The GAP Character Table Library

Version 1.0

maintained by

Thomas Breuer

Lehrstuhl D für Mathematik,
RWTH Aachen, Germany

Contents

1 Introduction to the GAP Character Table Library

The usefulness of GAP for character theoretic tasks depends on the availability of many known character tables, and there is a lot of character tables in the GAP table library. Of course, this library is "open" in the sense that it shall be extended. So we would be grateful for any further tables of interest sent to us for inclusion into our library. Please send interesting new character tables via e-mail to `sam@math.rwth-aachen.de`.

It depends on your GAP installation whether the character table library is available. You can check this as follows.

```
gap> IsBound( LOADED_PACKAGES.ctbllib );
true
```

If the result is `false` then the library is not installed, and you may ask your system administrator for installing it, or install the library in your home directory (see 1.2).

For general information about character tables in GAP, see Chapter 69 in the GAP Reference Manual.

The `doc` directory of the GAP Character Table Library contains several files with demonstrations of computations with character tables. Currently these are `ctbldeco.dvi`, `ctblpope.dvi`, and `multfree.dvi`.

If you use the GAP Character Table Library to solve a problem then please send a short email to `sam@math.rwth-aachen.de` about it. The GAP Character Table Library database should be referenced with the entry [Bre01] in the bibliography of this manual.

For referencing the GAP system in general, use the entry [GAP01] in the bibliography of this manual.

1.1 History of the GAP Character Table Library

The first version of the GAP Character Table Library was released with GAP 3.1 in March 1992.

It was the first aim of this library to continue the character table library of the CAS system (see [NPP84]) in GAP, as a part of the process of reimplementing the algorithms of CAS in GAP (see 69.2 in the GAP Reference Manual). GAP 3.1 provided only very restricted methods for computing character tables from groups, so its character theory part was concerned mainly with library tables.

A second aspect of the character table library was to make all character tables shown in the ATLAS of Finite Groups ([CCN+85]) available in GAP. In fact GAP turned out to provide a very good environment for systematic checks of these character tables.

To some extent, the access to the (ordinary) character tables in [CCN+85] was a prerequisite for storing also the corresponding Brauer character tables in the GAP character table library. Already GAP 3.1 contained many of these tables. They have been computed mainly "outside of GAP", using the methods described in [HJLP], and part of the library has been published in the ATLAS of Brauer Characters ([JLPW95]). One of the roles of GAP was again to perform systematic checks.

Besides these projects, many individual character tables have been added to the GAP Character Table Library since the times of GAP 3.1. They were computed from groups or with character theoretic methods or using a combination of these two possibilities (see, e.g., [NPP84] and [LP91]). Section 2.1 lists some of the sources. The changes in the GAP Character Table Library since the release of GAP 4.1 (in July 1999) are individually documented in the file `ctbldiff.dvi` which is part of the distribution of the library.

In the meantime, a rudimentary interface between groups in GAP and the tables in the GAP Character Table Library has been provided (see 69.6 in the GAP Reference Manual). Similarly, there is an interface to the GAP Libary of Tables of Marks (see 68.12 in the GAP Reference Manual).

Currently the main focus in the development of the GAP Character Table Library is –besides the addition of tables that appear to be interesting– the better interaction with other databases, such as the ATLAS of Group Representations (see the GAP 4 package AtlasRep), and an improvement of the "database" aspect of the character table library itself, for example by providing a "WWW table of contents".

Until the release of GAP 4.3 in spring 2002, the GAP Character Table Library had been a part of the main GAP library. With GAP 4.3, it is "split off" as a GAP package.

1.2 Installing the GAP Character Table Library

To install the GAP package `ctbllib`, unpack the archive file in a directory in the `pkg` directory of your local copy of GAP 4. This might be the `pkg` directory of the GAP 4 home directory, see Section 74.1 of the GAP 4 Reference Manual for details. It is however also possible to keep an additional `pkg` directory in your private directories, see 74.2 of the GAP 4 Reference Manual. The latter possibility **must** be chosen if you do not have write access to the GAP root directory.

The package consists entirely of GAP code, no external binaries need to be compiled.

Both dvi and PostScript versions of the manual for the GAP Character Table Library are available (as `manual.dvi` and `manual.ps` respectively) in the `doc` directory of the `ctbllib` package, and an HTML version can be found in the `htm` directory.

1.3 Loading the GAP Character Table Library

The GAP Character Table Library may be loaded automatically when GAP is started, or it has to be loaded within GAP as follows.

```
gap> RequirePackage( "ctbllib" );
```

See 74.3 in the GAP Reference Manual for details about these alternatives; also the possibility to disable automatic loading of the library is described in this manual section. The default is that the GAP Character Table Library is loaded automatically.

1.4 Acknowledgements

The functions for the conversion of CAS tables to GAP format have been written by Götz Pfeiffer. The functions for converting the "Cambridge format" (in which the original data of the ATLAS of Finite Groups had been stored) to GAP format have been written by Christoph Jansen. The functions for checking library tables have been written by Thomas Breuer.

The development of the GAP Character Table Library has been supported by several DFG grants, in particular the project "Representation Theory of Finite Groups and Finite Dimensional Algebras" (until 1991), and the Schwerpunkt "Algorithmische Zahlentheorie und Algebra" (from 1991 until 1997).

2 The GAP Character Table Library

This chapter informs you about

the currently available character tables (see 2.1),

how to access library tables (see 2.2),

generic character tables (see 2.3 and 2.4),

the subsets of ATLAS tables (see 2.5 and 2.6) and CAS tables (see 2.7),

the organization of the table library (see 2.8),

and how to extend the library (see 2.9).

The latter two sections are rather technical, they are thought only for those who want to maintain or extend the table library.

2.1 Contents of the GAP Character Table Library

This section gives a brief overview of the contents of the GAP character table library. For the details about, e.g., the structure of data files, see 2.8.

The changes in the character table library are listed in a file that can be reached from

`http://www.gap-system.org/Info4/changes.html`

There are three different kinds of character tables in the GAP library, namely **ordinary character tables**, **Brauer tables**, and **generic character tables**. Note that the Brauer table and the corresponding ordinary table of a group determine the **decomposition matrix** of the group (or the decomposition matrices of its blocks). These decomposition matrices can be computed with GAP (see 69.9 in the GAP Reference Manual for details). A collection of DVI and PostScript files of the known decomposition matrices of almost simple groups in the GAP table library can also be found at

`http://www.math.rwth-aachen.de/~MOC/decomposition/`

Ordinary Character Tables

Two different aspects are useful to list the ordinary character tables available in GAP, namely the aspect of the **source** of the tables and that of **connections** between the tables.

As for the source, there are two big sources, namely the ATLAS of Finite Groups (see 2.5) and the CAS library of character tables (see [NPP84]). Many ATLAS tables are contained in the CAS library, and difficulties may arise because the succession of characters and classes in CAS tables and ATLAS tables are in general different, so see 2.7 for the relations between these two variants of character tables of the same group. A large subset of the CAS tables is the set of tables of Sylow normalizers of sporadic simple groups as published in [Ost86] –this may be viewed as another source of character tables. The library also contains the character tables of factor groups of space groups (computed by W. Hanrath, see [Han88]) that are part of [HP89] in form of two microfiches; these tables are given in CAS format (see 2.7) on the microfiches, but they had not been part of the "official" CAS library.

To avoid confusion about the ordering of classes and characters in a given table, authorship and so on, the `InfoText` (see 69.8.12 in the GAP Reference Manual) value of the table contains the information

`origin: ATLAS of finite groups`
: for ATLAS tables (see 2.5),

`origin: Ostermann`
: for tables contained in [Ost86],

`origin: CAS library`
: for any table of the CAS table library that is contained neither in the ATLAS nor in [Ost86], and

`origin: Hanrath library`
: for tables contained in the microfiches in [HP89].

The `InfoText` value usually contains more detailed information, for example that the table in question is the character table of a maximal subgroup of an almost simple group. If the table was contained in the CAS library then additional information may be available via the `CASInfo` value (see 2.7.1).

If one is interested in the aspect of connections between the tables, i.e., the internal structure of the library of ordinary tables, the contents can be listed up the following way.

We have

- all ATLAS tables (see 2.5), i.e., the tables of the simple groups which are contained in the ATLAS of Finite Groups, and the tables of cyclic and bicyclic extensions of these groups,
- most tables of maximal subgroups of sporadic simple groups (**not all** for the groups F_{3+}, B, M),
- some tables of maximal subgroups of other ATLAS tables, where the list of maximal subgroups is complete if the `Maxes` value for the table is known (see 2.2.2),
- the tables of most nontrivial Sylow normalizers of sporadic simple groups as printed in [Ost86], where nontrivial means that the Sylow p normalizer is not contained in $p:(p-1)$ (**not** J_4N2, Co_1N2, Co_1N5, all of Fi_{23}, Fi'_{24}, B, M, HN, and $Fi_{22}N2$)
- some tables of element centralizers
- some tables of Sylow subgroups
- a few other tables, e.g. `W(F4)` (**namely which?**)

Note that class fusions stored on library tables are neither guaranteed to be consistent for any two subgroups of a group and their intersection, nor tested to be consistent with respect to composition of maps.

Brauer Tables

The library contains all tables of the ATLAS of Brauer Tables ([JLPW95]), and many other Brauer tables of bicyclic extensions of simple groups which are known yet. There is ongoing work in computing new tables, so this part of the library is growing.

The Brauer tables in the library contain the information

```
origin: modular ATLAS of finite groups
```

in their `InfoText` string (see 69.8.12 in the GAP Reference Manual).

Generic Character Tables

See 2.3 for an overview of generic tables available.

2.2 Access to Library Character Tables

This section describes how to access a specific character table (see 2.2.1), known character tables of maximal subgroups (see 2.2.2), and how to select character tables with prescribed properties (see 2.2.3).

1 ▶ `CharacterTableFromLibrary(` *tblname* `)` F
▶ `CharacterTableFromLibrary(` *series*`,` *param1*`[,` *param2*`] )` F

If the only argument is a string *tblname* and if this is an admissible name (see below) of a library character table then `CharacterTableFromLibrary` returns this library table, otherwise `fail`.

If `CharacterTableFromLibrary` is called with more than one argument then the first must be a string *series* specifying a series of groups which is implemented via a generic character table, for example `"Symmetric"` for symmetric groups; the remaining arguments specialise then the desired member of the series (see 2.3 for a list of available generic tables). If no generic table with name *series* is available or if the parameters are not admissible then `CharacterTableFromLibrary` returns `fail`.

A call of `CharacterTableFromLibrary` may cause to read some library files and to construct the table object from the data stored in these files, so fetching a library table may take more time than on expects.

`CharacterTableFromLibrary` is called by `CharacterTable` if the first argument is a string, so one may also call `CharacterTable`.

Admissible names for the **ordinary character table** t of the group G are

- an ATLAS like name if t is an ATLAS table (see 2.5), for example `"M22"` for the table of the Mathieu group M_{22}, `"L2(13).2"` for $L_2(13) : 2$, and `"12_1.U4(3).2_1"` for $12_1.U_4(3).2_1$,

  (The difference to the name printed in the ATLAS is that subscripts and superscripts are omitted except if they are used to qualify integer values, and double dots are replaced by a single dot.)

- the names that were admissible for tables of G in CAS if the CAS table library contained a table of G, for example `sl42` for the table of the alternating group A_8,

  (But note that the GAP table may be different from that in CAS, see 2.7.)

- some "relative" names, as follows.

If G is the n–th maximal subgroup (in decreasing group order) of a group whose library table s is available in GAP and stores the `Maxes` value (see 2.2.2), and if *name* is an admissible name for s then *name*`Mn` is admissible for t. For example, the name `"J3M2"` can be used to access the second maximal subgroup of the sporadic simple Janko group J_3 which has the admissible name `J3`.

If G is a nontrivial Sylow p normalizer in a sporadic simple group with admissible name *name*, –where nontrivial means that G is not isomorphic to a subgroup of $p : (p-1)$– then *name*`N`p is an admissible name of t. For example, the name `"J4N11"` can be used to access the table of the Sylow 11 normalizer in the sporadic simple Janko group J_4.

In a few cases, the table of the Sylow p subgroup of G is accessible via the name *name*`Syl`p where *name* is an admissible name of the table of G. For example, `"A11Syl2"` is an admissible name for the table of the Sylow 2 subgroup of the alternating group A_{11}.

In a few cases, the table of an element centralizer in G is accessible via the name *name*`C`cl where *name* is an admissible name of the table of G. For example, `"M11C2"` is an admissible name for the table of an involution centralizer in the Mathieu group M_{11}.

The recommended way to access **Brauer tables** from the library is via the `mod` operator from the ordinary table and the desired characteristic (see 69.3.2 and 69.7 in the GAP Reference Manual), so it is not necessary to define admissible names of Brauer tables.

A **generic character table** (see 2.3) is accessible only by the name given by its `Identifier` value (see 69.8.11 in the GAP Reference Manual).

Case is not significant for character table names. For example, both "suzm3" and "SuzM3" are admissible names for the third maximal subgroup of the sporadic simple Suzuki group.

```
gap> s5:= CharacterTable( "A5.2" );
CharacterTable( "A5.2" )
gap> sym5:= CharacterTable( "Symmetric", 5 );
CharacterTable( "Sym(5)" )
gap> TransformingPermutationsCharacterTables( s5, sym5 );
rec( columns := (2,3,4,7,5), rows := (1,7,3,4,6,5,2), group := Group(()) )
```

The two tables are tables of the symmetric group on five letters; the first is in ATLAS format (see 2.5), the second is constructed from the generic table for symmetric groups (see 2.3).

```
gap> CharacterTable( "J5" );
fail
gap> CharacterTable( "A5" ) mod 2;
BrauerTable( "A5", 2 )
```

2 ▶ `Maxes( `*tbl*` )` A

is a list of identifiers of the tables of all maximal subgroups of *tbl*. This is meaningful usually only for library tables, and there is no default method to compute the value.

If the Maxes value of *tbl* is stored then it lists exactly one representative for each conjugacy class of maximal subgroups of the group of *tbl*; the tables of these maximal subgroups are then available in the GAP table library, and the fusions to *tbl* are stored on these tables.

```
gap> tbl:= CharacterTable( "M11" );;
gap> HasMaxes( tbl );
true
gap> maxes:= Maxes( tbl );
[ "A6.2_3", "L2(11)", "3^2:Q8.2", "A5.2", "2.S4" ]
gap> CharacterTable( maxes[1] );
CharacterTable( "A6.2_3" )
```

3 ▶ `AllCharacterTableNames( `*func*`, `*val*` )` F
▶ `AllCharacterTableNames( `*func*`, `*val*`, ..., OfThose, `*func*` )` F

Similar to group libraries (see Chapter 48 in the GAP Reference Manual), the GAP character table library can be used to search for ordinary character tables with prescribed properties.

A specific library table can be selected by an admissible name (see 2.2.1).

The selection function for character tables with certain abstract properties is AllCharacterTableNames. Contrary to the situation in the case of group libraries, the selection function returns a list not of library character tables but of their names; using CharacterTable one can then access the tables themselves.

AllCharacterTableNames takes an arbitrary even number of arguments. The argument at each odd position must be a function, and the argument at the subsequent even position must be a value that this function must return when called for the character table in question, in order to have the name of the table included in the selection, or a list of such values. For example,

```
gap> names:= AllCharacterTableNames();;
```

returns a list containing one admissible name of each ordinary character table in the GAP library, and

```
gap> simpnames:= AllCharacterTableNames( IsSimple, true );;
gap> AllCharacterTableNames( IsSimple, true, Size, [ 1 .. 100 ] );
[ "A5" ]
```

return lists containing an admissible name of each ordinary character table in the GAP library whose groups are simple or are simple and have order at most 100, respectively.

For the sake of efficiency, the arguments `IsSimple` and `IsSporadicSimple` followed by `true` are handled in a special way, GAP need not read all files of the table library in these cases in order to find the desired names.

If the function `OfThose` is an argument at an odd position then the following argument *func* must be a function that takes a character table and returns a name of a character table or a list of names; this is interpreted as replacement of the names computed up to this position by the union of names returned by *func*. For example, *func* may be `Maxes` (see 2.2.2) or `NamesOfFusionSources` (see 71.2.5 in the GAP Reference Manual).

```
gap> maxesnames:= AllCharacterTableNames( IsSporadicSimple, true,
>                                         HasMaxes, true,
>                                         OfThose, Maxes );;
```

returns the union of names of ordinary tables of those maximal subgroups of sporadic simple groups that are contained in the table library.

2.3 Generic Character Tables

Generic character tables provide a means for writing down the character tables of all groups in a (usually infinite) series of similar groups, e.g., cyclic groups, or symmetric groups, or the general linear groups $\mathrm{GL}(2, q)$ where q ranges over certain prime powers.

Let $\{G_q | q \in I\}$ be such a series, where I is an index set. The character table of one fixed member G_q could be computed using a function that takes q as only argument and constructs the table of G_q. It is, however, often desirable to compute not only the whole table but to access just one specific character, or to compute just one character value, without computing the whole character table.

For example, both the conjugacy classes and the irreducible characters of the symmetric group S_n are in bijection with the partitions of n. Thus for given n it makes sense to ask for the character corresponding to a particular partition, or just for its character value at a partition.

A generic character table in GAP allows one such local evaluations. In this sense, GAP can deal also with character tables that are too big to be computed and stored as a whole.

Currently the only operations for generic tables supported by GAP are the specialisation of the parameter q in order to compute the whole character table of G_q, and local evaluation (see 2.3.2 for an example). It is **not** possible to compute, e.g., generic scalar products.

Currently, generic tables of the following groups –in alphabetical order– are available in GAP. (A list of the names of generic tables known to GAP is `LIBTABLE.GENERIC.firstnames`.) We list the function calls needed to get a specialized table, the generic table itself can be accessed by calling `CharacterTable` with the first argument only; for example, `CharacterTable( "Cyclic" )` yields the generic table of cyclic groups.

`CharacterTable( "Alternating", `*n*` )`, the table of the **alternating** group on n letters,

`CharacterTable( "Cyclic", `*n*` )`, the table of the **cyclic** group of order n,

`CharacterTable( "Dihedral", `*2n*` )`, the table of the **dihedral** group of order $2n$,

`CharacterTable( "GL", 2, `*q*` )`, the table of the **general linear** group $\mathrm{GL}(2, q)$, for a prime power q,

`CharacterTable( "GU", 3, q )`, the table of the **general unitary** group $\mathrm{GU}(3,q)$, for a prime power q,

`CharacterTable( "P:Q", [ p, q ] )`, the table of the **Frobenius extension** of the cyclic group of prime order p by a cyclic group of order q where q divides $p-1$,

`CharacterTable( "PSL", 2, q )`, the table of the **projective special linear** group $\mathrm{PSL}(2,q)$, for a prime power q,

`CharacterTable( "SL", 2, q )`, the table of the **special linear** group $\mathrm{SL}(2,q)$, for a prime power q,

`CharacterTable( "SU", 3, q )`, the table of the **special unitary** group $\mathrm{SU}(3,q)$, for a prime power q,

`CharacterTable( "Suzuki", q )`, the table of the **Suzuki** group $Sz(q) = {}^2B_2(q)$, for q an odd power of 2,

`CharacterTable( "Symmetric", n )`, the table of the **symmetric** group on n letters,

`CharacterTable( "WeylB", n )`, the table of the **Weyl** group of type B_n,

`CharacterTable( "WeylD", n )`, the table of the **Weyl** group of type D_n.

In addition to the above calls that really use generic tables, the following calls to `CharacterTable` are to some extent "generic" constructions. But note that no local evaluation is possible in these cases, as no generic table object exists in GAP that can be asked for local information.

`CharacterTable( "Quaternionic", 4n )`, the table of the **quaternionic** (dicyclic) group of order $4n$,

`CharacterTableWreathSymmetric( tbl, n )`, the character table of the wreath product of the group whose table is *tbl* with the symmetric group on n letters (see 69.17.4 in the GAP Reference Manual).

1 ▶ `CharacterTableSpecialized( generic_table, q )` F

For a record *generic_table* representing a generic character table, and a parameter value q, `CharacterTableSpecialized` returns a character table object computed by evaluating *generic_table* at q.

```
gap> c5:= CharacterTableSpecialized( CharacterTable( "Cyclic" ), 5 );
CharacterTable( "C5" )
gap> Display( c5 );
C5

     5  1  1  1  1  1

        1a 5a 5b 5c 5d
    5P  1a 1a 1a 1a 1a

X.1     1  1  1  1  1
X.2     1  A  B /B /A
X.3     1  B /A  A /B
X.4     1 /B  A /A  B
X.5     1 /A /B  B  A

A = E(5)
B = E(5)^2
```

(Also `CharacterTable( "Cyclic", 5 )` could have been used to construct the above table.)

While the numbers of conjugacy classes for the members of a series of groups are usually not bounded, there is always a fixed finite number of **types** (equivalence classes) of conjugacy classes; very often the equivalence relation is isomorphism of the centralizers of the representatives.

For each type t of classes and a fixed $q \in I$, a **parametrisation** of the classes in t is a function that assigns to each conjugacy class of G_q in t a **parameter** by which it is uniquely determined. Thus the classes are indexed by pairs $[t, p_t]$ consisting of a type t and a parameter p_t for that type.

For any generic table, there has to be a fixed number of types of irreducible characters of G_q, too. Like the classes, the characters of each type are parametrised.

In GAP, the parametrisations of classes and characters for tables computed from generic tables is stored using the attributes `ClassParameters` and `CharacterParameters`.

2 ▶ `ClassParameters( tbl )` A
▶ `CharacterParameters( tbl )` A

are lists containing a parameter for each conjugacy class or irreducible character, respectively, of the character table *tbl*.

It depends on *tbl* what these parameters are, so there is no default to compute class and character parameters.

For example, the classes of symmetric groups can be parametrized by partitions, corresponding to the cycle structures of permutations. Character tables constructed from generic character tables (see 2.3) usually have class and character parameters stored.

If *tbl* is a p-modular Brauer table such that class parameters are stored in the underlying ordinary table (see 69.8.4 in the GAP Reference Manual) of *tbl* then `ClassParameters` returns the sublist of class parameters of the ordinary table, for p-regular classes.

```
gap> HasClassParameters( c5 );  HasCharacterParameters( c5 );
true
true
gap> ClassParameters( c5 );  CharacterParameters( c5 );
[ [ 1, 0 ], [ 1, 1 ], [ 1, 2 ], [ 1, 3 ], [ 1, 4 ] ]
[ [ 1, 0 ], [ 1, 1 ], [ 1, 2 ], [ 1, 3 ], [ 1, 4 ] ]
gap> ClassParameters( CharacterTable( "Symmetric", 3 ) );
[ [ 1, [ 1, 1, 1 ] ], [ 1, [ 2, 1 ] ], [ 1, [ 3 ] ] ]
```

Here are examples for local evaluation of generic character tables, first a character value of the cyclic group shown above, then a character value and a representative order of a symmetric group.

```
gap> CharacterTable( "Cyclic" ).irreducibles[1][1]( 5, 2, 3 );
E(5)
gap> tbl:= CharacterTable( "Symmetric" );;
gap> tbl.irreducibles[1][1]( 5, [ 3, 2 ], [ 2, 2, 1 ] );
1
gap> tbl.orders[1]( 5, [ 2, 1, 1, 1 ] );
2
```

Any generic table in GAP is represented by a record. The following components are supported for generic character table records.

`centralizers`
: list of functions, one for each class type t, with arguments q and p_t, returning the centralizer order of the class $[t, p_t]$,

`charparam`
: list of functions, one for each character type t, with argument q, returning the list of character parameters of type t,

`classparam`
: list of functions, one for each class type t, with argument q, returning the list of class parameters of type t,

`classtext`
: list of functions, one for each class type t, with arguments q and p_t, returning a representative of the class with parameter $[t, p_t]$,

`domain`
: function of q returning `true` if q is a valid parameter, and `false` otherwise,

`identifier`
: identifier string of the generic table,

`irreducibles`
: list of list of functions, in row i and column j the function of three arguments, namely q and the parameters p_t and p_s of the class type t and the character type s,

`isGenericTable`
: always `true`

`libinfo`
: record with components `firstname` (`Identifier` value of the table) and `othernames` (list of other admissible names)

`matrix`
: function of q returning the matrix of irreducibles of G_q,

`orders`
: list of functions, one for each class type t, with arguments q and p_t, returning the representative order of elements of type t and parameter p_t,

`powermap`
: list of functions, one for each class type t, each with three arguments q, p_t, and k, returning the pair $[s, p_s]$ of type and parameter for the k-th power of the class with parameter $[t, p_t]$,

`size`
: function of q returning the order of G_q,

`specializedname`
: function of q returning the `Identifier` value of the table of G_q,

`text`
: string informing about the generic table

In the specialized table, the `ClassParameters` and `CharacterParameters` values are the lists of parameters $[t, p_t]$ of classes and characters, respectively.

If the `matrix` component is present then its value implements a method to compute the complete table of small members G_q more efficiently than via local evaluation; this method will be called when the generic table is used to compute the whole character table for a given q (see 2.3.1).

2.4 Examples of Generic Character Tables

1. The generic table of cyclic groups.

For the cyclic group $C_q = \langle x \rangle$ of order q, there is one type of classes. The class parameters are integers $k \in \{0, \ldots, q-1\}$, the class with parameter k consists of the group element x^k. Group order and centralizer orders are the identity function $q \mapsto q$, independent of the parameter k. The representative order function maps the parameter pair $[q, k]$ to $\frac{q}{\gcd(q,k)}$, which is the order of x^k in C_q; the p-th power map is the function mapping the triple (q, k, p) to the parameter $[1, (kp \bmod q)]$.

There is one type of characters, with parameters $l \in \{0, \ldots, q-1\}$; for e_q a primitive complex q-th root of unity, the character values are $\chi_l(x^k) = e_q^{kl}$.

The library file contains the following generic table.

```
rec(
identifier := "Cyclic",
specializedname := ( q -> Concatenation( "C", String(q) ) ),
size := ( n -> n ),
text := "generic character table for cyclic groups",
centralizers := [ function( n, k ) return n; end ],
classparam := [ ( n -> [ 0 .. n-1 ] ) ],
charparam := [ ( n -> [ 0 .. n-1 ] ) ],
powermap := [ function( n, k, pow ) return [ 1, k*pow mod n ]; end ],
orders := [ function( n, k ) return n / Gcd( n, k ); end ],
irreducibles := [ [ function( n, k, l ) return E(n)^(k*l); end ] ],
domain := IsPosInt,
libinfo := rec( firstname:= "Cyclic", othernames:= [] ),
isGenericTable := true )
```

2. The generic table of the general linear group GL(2, q).

We have four types t_1, t_2, t_3, t_4 of classes, according to the rational canonical form of the elements. t_1 describes scalar matrices, t_2 nonscalar diagonal matrices, t_3 companion matrices of $(X-\rho)^2$ for elements $\rho \in \mathbb{F}_q^*$, and t_4 companion matrices of irreducible polynomials of degree 2 over $\mathbb{F}_q$.

The sets of class parameters of the types are in bijection with $\mathbb{F}_q^*$ for t_1 and t_3, with the set $\{\{\rho, \tau\}; \rho, \tau \in \mathbb{F}_q^*, \rho \neq \tau\}$ for t_2, and with the set $\{\{\epsilon, \epsilon^q\}; \epsilon \in \mathbb{F}_{q^2} \setminus \mathbb{F}_q\}$ for t_4.

The centralizer order functions are $q \mapsto (q^2-1)(q^2-q)$ for type t_1, $q \mapsto (q-1)^2$ for type t_2, $q \mapsto q(q-1)$ for type t_3, and $q \mapsto q^2-1$ for type t_4.

The representative order function of t_1 maps (q, ρ) to the order of ρ in $\mathbb{F}_q$, that of t_2 maps $(q, \{\rho, \tau\})$ to the least common multiple of the orders of ρ and τ.

The file contains something similar to the following table.

```
rec(
identifier := "GL2",
specializedname := ( q -> Concatenation( "GL(2,", String(q), ")" ) ),
size := ( q -> (q^2-1)*(q^2-q) ),
text := "generic character table of GL(2,q), see Robert Steinberg: ...",
centralizers := [ function( q, k ) return (q^2-1) * (q^2-q); end,
                  ..., ..., ... ],
classparam := [ ( q -> [ 0 .. q-2 ] ), ..., ..., ... ],
charparam := [ ( q -> [ 0 .. q-2 ] ), ..., ..., ... ],
powermap := [ function( q, k, pow ) return [ 1, (k*pow) mod (q-1) ]; end,
              ..., ..., ... ],
```

```
orders:= [ function( q, k ) return (q-1)/Gcd( q-1, k ); end,
           ..., ..., ... ],
irreducibles := [ [ function( q, k, l ) return E(q-1)^(2*k*l); end,
                    ..., ..., ... ],
                  [ ..., ..., ..., ... ],
                  [ ..., ..., ..., ... ],
                  [ ..., ..., ..., ... ] ],
classtext := [ ..., ..., ..., ... ],
domain := ( q -> IsInt(q) and q > 1 and Length( Set( FactorsInt(q) ) ) = 1 ),
isGenericTable := true )
```

2.5 ATLAS Tables

The GAP character table library contains all character tables that are included in the ATLAS of Finite Groups ([CCN+85], from now on called ATLAS), and the Brauer tables contained in the ATLAS of Brauer Characters ([JLPW95]).

These tables have the information

```
origin: ATLAS of finite groups
```

or

```
origin: modular ATLAS of finite groups
```

in their `InfoText` value (see 69.8.12 in the GAP Reference Manual), they are simply called ATLAS tables further on.

For displaying ATLAS tables with the row labels used in the ATLAS, or for displaying decomposition matrices, see 69.9.5 in the GAP Reference Manual and 2.5.1.

In addition to the information given in Chapters 6–8 of the ATLAS which tell you how to read the printed tables, there are some rules relating these to the corresponding GAP tables.

Improvements

For the GAP library not the printed ATLAS is relevant but the revised version given by the lists of Improvements to the ATLAS maintained by Simon Norton. The first list is contained in [BN95], and printed in the Appendix of [JLPW95]; it contains the improvements that had been known until the "ATLAS of Brauer Characters" was published. The second list can be found in the internet, namely, an HTML version at

```
http://www.mat.bham.ac.uk/atlas/html/atlasmods.html
```

and a DVI version at

```
http://www.mat.bham.ac.uk/atlas/html/atlasmods.dvi
```

This list contains the improvements found since the publication of [JLPW95], it is updated regularly.

Also some tables are regarded as ATLAS tables which are not printed in the ATLAS but available in ATLAS format from Cambridge, according to the lists of improvements mentioned above. Currently these are the tables related to $L_2(49)$, $L_2(81)$, $L_6(2)$, $O_8^-(3)$, $O_8^+(3)$, $S_{10}(2)$, and ${}^2E_6(2).3$.

Power Maps

For the tables of $3.McL$, $3_2.U_4(3)$ and its covers, and $3_2.U_4(3).2_3$ and its covers, the power maps are not uniquely determined by the information from the ATLAS but determined only up to matrix automorphisms (see 69.19.1 in the GAP Reference Manual) of the irreducible characters. In these cases, the first possible map according to lexicographical ordering was chosen, and the automorphisms are listed in the `InfoText` strings of the tables.

Projective Characters and Projections

If G (or $G.a$) has a nontrivial Schur multiplier then the component `projectives` of the GAP table object of G (or $G.a$) is present, whose value is a list of records, each with the following components.

`name`
: the `Identifier` value of the character table of the covering whose faithful irreducible characters are described by the record,

`chars`
: the list of values lists of those faithful projective irreducibles that are printed in the ATLAS (so–called *proxy characters*), and

`map`
: a list of positions that maps each class of G to that preimage in the covering for which the column is printed in the ATLAS (a so–called *proxy class*, this preimage is denoted by g_0 in Chapter 7, Section 14 of the ATLAS). In a sense, a projection map is an inverse of the factor fusion from the table of the covering to the given table (see 71.3.3 in the GAP Reference Manual).

```
gap> CharacterTable( "A5" )!.projectives;
[ rec( name := "2.A5",
      chars := [ [ 2, 0, -1, E(5)+E(5)^4, E(5)^2+E(5)^3 ], [ 2, 0, -1,
              E(5)^2+E(5)^3, E(5)+E(5)^4 ], [ 4, 0, 1, -1, -1 ],
          [ 6, 0, 0, 1, 1 ] ], map := [ 1, 3, 4, 6, 8 ] ) ]
```

Tables of Isoclinic Groups

As described in Chapter 6, Section 7 and in Chapter 7, Section 18 of the ATLAS, there exist two (in general nonisomorphic) groups of structure $2.G.2$ for a simple group G, which are isoclinic. The table in the GAP library is the one printed in the ATLAS, the table of the isoclinic group can be constructed using `CharacterTableIsoclinic` (see 69.17.3 in the GAP Reference Manual).

Ordering of Characters and Classes

(Throughout this paragraph, G always means the simple group involved.)

1. For G itself, the ordering of classes and characters in the GAP table coincides with the one in the ATLAS.
2. For an automorphic extension $G.a$, there are three types of characters.

   If a character χ of G extends to $G.a$ then the different extensions $\chi^0, \chi^1, \ldots, \chi^{a-1}$ are consecutive in the table of $G.a$ (see Chapter 7, Section 16 of the ATLAS).

   If some characters of G fuse to give a single character of $G.a$ then the position of that character in the table of $G.a$ is given by the position of the first involved character of G.

   If both extension and fusion occur for a character then the resulting characters are consecutive in the table of $G.a$, and each replaces the first involved character of G.
3. Similarly, there are different types of classes for an automorphic extension $G.a$, as follows.

   If some classes collapse then the resulting class replaces the first involved class of G.

   For $a > 2$, any proxy class and its algebraic conjugates that are not printed in the ATLAS are consecutive in the table of $G.a$; if more than two classes of $G.a$ have the same proxy class (the only case that actually occurs is for $a = 5$) then the ordering of non-printed classes is the natural one of corresponding Galois conjugacy operators $*k$ (see Chapter 7, Section 19 in the ATLAS).

   For a_1, a_2 dividing a such that $a_1 < a_2$, the classes of $G.a_1$ in $G.a$ precede the classes of $G.a_2$ not contained in $G.a_1$. This ordering is the same as in the ATLAS, with the only exception $U_3(8).6$.
4. For a central extension $M.G$, there are two different types of characters, as follows.

Each character can be regarded as a faithful character of a factor group $m.G$, where m divides M. Characters with the same kernel are consecutive as in the ATLAS, the ordering of characters with different kernels is given by the order of precedence $1, 2, 4, 3, 6, 12$ for the different values of m.

If $m > 2$, a faithful character of $m.G$ that is printed in the ATLAS (a so-called *proxy character*) represents two or more Galois conjugates. In each ATLAS table in GAP, a proxy character always precedes the non-printed characters with this proxy. The case $m = 12$ is the only one that actually occurs where more than one character for a proxy is not printed. In this case, the non-printed characters are ordered according to the corresponding Galois conjugacy operators $*5$, $*7$, $*11$ (in that succession).

5. For the classes of a central extension we have the following.

The preimages of a G-class in $M.G$ are subsequent, the ordering is the same as that of the lifting order rows in the ATLAS (see Chapter 7, Section 7 there).

The primitive roots of unity chosen to represent the generating central element (i.e., the element in the second class of the GAP table) are `E(3)`, `E(4)`, `E(6)^5` (`= E(2) * E(3)`), and `E(12)^7` (`= E(3) * E(4)`), for $m = 3$, 4, 6, and 12, respectively.

6. For tables of bicyclic extensions $m.G.a$, both the rules for automorphic and central extensions hold. Additionally we have the following three rules.

Whenever classes of the subgroup $m.G$ collapse in $m.G.a$ then the resulting class replaces the first involved class.

Whenever characters of the subgroup $m.G$ collapse fuse in $m.G.a$ then the result character replaces the first involved character.

Extensions of a character are subsequent, and the extensions of a proxy character precede the extensions of characters with this proxy that are not printed.

Preimages of a class of $G.a$ in $m.G.a$ are subsequent, and the preimages of a proxy class precede the preimages of non-printed classes with this proxy.

1 ▶ `AtlasLabelsOfIrreducibles(` *tbl*`[, "short"] )` F

Let *tbl* be the (ordinary or Brauer) character table of a bicyclic extension of a simple group that occurs in the ATLAS of Finite Groups [CCN+85] or the ATLAS of Brauer Characters [JLPW95]. `AtlasLabelsOfIrreducibles` returns a list of strings, the i-th entry being a label for the i-th irreducible character of *tbl*.

The labels have the following form. We state the rules only for ordinary characters, the rules for Brauer characters are obtained by replacing χ by φ.

First consider only downward extensions $m.G$ of a simple group G. If $m \leq 2$ then only labels of the form χ_i occur, which denotes the i-th ordinary character shown in the ATLAS.

The labels of faithful ordinary characters of groups $m.G$ with $m \geq 3$ are of the form χ_i, χ_i^*, or χ_i^{*k}, which means the i-th character printed in the ATLAS, the unique character that is not printed and for which χ_i acts as proxy (see Sections 8 and 19 of Chapter 7 in the ATLAS of Finite Groups), and the image of the printed character χ_i under the algebraic conjugacy operator $*k$, respectively.

For groups $m.G.a$ with $a > 1$, the labels of the irreducible characters are derived from the labels of the irreducible constituents of their restrictions to $m.G$, as follows.

1. If the ordinary irreducible character χ_i of $m.G$ extends to $m.G.a$ then the a' extensions are denoted by $\chi_{i,0}, \chi_{i,1}, \ldots, \chi_{i,a'}$, where $\chi_{i,0}$ is the character whose values are printed in the ATLAS.
2. The label $\chi_{i_1+i_2+\cdots+i_a}$ means that a different characters $\chi_{i_1}, \chi_{i_2}, \ldots, \chi_{i_a}$ of $m.G$ induce to an irreducible character of $m.G.a$ with this label.

If the string `"short"` was entered as the second argument then the label has the short form χ_{i_1+}. Note that $i_2, i_3, \ldots, i_a$ can be read off from the fusion signs in the ATLAS.

3. Finally, the label $\chi_{i_1,j_1+i_2,j_2+\cdots+i_{a'},j_{a'}}$ means that the characters $\chi_{i_1}, \chi_{i_2}, \ldots, \chi_{i_{a'}}$ of $m.G$ extend to a group that lies properly between $m.G$ and $m.G.a$, and the extensions $\chi_{i_1,j_1}, \chi_{i_2,j_2}, \ldots \chi_{i_{a'},j_{a'}}$ induce to an irreducible character of $m.G.a$ with this label.

   Again, if the string `"short"` was entered as the second argument then the label has a short form, namely $\chi_{i,j+}$.

```
gap> AtlasLabelsOfIrreducibles( CharacterTable( "3.A7.2" ) );
[ "\\chi_{1,0}", "\\chi_{1,1}", "\\chi_{2,0}", "\\chi_{2,1}", "\\chi_{3+4}",
  "\\chi_{5,0}", "\\chi_{5,1}", "\\chi_{6,0}", "\\chi_{6,1}", "\\chi_{7,0}",
  "\\chi_{7,1}", "\\chi_{8,0}", "\\chi_{8,1}", "\\chi_{9,0}", "\\chi_{9,1}",
  "\\chi_{17+17\\ast 2}", "\\chi_{18+18\\ast 2}", "\\chi_{19+19\\ast 2}",
  "\\chi_{20+20\\ast 2}", "\\chi_{21+21\\ast 2}", "\\chi_{22+23\\ast 8}",
  "\\chi_{22\\ast 8+23}" ]
gap> AtlasLabelsOfIrreducibles( CharacterTable( "3.A7.2" ), "short" );
[ "\\chi_{1,0}", "\\chi_{1,1}", "\\chi_{2,0}", "\\chi_{2,1}", "\\chi_{3+}",
  "\\chi_{5,0}", "\\chi_{5,1}", "\\chi_{6,0}", "\\chi_{6,1}", "\\chi_{7,0}",
  "\\chi_{7,1}", "\\chi_{8,0}", "\\chi_{8,1}", "\\chi_{9,0}", "\\chi_{9,1}",
  "\\chi_{17+}", "\\chi_{18+}", "\\chi_{19+}", "\\chi_{20+}", "\\chi_{21+}",
  "\\chi_{22+}", "\\chi_{23+}" ]
```

2.6 Examples of the ATLAS Format for GAP Tables

We give three little examples for the conventions stated in 2.5, listing both the ATLAS format and the table displayed by GAP.

First, let G be the trivial group. We consider the cyclic group C_6 of order 6. It can be viewed in several ways, namely

1. as a downward extension of the factor group C_2 which contains G as a subgroup, or equivalently, as an upward extension of the subgroup C_3 which has a factor group isomorphic to G,
2. as a downward extension of the factor group C_3 which contains G as a subgroup, or equivalently, as an upward extension of the subgroup C_2 which has a factor group isomorphic to G,
3. as a downward extension of the factor groups C_3 and C_2 which have G as a factor group, or
4. as an upward extension of the subgroups C_3 or C_2 which both contain a subgroup isomorphic to G.

Situation 1. is shown here.

```
 -------   -------         ;   @   ;   ;   @      2   1   1   1   1   1   1
|       | |       |            1           1      3   1   1   1   1   1   1
|   G   | |  G.2  |      p power           A
|       | |       |      p' part           A         1a  3a  3b  2a  6a  6b
 -------   -------       ind  1A fus ind  2A     2P  1a  3b  3a  1a  3b  3a
 -------   -------                            3P  1a  1a  1a  2a  2a  2a
|       | |       |  chi1  +   1   :  ++   1
|  3.G  | | 3.G.2 |                          X.1    1   1   1   1   1   1
|       | |       |      ind   1 fus ind   2  X.2    1   1   1  -1  -1  -1
 -------   -------             3           6  X.3    1   A  /A   1   A  /A
                             3           6  X.4    1   A  /A  -1  -A -/A
                                            X.5    1  /A   A   1  /A   A
                     chi2 o2   1   : oo2   1  X.6    1  /A   A  -1 -/A  -A
```

```
A = E(3)
  = (-1+ER(-3))/2 = b3
```

`X.1`, `X.2` extend χ_1. `X.3`, `X.4` extend the proxy character χ_2. `X.5`, `X.6` extend the not printed character with proxy χ_2. The classes `1a`, `3a`, `3b` are preimages of `1A`, and `2a`, `6a`, `6b` are preimages of `2A`.

Situation 2. is shown here.

```
 -------   -------          ;   @   ;   ;   @      2   1   1   1   1   1   1
|       | |       |           1           1      3   1   1   1   1   1   1
|   G   | |  G.3  |      p power           A
|       | |       |      p' part           A         1a  2a  3a  6a  3b  6b
 -------   -------       ind  1A fus ind  3A     2P  1a  1a  3b  3b  3a  3a
 -------   -------                               3P  1a  2a  1a  2a  1a  2a
|       | |       |  chi1  +   1   : +oo   1
|  2.G  | | 2.G.3 |                           X.1    1   1   1   1   1   1
|       | |       |      ind   1 fus ind   3  X.2    1   1   A   A  /A  /A
 -------   -------             2           6  X.3    1   1  /A  /A   A   A
                                               X.4    1  -1   1  -1   1  -1
                     chi2  +   1   : +oo   1  X.5    1  -1   A  -A  /A -/A
                                              X.6    1  -1  /A -/A   A  -A

                                              A = E(3)
                                                = (-1+ER(-3))/2 = b3
```

`X.1`–`X.3` extend χ_1, `X.4`–`X.6` extend χ_2. The classes `1a` and `2a` are preimages of `1A`, `3a` and `6a` are preimages of the proxy class `3A`, and `3b` and `6b` are preimages of the not printed class with proxy `3A`.

Situation 3. is shown here.

```
 -------             ;   @        2   1   1   1   1   1   1
|       |            1   |        3   1   1   1   1   1   1
|   G   |          p power
|       |          p' part           1a  6a  3a  2a  3b  6b
 -------           ind  1A       2P  1a  3a  3b  1a  3a  3b
 -------                         3P  1a  2a  1a  2a  1a  2a
|       |      chi1  +   1
|  2.G  |                      X.1    1   1   1   1   1   1
|       |          ind   1     X.2    1  -1   1  -1   1  -1
 -------           2   |      X.3    1   A  /A   1   A  /A
 -------                       X.4    1  /A   A   1  /A   A
|       |      chi2  +   1     X.5    1  -A  /A  -1   A -/A
|  3.G  |                      X.6    1 -/A   A  -1  /A  -A
|       |          ind   1
 -------           3   |      A = E(3)
 -------           3           = (-1+ER(-3))/2 = b3
|       |
|  6.G  |      chi3 o2   1
|       |
 -------           ind   1
                         6
                         3
                         2
                         3
                         6
```

```
                chi4 o2   1
```

`X.1`, `X.2` correspond to χ_1, χ_2, respectively; `X.3`, `X.5` correspond to the proxies χ_3, χ_4, and `X.4`, `X.6` to the not printed characters with these proxies. followers. The factor fusion onto $3.G$ is given by `[ 1, 2, 3, 1, 2, 3 ]`, that onto $G.2$ by `[ 1, 2, 1, 2, 1, 2 ]`.

Finally, situation 4. is shown here.

```
 -------   -------   -------   -------
|       | |       | |       | |       |
|   G   | |  G.2  | |  G.3  | |  G.6  |
|       | |       | |       | |       |
 -------   -------   -------   -------

     ;   @   ;   ;   @   ;   ;   @   ;     ;   @

          1           1           1            1
    p power           A           A           AA
    p' part           A           A           AA
    ind  1A fus ind  2A fus ind  3A fus   ind  6A

chi1  +   1   :  ++   1   : +oo   1   :+oo+oo   1


    2   1   1   1   1   1   1
    3   1   1   1   1   1   1

       1a  2a  3a  3b  6a  6b
  2P   1a  1a  3b  3a  3b  3a
  3P   1a  2a  1a  1a  2a  2a
X.1     1   1   1   1   1   1
X.2     1  -1   A  /A  -A -/A
X.3     1   1  /A   A  /A   A
X.4     1  -1   1   1  -1  -1
X.5     1   1   A  /A   A  /A
X.6     1  -1  /A   A -/A  -A

A = E(3)
  = (-1+ER(-3))/2 = b3
```

The classes `1a`, `2a` correspond to $1A$, $2A$, respectively. `3a`, `6a` correspond to the proxies $3A$, $6A$, and `3b`, `6b` to the not printed classes with these proxies.

The second example explains the fusion case; again, G is the trivial group.

```
 -------   -------        ;   @   ;   ;  @      3.G.2
|       | |       |          1           1
|   G   | |  G.2  |     p power          A         2   1   .   1
|       | |       |     p' part          A         3   1   1   .
 -------   -------      ind  1A fus ind 2A
 -------   -------                                    1a 3a 2a
|       | |       |   X1  +   1   :  ++  1        2P  1a 3a 1a
|  2.G  | | 2.G.2 |                               3P  1a 1a 2a
|       | |       |     ind   1 fus ind  2
 -------   -------            2          2      X.1    1  1  1
```

```
 -------   -------                                 X.2    1  1 -1
|       | |          X2  +   1   :  ++  1         X.3    2 -1  .
|  3.G  | | 3.G.2
|       | |              ind   1 fus ind  2
 -------                      3                  6.G.2
 -------   -------            3
|       | |                                        2   2  1  1  2  2  2
|  6.G  | | 6.G.2      X3 o2   1   *   +           3   1  1  1  1  .  .
|       | |
 -------                ind   1 fus ind  2             1a 6a 3a 2a 2b 2c
                              6          2          2P  1a 3a 3a 1a 1a 1a
                              3                     3P  1a 2a 1a 2a 2b 2c
                              2
                              3                  Y.1    1  1  1  1  1  1
                              6                  Y.2    1  1  1  1 -1 -1
                                                 Y.3    1 -1  1 -1  1 -1
                         X4 o2   1   *   +       Y.4    1 -1  1 -1 -1  1
                                                 Y.5    2 -1 -1  2  .  .
                                                 Y.6    2  1 -1 -2  .  .
```

The tables of G, $2.G$, $3.G$, $6.G$ and $G.2$ are known from the first example, that of $2.G.2$ will be given in the next one. So here we print only the GAP tables of $3.G.2 \cong D_6$ and $6.G.2 \cong D_{12}$.

In $3.G.2$, the characters `X.1`, `X.2` extend χ_1; χ_3 and its non-printed partner fuse to give `X.3`, and the two preimages of `1A` of order 3 collapse.

In $6.G.2$, `Y.1`–`Y.4` are extensions of χ_1, χ_2, so these characters are the inflated characters from $2.G.2$ (with respect to the factor fusion `[ 1, 2, 1, 2, 3, 4 ]`). `Y.5` is inflated from $3.G.2$ (with respect to the factor fusion `[ 1, 2, 2, 1, 3, 3 ]`), and `Y.6` is the result of the fusion of χ_4 and its non-printed partner.

For the last example, let G be the elementary abelian group 2^2 of order 4. Consider the following tables.

```
 -------   -------              ;   @   @   @   @   ;   ;   @
|       | |       |                4   4   4   4           1
|   G   | |  G.3  |          p power   A   A   A           A
|       | |       |          p' part   A   A   A           A
 -------   -------           ind  1A  2A  2B  2C fus ind  3A
 -------   -------
|       | |       |      chi1  +   1   1   1   1   : +oo   1
|  2.G  | | 2.G.3 |      chi2  +   1   1  -1  -1   .   +   0
|       | |       |      chi3  +   1  -1   1  -1   |
 -------   -------       chi4  +   1  -1  -1   1   |

                           ind   1   4   4   4 fus ind   3
                                 2                       6

                          chi5  -   2   0   0   0   : -oo   1
```

```
 G.3

    2   2   2   .   .
    3   1   .   1   1
```

```
      1a  2a  3a  3b
  2P  1a  1a  3b  3a
  3P  1a  2a  1a  1a

X.1    1   1   1   1
X.2    1   1   A  /A
X.3    1   1  /A   A
X.4    3  -1   .   .

A = E(3)
  = (-1+ER(-3))/2 = b3
```

```
2.G

   2   3   3   2   2   2
                       
      1a  2a  4a  4b  4c
  2P  1a  1a  2a  1a  1a
  3P  1a  2a  4a  4b  4c
                       
X.1    1   1   1   1   1
X.2    1   1   1  -1  -1
X.3    1   1  -1   1  -1
X.4    1   1  -1  -1   1
X.5    2  -2   .   .   .
```

```
2.G.3

   2   3   3   2   1   1   1   1
   3   1   1   .   1   1   1   1
                               
       1a  2a  4a  3a  6a  3b  6b
  2P   1a  1a  2a  3b  3b  3a  3a
  3P   1a  2a  4a  1a  2a  1a  2a
                               
X.1    1   1   1   1   1   1   1
X.2    1   1   1   A   A  /A  /A
X.3    1   1   1  /A  /A   A   A
X.4    3   3  -1   .   .   .   .
X.5    2  -2   .   1   1   1   1
X.6    2  -2   .   A  -A  /A -/A
X.7    2  -2   .  /A -/A   A  -A

A = E(3)
  = (-1+ER(-3))/2 = b3
```

In the table of $G.3 \cong A_4$, the characters χ_2, χ_3, and χ_4 fuse, and the classes `2A`, `2B` and `2C` collapse. For getting the table of $2.G \cong Q_8$, one just has to split the class `2A` and adjust the representative orders. Finally, the table of $2.G.3 \cong SL_2(3)$ is given; the class fusion corresponding to the injection $2.G \hookrightarrow 2.G.3$ is `[ 1, 2, 3, 3, 3 ]`, and the factor fusion corresponding to the epimorphism $2.G.3 \to G.3$ is `[ 1, 1, 2, 3, 3, 4, 4 ]`.

(The beautiful LaTeX pictures that were part of the GAP 3 manual will be reintroduced as soon as the decision to use TeX for the manual will be revised.)

2.7 CAS Tables

All character tables of the CAS table library (see [NPP84]) are available in GAP. This sublibrary has been completely revised before it was included in GAP, for example, errors have been corrected and power maps have been completed.

Any CAS table is accessible by each of its CAS names, that is, the table name or the filename used in CAS.

```
gap> tbl:= CharacterTable( "m10" );
CharacterTable( "A6.2_3" )
```

1 ▶ `CASInfo( `*tbl*` )` A

Let *tbl* be an ordinary character table *tbl* in the GAP library that was contained already in the CAS table library. When one fetches *tbl* from the library, one does in general not get the original CAS table. Namely,

in many cases (mostly ATLAS tables, see 2.5) the identifier of the table (see 69.8.11 in the GAP Reference Manual) as well as the ordering of classes and characters are different for the CAS table and its GAP version.

Note that in several cases, the CAS library contains different tables of the same group, in particular these tables may have different names and orderings of classes and characters.

The `CASInfo` value of *tbl*, if stored, is a list of records, each describing the relation between *tbl* and a character table in the CAS library. The records have the components

`name`
: the name of the CAS table,

`permchars` and `permclasses`
: permutations of the `Irr` values and the classes of *tbl*, respectively, that must be applied in order to get the orderings in the original CAS table, and

`text`
: the text that was stored on the CAS table (which may contain incorrect statements).

```
gap> HasCASInfo( tbl );
true
gap> CASInfo( tbl );
[ rec( name := "m10", permchars := (3,5)(4,8,7,6), permclasses := (),
      text := "names:     m10\norder:     2^4.3^2.5 = 720\nnumber of classes: \
8\nsource:    cambridge atlas\ncomments:  point stabilizer of mathieu-group m1\
1\ntest:      orth, min, sym[3]\n" ) ]
```

The class fusions stored on tables from the CAS library have been computed anew; the `text` component of such a fusion record tells if the fusion map is equal to that in the CAS library –of course modulo the permutation of classes between the table in CAS and its GAP version.

```
gap> First( ComputedClassFusions( tbl ), x -> x.name = "M11" );
rec( name := "M11", map := [ 1, 2, 3, 4, 5, 4, 7, 8 ],
  text := "fusion is unique up to table automorphisms,\nthe representative is \
equal to the fusion map on the CAS table" )
```

2.8 Organization of the Character Table Library

The data files of the GAP character table library reside in the `data` directory of the `ctbllib` package.

The filenames start with `ct` (for "character table"), followed by either `o` (for "ordinary"), `b` (for "Brauer"), or `g` (for "generic"), then a description of the contents (up to 5 characters, e.g., `alter` for the tables of alternating and related groups), and the suffix `.tbl`.

The file `ctb`*descr*`.tbl` contains the known Brauer tables corresponding to the ordinary tables in the file `cto`*descr*`.tbl`.

Each data file of the table library is supposed to consist of

1. comment lines, starting with in the first column,
2. assignments to `ALN` (short for "add library name", see 2.9.1) and to a component of `Revision`, at the beginning of the file, for example in the file with name `ctoalter.tbl` a value is assigned to `Revision.ctoalter_tbl`,
3. assignments to `ALN` and to a component of `LIBTABLE.LOADSTATUS`, at the end of the file, and
4. function calls of the form `SET_TABLEFILENAME(` *filename* `);`, `MBT(` *name*`,` *data* `);` ("make Brauer table"), `MOT(` *name*`,` *data* `);` ("make ordinary table"), `ALF(` *from*`,` *to*`,` *map* `);`, `ALF(` *from*`,` *to*`,`

map, *textlines*); ("add library fusion"), ALN(*name*, *listofnames*);, and ARC(*name*, *component*, *compdata*); ("add record component").

Here *filename* must be a string corresponding to the filename but without suffix, for example "ctoalter" if the file has the name ctoalter.tbl; *name* must be the identifier value of the ordinary character table corresponding to the table to which the command refers; *data* must be a comma separated sequence of GAP objects; *from* and *to* must be identifier values of ordinary character tables, *map* a list of positive itegers, *textlines* and *listofnames* lists list of strings, *component* a string, and *compdata* any GAP object.

MOT, ALF, ALN, and ARC occur only in files containing ordinary character tables, and MBT occurs only in files containing Brauer tables.

Besides the above calls, the data in files containing ordinary and Brauer tables may contain only the following GAP functions. (Files containing generic character tables may contain arbitrary GAP functions.)

ACM, Concatenation, ConstructClifford, ConstructDirectProduct, ConstructFactor, ConstructGS3, ConstructIsoclinic, ConstructMixed, ConstructPermuted, ConstructProj, ConstructSubdirect, ConstructV4G, E, EvalChars, GALOIS, Length, NotifyCharTableName, ShallowCopy, TENSOR, and TransposedMat.

The awk script maketbl in the etc directory of the ctbllib package expects the file format described above, and to some extent this format is checked by this script.

The function calls may be continued over several lines of a file. The ; is assumed to be the last character in its line if and only if it terminates a function call.

Names of character tables are strings (see Chapter 26 in the GAP Reference Manual), i.e., they are enclosed in double quotes; strings in table library files must not be split over several lines, because otherwise the awk script may get confused. Additionally, no character table name is allowed to contain double quotes.

The knowledge of GAP about the ordinary tables in the table library is contained in the file ctprimar.tbl (the "primary file" of the table library). This file is automatically produced from the library files by the script maketbl in the etc directory of the ctbllib package. The information is stored in the global variable LIBLIST, which is a record with the following components.

firstnames
: the list of Identifier (see 69.8.11 in the GAP Reference Manual) values of the ordinary tables,

files
: the list of filenames containing the data of ordinary tables,

filenames
: a list of positive integers, value j at position i means that the table whose identifier is the i–th in the firstnames list is contained in the j-th file of the files component,

fusionsource
: a list containing at position i the list of names of tables that store a fusion into the table whose identifier is the i–th in the firstnames list,

allnames
: a list of all admissible names of ordinary library tables,

position
: a list that stores at position i the position in firstnames of the identifier of the table with the i–th admissible name in allnames,

projections
: a list of triples [*name*, *factname*, *map*] describing a factor fusion *map* from the table with identifier *name* to the table with identifier *factname* (this is used to construct the table of *name* using the data of the table of *factname*),

`simpleinfo`
: a list of triples $[m, name, a]$ describing the tables of simple groups in the library; *name* is the identifier of the table, m`.`*name* and *name*`.`a are admissible names for its Schur multiplier and automorphism group, respectively,

`sporadicSimple`
: a list of identifiers of the tables of the 26 sporadic simple groups, and

`GENERIC`
: a record with information about generic tables (see 2.3).

There are three different ways how the table data can be stored in the file.

Full ordinary tables are encoded by a call to the function `MOT`, where the arguments correspond to the relevant attribute values; each fusion into another library table is added by a call to `ALF`, values to be stored in components of the table object are added with `ARC`, and admissible names are notified with `ALN`. The argument of `MOT` that encodes the irreducible characters is abbreviated as follows. For each subset of characters that differ just by multiplication with a linear character or by Galois conjugacy, only the first one is given by its values, the others are replaced by `[TENSOR,[`i`,`j`]]` (which means that the character is the tensor product of the i-th and the j-th character in the list) or `[GALOIS,[`i`,`j`]]` (which means that the character is obtained from the i-th character by applying `GaloisCyc( ., `j` )` to it.

Brauer tables are stored relative to the corresponding ordinary tables; attribute values that can be got by restriction from the ordinary table to p–regular classes are not stored, and instead of the irreducible characters the files contain (inverses of) decomposition matrices or Brauer trees for the blocks of nonzero defects.

Ordinary construction tables have a component `construction`, with value a function of one argument that is called by `CharacterTable` when the table is constructed (**not** when the file containing the table is read). The aim of this mechanism is to store structured character tables such as tables of direct products and tables of central extensions of other tables in a very compact way.

2.9 How to Extend the Character Table Library

GAP users may want to extend the character table library in different respects. Probably the easiest change is to add new admissible names to library tables, in order to use these names in calls of `CharacterTable` (see 69.3.1 in the GAP Reference Manual, and 2.2.1). This can be done as follows.

1 ▶ `NotifyNameOfCharacterTable(` *firstname*`,` *newnames* `)` F
▶ `ALN(` *firstname*`,` *newnames* `)` F

notifies the strings in the list *newnames* as new admissible names for the library table with `Identifier` value *firstname*, see 69.8.11 in the GAP Reference Manual. If there is already another library table for which some of these names are admissible then an error is signaled.

`NotifyNameOfCharacterTable` modifies the global variable `LIBLIST`.

`ALN` is a shorthand for `NotifyNameOfCharacterTable`. In those library files for which the `maketbl` script has produced the necessary information for `LIBLIST`, `ALN` is set to `Ignore` in the beginning and back to `NotifyNameOfCharacterTable` in the end.

```
gap> CharacterTable( "private" );
fail
gap> NotifyNameOfCharacterTable( "A5", [ "private" ] );
gap> a5:= CharacterTable( "private" );
CharacterTable( "A5" )
```

The next kind of changes is the addition of new fusions between library tables. Once a fusion map is known, it can be added to the library file containing the table of the subgroup, using the format produced by `LibraryFusion`.

2 ► `ALF(` *from*, *to*, *map*`[,` *text*`]` `)` F

`ALF` stores the fusion map *map* between the ordinary character tables with identifier strings *from* and *to* in the record encoding the table with identifier *from*. If the string *text* is given then it is added as `text` component of the fusion.

`ALF` changes the global list `LIBLIST.fusionsource`.

Note that the `ALF` statement should be placed in the file containing the data for the table with identifier *from*.

3 ► `LibraryFusion(` *name*, *fus* `)` F

For a string *name* that is an `Identifier` value (see 69.8.11 in the GAP Reference Manual) of an ordinary character table in the GAP library, and a record *fus* with components `name` (the identifier of the destination table), `map` (the fusion map, a list of image positions), and optionally `text` (a string containing information about the fusion), `LibraryFusion` returns a string whose printed value can be used to add the fusion in question to the library file containing the data for the table with identifier *name*.

name may also be a character table, in this case its `Identifier` value is used as string.

```
gap> s5:= CharacterTable( "S5" );
CharacterTable( "A5.2" )
gap> fus:= PossibleClassFusions( a5, s5 );
[ [ 1, 2, 3, 4, 4 ] ]
gap> fusion:= rec( name:= Identifier( s5 ), map:= fus[1], text:= "unique" );;
gap> Print( LibraryFusion( "A5", fusion ) );
ALF("A5","A5.2",[1,2,3,4,4],[
"unique"
]);
```

The last kind of changes is the addition of new character tables to the GAP character table library. Data files containing tables in library format (i.e., in the form of calls to `MOT` or `MBT`) can be produced using `PrintToLib`.

4 ► `PrintToLib(` *file*, *tbl* `)` F

prints the (ordinary or Brauer) character table *tbl* in library format to the file *file*`.tbl` or *file* (if this has already the suffix `.tbl`), respectively.

If *tbl* is an ordinary table then the value of the attribute `NamesOfFusionSources` is ignored by `PrintToLib`, since for library tables this information is extracted from the source files by the `maketbl` script.

```
gap> PrintToLib( "private", a5 );
```

The above command appends the data of the table `a5` to the file `private.tbl`; the first lines printed to this file are

```
SET_TABLEFILENAME("private");
MOT("A5",
[
"origin: ATLAS of finite groups, tests: 1.o.r., pow[2,3,5]"
],
[60,4,3,5,5],
[,[1,1,3,5,4],[1,2,1,5,4],,[1,2,3,1,1]],
[[1,1,1,1,1],[3,-1,0,-E(5)-E(5)^4,-E(5)^2-E(5)^3],
[GALOIS,[2,2]],[4,0,1,-1,-1],[5,1,-1,0,0]],
[(4,5)]);
ARC("A5","projectives",["2.A5",[[2,0,-1,E(5)+E(5)^4,E(5)^2+E(5)^3],
```

```
[GALOIS,[1,2]],[4,0,1,-1,-1],[6,0,0,1,1]],]);
ARC("A5","extInfo",["2","2"]);
```

If you have an ordinary character table in library format which you want to add to the table library, for example because it shall be accessible via `CharacterTable` (see 2.2.1), you must notify this table, i.e., tell GAP in which file it can be found, and which names shall be admissible for it.

5 ► `NotifyCharacterTable(` *firstname*`,` *filename*`,` *othernames* `)` F

notifies a new ordinary table to the library. This table has `identifier` component *firstname*, it is contained (in library format, see 2.9.4) in the file with name *filename* (without suffix `.tbl`, and relative to the `pkg/ctbllib/data` directory of the GAP installation), and the names contained in the list *othernames* are admissible for it.

`NotifyCharacterTable` modifies the global variable `LIBLIST` for the current GAP session, after having checked that there is no other library table yet with an admissible name equal to *firstname* or contained in *othernames*.

For example, let us change the name `A5` to `icos` wherever it occurs in the file `private.tbl` that was produced above, and then notify the "new" table in this file as follows. (The name change is needed because GAP knows already a table with name `A5` and would not accept to add another table with this name.)

```
gap> NotifyCharacterTable( "icos", "private", [] );
gap> icos:= CharacterTable( "icos" );
#W  revision entry missing in "private.tbl"
CharacterTable( "icos" )
gap> Display( icos );
icos

     2  2  2  .  .  .
     3  1  .  1  .  .
     5  1  .  .  1  1

       1a 2a 3a 5a 5b
    2P 1a 1a 3a 5b 5a
    3P 1a 2a 1a 5b 5a
    5P 1a 2a 3a 1a 1a

X.1     1  1  1  1  1
X.2     3 -1  .  A *A
X.3     3 -1  . *A  A
X.4     4  .  1 -1 -1
X.5     5  1 -1  .  .

A = -E(5)-E(5)^4
  = (1-ER(5))/2 = -b5
```

So the private table is treated as a library table. Note that the table can be accessed only if it has been notified in the current GAP session. For frequently used private tables, it may be reasonable to put the `NotifyCharacterTable` statements into your `.gaprc` file (see 3.4 in the GAP Reference Manual), or into a file that is read via the `.gaprc` file. For adding character tables to the GAP distribution, please send the tables to the e-mail address mentioned in the first paragraph of this chapter.

Bibliography

[BN95] Thomas Breuer and Simon P. Norton. *Improvements to the Atlas*, pages 297–327. Volume 11 of *London Math. Soc. Monographs* [JLPW95], 1995.

[Bre01] Thomas Breuer. *Manual for the GAP Character Table Library, Version 1.0*. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 2001.

[CCN+85] J[ohn] H. Conway, R[obert] T. Curtis, S[imon] P. Norton, R[ichard] A. Parker, and R[obert] A. Wilson. *Atlas of finite groups*. Oxford University Press, 1985.

[GAP01] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.3*, 2001.

`http://www.gap-system.org`.

[Han88] W[ilhelm] Hanrath. *Irreduzible Darstellungen von Raumgruppen*. Dissertation, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1988.

[HJLP] Gerhard Hiss, Christoph Jansen, Klaus Lux, and Richard [A.] Parker. *Computational Modular Character Theory*.

[HP89] Derek F. Holt and W[ilhelm] Plesken. *Perfect Groups*. Oxford Math. Monographs. Oxford University Press, 1989.

[JLPW95] Christoph Jansen, Klaus Lux, Richard [A.] Parker, and Robert [A.] Wilson. *An Atlas of Brauer Characters*, volume 11 of *London Math. Soc. Monographs*. Oxford University Press, 1995.

[LP91] Klaus Lux and Herbert Pahlings. Computational aspects of representation theory of finite groups. In G. O. Michler and C. R. Ringel, editors, *Representation theory of finite groups and finite-dimensional algebras*, volume 95 of *Progress in Mathematics*, pages 37–64. Birkhäuser, Basel, 1991.

[NPP84] J[oachim] Neubüser, H[erbert] Pahlings, and W[ilhelm] Plesken. CAS; design and use of a system for the handling of characters of finite groups. In Michael D. Atkinson, editor, *Computational Group Theory, Proceedings LMS Symposium on Computational Group Theory, Durham 1982*, pages 195–247. Academic Press, 1984.

[Ost86] Th[omas] Ostermann. Charaktertafeln von Sylownormalisatoren sporadischer einfacher Gruppen. Vorlesungen aus dem Fachbereich Mathematik 14, Universität Essen, Essen, Germany, 1986.

Index

This index covers only this manual. A page number in *italics* refers to a whole section which is devoted to the indexed subject. Keywords are sorted with case and spaces ignored, e.g., "`PermutationCharacter`" comes before "permutation group".