes of S_n are naturally parameterized by the cycle types of their elements, and each cycle type corresponds to a partition of n. Therefore a conjugacy class C of S_n is characterized by its *type* $c \in P(n)$. The corresponding entry in the list of class parameters is [1,c]. Assume that $C \subset A_n$. Then C is also a conjugacy class of A_n if and only if not all parts of c are odd and pairwise distinct. Otherwise, C splits as the union of two distinct A_n -classes of the same size, C^+ of type c^+ and C^- of type c^-. The corresponding entries in the list of class parameters are [1,[c,'+']] and [1,[c,'-']], respectively.

Furthermore, $\tilde{C} = C^{\pi^{-1}} \subset \tilde{S}_n$ is either a conjugacy class of $\tilde{S}_n$ of type c with class parameter [1,c], or $\tilde{C}$ splits as the union of two distinct $\tilde{S}_n$-classes $\tilde{C}_1$ and $\tilde{C}_2 = z\tilde{C}_1$, both of type c with corresponding class parameters [1,c] and [2,c], respectively. An analogous description applies for the conjugacy classes of $\tilde{A}_n$.

Example

```
gap> ct:= CharacterTable( "Sym(3)" );;
gap> ClassParameters(ct);
[ [ 1, [ 1, 1, 1 ] ], [ 1, [ 2, 1 ] ], [ 1, [ 3 ] ] ]
gap> ct:= CharacterTable( "Alt(3)" );;
gap> ClassParameters(ct);
[ [ 1, [ 1, 1, 1 ] ], [ 1, [ [ 3 ], '+' ] ], [ 1, [ [ 3 ], '-' ] ] ]
gap> ct:= CharacterTable( "2.Sym(3)" );;
gap> ClassParameters(ct);
[ [ 1, [ 1, 1, 1 ] ], [ 2, [ 1, 1, 1 ] ], [ 1, [ 2, 1 ] ], [ 2, [ 2, 1 ] ],
```

```
    [ 1, [ 3 ] ], [ 2, [ 3 ] ] ]
gap> ct:= CharacterTable( "2.Alt(3)" );;
gap> ClassParameters(ct);
[ [ 1, [ 1, 1, 1 ] ], [ 2, [ 1, 1, 1 ] ],
  [ 1, [ [ 3 ], '+' ] ], [ 2, [ [ 3 ], '+' ] ],
  [ 1, [ [ 3 ], '-' ] ], [ 2, [ [ 3 ], '-' ] ] ]
```

To each conjugacy class of $\tilde{S}_n$ or $\tilde{A}_n$ a certain standard representative is assigned in the following way. Let $c = [c_1, c_2, \ldots, c_m]$ be a partition of n. We set $d_1 = 0$, $d_i = c_1 + \ldots + c_{i-1}$ for $i \geq 2$, and

$$t(c_i, d_i) = t_{d_i+1} t_{d_i+2} \ldots t_{d_i+c_i-1}$$

for $1 \leq i \leq m-1$, where $t(c_i, d_i) = 1$ if $c_i = 1$. The *standard representative of type c* is defined as

$$t_c = t(c_1, d_1) t(c_2, d_2) \cdots t(c_{m-1}, d_{m-1}).$$

Furthermore, we define the standard representatives of type c^+ =`[c,'+']` and c^- =`[c,'-']` to be $t_{c^+} = t_c$ and $t_{c^-} = t_1^{-1} t_c t_1$, respectively.

For example, the standard representative of type $c = [7,4,3,1] \in P(15)$ is

$$t_c = t_1 t_2 t_3 t_4 t_5 t_6 t_8 t_9 t_{10} t_{12} t_{13}.$$

Now $\tilde{C}$ is a conjugacy class of $\tilde{S}_n$ or $\tilde{A}_n$ with parameter

`[1,c]` if and only if $t_c \in \tilde{C}$,

`[2,c]` if and only if $z t_c \in \tilde{C}$,

`[1,[c,'+']]` if and only if $t_{c^+} \in \tilde{C}$,

`[2,[c,'+']]` if and only if $z t_{c^+} \in \tilde{C}$,

`[1,[c,'-']]` if and only if $t_{c^-} \in \tilde{C}$,

`[2,[c,'-']]` if and only if $z t_{c^-} \in \tilde{C}$.

2.3.1 SpinSymStandardRepresentative

▷ `SpinSymStandardRepresentative(c, rep)` (function)

Returns: the image of the standard representative of type `c` under a given $\tilde{S}_n$ -representation.

Expecting the second entry of a class parameter of $\tilde{S}_n$ or $\tilde{A}_n$, say `c`, the standard representative of type `c` under a given representation of $\tilde{S}_n$ is computed. The argument `rep` is assumed to be a list $[t_1^R, t_2^R, \ldots, t_{n-1}^R]$ given by the images of the generators $t_1, \ldots, t_{n-1}$ of $\tilde{S}_n$ under a (not necessarily faithful) representation R of $\tilde{S}_n$.

Example

```
gap> ct:= CharacterTable("2.Sym(15)") mod 5;;
gap> cl:= ClassParameters(ct)[99];
[ 1, [ 7, 4, 3, 1 ] ]
gap> c:= cl[2];;
gap> rep:= BasicSpinRepresentationOfSymmetricGroup(15,5);;
gap> t:= SpinSymStandardRepresentative(c,rep);
< immutable compressed matrix 64x64 over GF(25) >
```

```
gap> OrdersClassRepresentatives(ct)[99];
168
gap> Order(t);
168
gap> BrauerCharacterValue(t);
0
```

2.3.2 SpinSymStandardRepresentativeImage

▷ `SpinSymStandardRepresentativeImage(c[, j])` (function)

Returns: the image of the standard representative of type `c` under the natural epimorphism $\pi : \tilde{S}_{\{j,\ldots,j+n-1\}} \to S_{\{j,\ldots,j+n-1\}}$.

Given the second entry `c` of a class parameter of $\tilde{S}_n$ or $\tilde{A}_n$, and optionally a positive integer `j`, the image of the standard representative of type `c` under $\pi : \tilde{S}_{\{j,\ldots,j+n-1\}} \to S_{\{j,\ldots,j+n-1\}}$ with $t_i^\pi = (i, i+1)$ for $j \leq i \leq j+n-2$ is computed by calling `SpinSymStandardRepresentative(c,rep)` where `rep` is the list `[(j,j+1),(j+1,j+2),...,(j+n-2,j+n-1)]`. By default, `j=1`.

Example

```
gap> s1:= SpinSymStandardRepresentativeImage([7,4,3,1]);
(1,7,6,5,4,3,2)(8,11,10,9)(12,14,13)
gap> s2:= SpinSymStandardRepresentativeImage([[7,4,3,1],'-']);
(1,2,7,6,5,4,3)(8,11,10,9)(12,14,13)
gap> s2 = s1^(1,2);
true
gap> SpinSymStandardRepresentativeImage([7,4,3,1],3);
(3,9,8,7,6,5,4)(10,13,12,11)(14,16,15)
```

2.3.3 SpinSymPreimage

▷ `SpinSymPreimage(c, rep)` (function)

Returns: a (standard) lift of the element `c` of S_n in $\tilde{S}_n$ under a given $\tilde{S}_n$ -representation.

See [2, (5.1.12)] for the definition of the lift that is returned by this function. The permutation `c` is written as a product of simple transpositions $(i, i+1)$, then these are replaced by the images of their canonical lifts t_i under a given representation R of $\tilde{S}_n$ (recall the beginning of Chapter 1 for the definition of t_i). Here `rep` is assumed to be the list $[t_1^R, t_2^R, \ldots, t_{n-1}^R]$.

Note that a more efficient computation may be achieved by computing and storing a list of all necessary transpositions once and for all, before lifting (many) elements (under a possibly large representation).

Example

```
gap> rep:= BasicSpinRepresentationOfSymmetricGroup(15);;
gap> c:= SpinSymStandardRepresentativeImage([5,4,3,2,1]);
(1,5,4,3,2)(6,9,8,7)(10,12,11)(13,14)
gap> C:= SpinSymPreimage(c,rep);
< immutable compressed matrix 64x64 over GF(9) >
gap> C = SpinSymStandardRepresentative([5,4,3,2,1],rep);
true
```

2.3.4 SpinSymBrauerCharacter

▷ `SpinSymBrauerCharacter(ccl, ords, rep)` (function)

Returns: the Brauer character afforded by a given representation of $\tilde{S}_n$.

This function is based on a simplified computation of the GAP attribute `BrauerCharacterValue(mat)` for an invertible matrix `mat` over a finite field whose characteristic is coprime to the order of `mat`.

The arguments *ccl* and *ords* are expected to be the values of the attributes `ClassParameters(modtbl)` and `OrdersClassRepresentatives(modtbl)` of a (possibly incomplete) p-modular SpinSym table `modtbl` of $\tilde{S}_n$.

The argument *rep* is assumed to be a list $[t_1^R, t_2^R, \ldots, t_{n-1}^R]$ given by the images of the generators $t_1, \ldots, t_{n-1}$ of $\tilde{S}_n$ under a (not necessarily faithful) $\tilde{S}_n$-representation R.

Example

```
gap> ct:= CharacterTable("DoubleCoverSymmetric",15);;
gap> bt:= CharacterTableRegular(ct,5);;
gap> fus:= GetFusionMap(bt,ct);;
gap> ccl:= ClassParameters(ct){fus};;
gap> ords:= OrdersClassRepresentatives(bt);;
gap> rep:= BasicSpinRepresentationOfSymmetricGroup(15,5);;
gap> phi:= SpinSymBrauerCharacter(ccl,ords,rep);;
gap> phi in Irr(ct mod 5);
true
```

2.3.5 SpinSymBasicCharacter

▷ `SpinSymBasicCharacter(modtbl)` (function)

Returns: a p-modular basic spin character of the (possibly incomplete) p-modular SpinSym table *modtbl* of $\tilde{S}_n$.

This is just a shortcut for constructing a basic spin representation of $\tilde{S}_n$ in characteristic p and computing its Brauer character by calling `SpinSymBrauerCharacter` (2.3.4) afterwards.

Example

```
gap> SetClassParameters(bt,ccl);
gap> SpinSymBasicCharacter(bt) = phi;
true
```

2.4 Young subgroups

Let k and l be integers greater than 1 and set $n = k + l$. The following subgroup of $\tilde{S}_n$,

$$\tilde{S}_{k,l} = \langle t_1, \ldots, t_{k-1}, t_{k+1}, \ldots, t_{n-1} \rangle,$$

is called a (maximal) *Young subgroup* of $\tilde{S}_n$. Similarly, $\tilde{A}_{k,l} = \tilde{S}_{k,l} \cap \tilde{A}_n$ is a (maximal) Young subgroup of $\tilde{A}_n$. Note that $(\tilde{S}_{k,l})^\pi \cong S_k \times S_l$ and $(\tilde{A}_{k,l})^\pi \cong A_k \times A_l$ but only $\tilde{A}_{k,l} \cong (\tilde{A}_k \times \tilde{A}_l)/\langle (z,z) \rangle$ is a central product. In between $\tilde{A}_{k,l}$ and $\tilde{S}_{k,l}$ there are further central products $\tilde{S}_k \circ \tilde{A}_l \cong (\tilde{S}_k \times \tilde{A}_l)/\langle (z,z) \rangle$ and $\tilde{A}_k \circ \tilde{S}_l \cong (\tilde{A}_k \times \tilde{S}_l)/\langle (z,z) \rangle$ which are π-preimages of $S_k \times A_l$ and $A_k \times S_l$, respectively. See [2, Section 5.2].

2.4.1 SpinSymCharacterTableOfMaximalYoungSubgroup

▷ `SpinSymCharacterTableOfMaximalYoungSubgroup(k, l, type)` (function)

Returns: the ordinary character table of a maximal Young subgroup depending on *type*.

For integers *k* and *l* greater than 1 the function returns the ordinary character table of $\tilde{A}_{k,l}$, $\tilde{A}_k \circ \tilde{S}_l$, $\tilde{S}_k \circ \tilde{A}_l$, or $\tilde{S}_{k,l}$ depending on the string *type* being `"Alternating"`, `"AlternatingSymmetric"`, `"SymmetricAlternating"`, or `"Symmetric"`, respectively.

If *type* is `"Symmetric"` then the output is computed by means of Clifford's theory from the character tables of $\tilde{S}_k \circ \tilde{A}_l$, $\tilde{A}_{k,l}$, and $\tilde{A}_k \circ \tilde{S}_l$ (see [2, Section 5.2]). These 'ingredients' are computed and then stored in the attribute `SpinSymIngredients` so they can be accessed during the construction (and for the construction of a relative Brauer table too, see `SpinSymBrauerTableOfMaximalYoungSubgroup` (2.4.2)).

The construction of the character tables of *type* `"Alternating"`, `"AlternatingSymmetric"`, or `"SymmetricAlternating"` is straightforward and may be accomplished by first constructing a direct product, for example, the character table of $\tilde{S}_k \times \tilde{A}_l$, followed by the construction of the character table of the factor group mod $\langle (z,z) \rangle$.

However, we use a faster method that builds up the table from scratch, using the appropriate component tables as ingredients (for example, the generic character tables of $\tilde{S}_k$ and $\tilde{A}_l$). In this way we can easily build up a suitable list of class parameters that are needed to determine the class fusion in the construction of *type* `"Symmetric"`.

Example

```
gap> 2AA:= SpinSymCharacterTableOfMaximalYoungSubgroup(8,5,"Alternating");
CharacterTable( "2.(Alt(8)xAlt(5))" )
gap> SpinSymCharacterTableOfMaximalYoungSubgroup(8,5,"AlternatingSymmetric");
CharacterTable( "2.(Alt(8)xSym(5))" )
gap> SpinSymCharacterTableOfMaximalYoungSubgroup(8,5,"SymmetricAlternating");
CharacterTable( "2.(Sym(8)xAlt(5))" )
gap> 2SS:= SpinSymCharacterTableOfMaximalYoungSubgroup(8,5,"Symmetric");
CharacterTable( "2.(Sym(8)xSym(5))" )
```

2.4.2 SpinSymBrauerTableOfMaximalYoungSubgroup

▷ `SpinSymBrauerTableOfMaximalYoungSubgroup(ordtbl, p)` (function)

Returns: the *p*-modular character table of the ordinary character table *ordtbl* returned by the function `SpinSymCharacterTableOfMaximalYoungSubgroup` (2.4.1).

If the rational prime *p* is odd, then the construction of the irreducible Brauer characters is really the same as in the ordinary case but it depends on the *p*-modular tables of *ordtbl*'s 'ingredients'. If some Brauer table that is necessary for the construction is not available then `fail` is returned.

Alternatively, the `mod`-operator may be used.

For $p = 2$ the Brauer table is essentially constructed as a direct product by standard GAP methods written by Thomas Breuer.

We call a character table returned by `SpinSymCharacterTableOfMaximalYoungSubgroup` (2.4.1) or `SpinSymBrauerTableOfMaximalYoungSubgroup` (2.4.2) a SpinSym table too. It has lists of class and character parameters whose format is explained in [2, Sections 5.2, 5.3].

Example

```
gap> SpinSymBrauerTableOfMaximalYoungSubgroup(2AA,3);
BrauerTable( "2.(Alt(8)xAlt(5))", 3 )
gap> 2SS mod 5;
```

```
BrauerTable( "2.(Sym(8)xSym(5))", 5 )
gap> ct:= 2SS mod 2;
BrauerTable( "2.(Sym(8)xSym(5))", 2 )
gap> ct1:= CharacterTable("Sym(8)") mod 2;;
gap> ct2:= CharacterTable("Sym(5)") mod 2;;
gap> Irr(ct1*ct2) = Irr(ct);
true
```

2.5 Class Fusions

The following functions determine class fusion maps between SpinSym tables by means of their class parameters. Such 'default' class fusion maps allow one to induce characters from various subgroups of $\tilde{S}_n$ or $\tilde{A}_n$ consistently.

2.5.1 SpinSymClassFusion

▷ `SpinSymClassFusion(`*ctSource*`, `*ctDest*`)` (function)

Returns: a fusion map from the SpinSym table *ctSource* to the SpinSym table *ctDest*. This map is stored if there is no other fusion map from *ctSource* to *ctDest* stored yet.

The possible input tables are expected to be either ordinary or p-modular SpinSym tables of the following pairs of groups

Source	$\to$	Dest
$\tilde{A}_n$		$\tilde{S}_n$
$\tilde{S}_k$		$\tilde{S}_n$
$\tilde{A}_k$		$\tilde{A}_n$
$\tilde{S}_{n-2}$		$\tilde{A}_n$
$\tilde{S}_{k,l}$		$\tilde{S}_{k+l}$
$\tilde{S}_k \circ \tilde{A}_l$		$\tilde{S}_{k,l}$
$\tilde{A}_k \circ \tilde{S}_l$		$\tilde{S}_{k,l}$
$\tilde{A}_{k,l}$		$\tilde{S}_k \circ \tilde{A}_l$
$\tilde{A}_{k,l}$		$\tilde{A}_k \circ \tilde{S}_l$
$\tilde{A}_{k,l}$		$\tilde{A}_{k+l}$

The appropriate function (see the descriptions below) is called to determine the fusion map `fus`. If `GetFusionMap(ctSource, ctDest)` fails, then `fus` is stored by calling `StoreFusion(ctSource, fus, ctDest)`.

Example

```
gap> ctD:= CharacterTable("2.Sym(18)");;
gap> ctS:= SpinSymCharacterTableOfMaximalYoungSubgroup(10,8,"Symmetric");;
gap> GetFusionMap(ctS,ctD);
fail
gap> SpinSymClassFusion(ctS,ctD);;
#I  SpinSymClassFusion: stored fusion map from 2.(Sym(10)xSym(8)) to 2.Sym(18)
gap> GetFusionMap(ctS,ctD) <> fail;
true
```

2.5.2 SpinSymClassFusion2Ain2S

▷ `SpinSymClassFusion2Ain2S(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{A}_n$ and $\tilde{S}_n$.

Given lists of class parameters *cclSource* and *cclDest* of (ordinary or p-modular) SpinSym tables of $\tilde{A}_n$ and $\tilde{S}_n$, respectively, a corresponding class fusion map is determined. See [2, (5.4.1)].

2.5.3 SpinSymClassFusion2Sin2S

▷ `SpinSymClassFusion2Sin2S(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{S}_k$ and $\tilde{S}_n$ for $k \leq n$.

Given lists of class parameters *cclSource* and *cclDest* of (ordinary or p-modular) SpinSym tables of $\tilde{S}_k$ and $\tilde{S}_n$ for $k \leq n$, respectively, a corresponding class fusion map is determined. See [2, (5.4.2)].

Example

```
gap> ctD:= CharacterTable("2.Sym(18)");;
gap> ctS:= CharacterTable("2.Sym(6)");;
gap> cclD:= ClassParameters(ctD);;
gap> cclS:= ClassParameters(ctS);;
gap> fus:= SpinSymClassFusion2Sin2S(cclS,cclD);;
gap> StoreFusion(ctS,fus,ctD);
```

2.5.4 SpinSymClassFusion2Ain2A

▷ `SpinSymClassFusion2Ain2A(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{A}_k$ and $\tilde{A}_n$ for $k \leq n$.

Given lists of class parameters *cclSource* and *cclDest* of (ordinary or p-modular) SpinSym tables of $\tilde{A}_k$ and $\tilde{A}_n$ for $k \leq n$, respectively, a corresponding class fusion map is determined. See [2, (5.4.3)].

2.5.5 SpinSymClassFusion2Sin2A

▷ `SpinSymClassFusion2Sin2A(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{S}_{n-2}$ and $\tilde{A}_n$.

Given lists of class parameters *cclSource* and *cclDest* of (ordinary or p-modular) SpinSym tables of $\tilde{S}_{n-2}$ and $\tilde{A}_n$, respectively, a corresponding class fusion map with respect to the embedding of $\langle t_1 t_{n-1}, \ldots, t_{n-3} t_{n-1} \rangle \cong \tilde{S}_{n-2}$ in $\tilde{A}_n$ is determined. See [2, (5.4.4)].

2.5.6 SpinSymClassFusion2SSin2S

▷ `SpinSymClassFusion2SSin2S(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{S}_{k,l}$ and $\tilde{S}_{k+l}$.

Given lists of class parameters *cclSource* and *cclDest* of (ordinary or p-modular) SpinSym tables of $\tilde{S}_{k,l}$ and $\tilde{S}_{k+l}$, respectively, a corresponding class fusion map is determined by means of [2, (5.1.6)].

Example

```
gap> ctD:= CharacterTable("2.Sym(18)");;
gap> ctS:= SpinSymCharacterTableOfMaximalYoungSubgroup(10,8,"Symmetric");;
gap> cclD:= ClassParameters(ctD);;
```

```
gap> cclS:= ClassParameters(ctS);;
gap> fus:= SpinSymClassFusion2SSin2S(cclS,cclD);;
gap> StoreFusion(ctS,fus,ctD);
```

2.5.7 SpinSymClassFusion2SAin2SS

▷ `SpinSymClassFusion2SAin2SS(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{S}_k \circ \tilde{A}_l$ and $\tilde{S}_{k,l}$.

Given lists of class parameters `cclSource` and `cclDest` of (ordinary or p-modular) SpinSym tables of $\tilde{S}_k \circ \tilde{A}_l$ and $\tilde{S}_{k,l}$, respectively, a corresponding class fusion map is determined. See [2, (5.4.6)].

2.5.8 SpinSymClassFusion2ASin2SS

▷ `SpinSymClassFusion2ASin2SS(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{A}_k \circ \tilde{S}_l$ and $\tilde{S}_{k,l}$.

Given lists of class parameters `cclSource` and `cclDest` of (ordinary or p-modular) SpinSym tables of $\tilde{A}_k \circ \tilde{S}_l$ and $\tilde{S}_{k,l}$, respectively, a corresponding class fusion map is determined analogously to [2, (5.4.6)].

2.5.9 SpinSymClassFusion2AAin2SA

▷ `SpinSymClassFusion2AAin2SA(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{A}_{k,l}$ and $\tilde{S}_k \circ \tilde{A}_l$.

Given lists of class parameters `cclSource` and `cclDest` of (ordinary or p-modular) SpinSym tables of $\tilde{A}_{k,l}$ and $\tilde{S}_k \circ \tilde{A}_l$, respectively, a corresponding class fusion map is determined. See [2, (5.4.7)].

2.5.10 SpinSymClassFusion2AAin2AS

▷ `SpinSymClassFusion2AAin2AS(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{A}_{k,l}$ and $\tilde{A}_k \circ \tilde{S}_l$.

Given lists of class parameters `cclSource` and `cclDest` of (ordinary or p-modular) SpinSym tables of $\tilde{A}_{k,l}$ and $\tilde{A}_k \circ \tilde{S}_l$, respectively, a corresponding class fusion map is determined analogously to [2, (5.4.7)].

2.5.11 SpinSymClassFusion2AAin2A

▷ `SpinSymClassFusion2AAin2A(cclSource, cclDest)` (function)

Returns: a fusion map between the SpinSym tables of $\tilde{A}_{k,l}$ and $\tilde{A}_{k+l}$.

Given lists of class parameters `cclSource` and `cclDest` of (ordinary or p-modular) SpinSym tables of $\tilde{A}_{k,l}$ and $\tilde{A}_{k+l}$, respectively, a corresponding class fusion map is determined. See [2, (5.4.8)].

References

[1] T. Breuer. CTblLib. The GAP Character Table Library. https://www.math.rwth-aachen.de/~Thomas.Breuer/ctbllib/. 4, 5

[2] L. Maas. Modular spin characters of symmetric groups. University of Duisburg--Essen, 2011. https://nbn-resolving.org/urn:nbn:de:hbz:464-20120116-135757-2. 4, 5, 11, 12, 13, 15, 16

[3] F. Noeske. Zur Darstellungstheorie der Schurschen Erweiterungen symmetrischer Gruppen. RWTH Aachen University, 2002. https://www.math.rwth-aachen.de/~Felix.Noeske/2Sn.pdf. 6

Index

SpinSymBasicCharacter, 12
SpinSymBrauerCharacter, 12
SpinSymBrauerTableOfMaximalYoung-
Subgroup, 13
SpinSymCharacterTableOfMaximalYoung-
Subgroup, 13
SpinSymClassFusion, 14
SpinSymClassFusion2AAin2A, 16
SpinSymClassFusion2AAin2AS, 16
SpinSymClassFusion2AAin2SA, 16
SpinSymClassFusion2Ain2A, 15
SpinSymClassFusion2Ain2S, 15
SpinSymClassFusion2ASin2SS, 16
SpinSymClassFusion2SAin2SS, 16
SpinSymClassFusion2Sin2A, 15
SpinSymClassFusion2Sin2S, 15
SpinSymClassFusion2SSin2S, 15
SpinSymPreimage, 11
SpinSymStandardRepresentative, 10
SpinSymStandardRepresentativeImage, 11