ResClasses

Set-Theoretic Computations with Residue Classes

Version 2.3.5

September 29, 2006

Stefan Kohl

Stefan Kohl — Email: `kohl@mathematik.uni-stuttgart.de`
— Homepage: `http://www.cip.mathematik.uni-stuttgart.de/ kohlsn`
— Address: Institut für Geometrie und Topologie
Pfaffenwaldring 57
Universität Stuttgart
70550 Stuttgart
Germany

Abstract

ResClasses is a package for GAP 4, which provides a fully-featured and easy-to-use implementation of residue classes of $\mathbb{Z}$ and a few other rings as sets. The package further provides slightly more specialized functionality for unions of residue classes with distinguished representatives and signed moduli. The ResClasses package is used in a group theoretical context by the RCWA package [Koh05].

Copyright

Contents

Chapter 1

Unions of Residue Classes

1.1 Entering residue classes and unions thereof

1.1.1 ResidueClass (R, m, r)

◊ `ResidueClass( R, m, r )` (function)
◊ `ResidueClass( m, r )` (function)
◊ `ResidueClass( r, m )` (function)

Returns: In the three-argument form the residue class `r` mod `m` of the ring `R`, and in the two-argument form the residue class `r` mod `m` of the integers.

In the two-argument case, `r` and `m` must not be negative, and `r` is assumed to lie in the range `[0..m-1]`. The latter is used to decide which of the two arguments is the modulus `m` and which is the residue `r`. For convenience, in the two-argument case it is permitted to enclose the argument list in list brackets.

Residue classes have the property `IsResidueClass`. Rings are regarded as residue class 0 (mod 1), and therefore have this property. There are operations `Modulus` and `Residue` to retrieve the modulus `m` resp. residue `r` of a residue class.

Example

```
gap> ResidueClass(2,3);
The residue class 2(3) of Z
gap> ResidueClass(Z_pi([2,5]),2,1);
The residue class 1(2) of Z_( 2, 5 )
gap> R := PolynomialRing(GF(2),1);;
gap> x := Indeterminate(GF(2),1);; SetName(x,"x");
gap> ResidueClass(R,x+One(R),Zero(R));
The residue class 0*Z(2) ( mod x+Z(2)^0 ) of GF(2)[x]
```

1.1.2 ResidueClassUnion (R, m, r)

◊ `ResidueClassUnion( R, m, r )` (function)
◊ `ResidueClassUnion( R, m, r, included, excluded )` (function)

Returns: The union of the residue classes `r`[*i*] mod `m` of the ring `R`, plus / minus finite sets `included` and `excluded` of elements of `R`.

```
Example
gap> ResidueClassUnion(Integers,6,[2,4]);
Union of the residue classes 2(6) and 4(6) of Z
gap> ResidueClassUnion(Integers,5,[1,2],[3,8],[-4,1]);
(Union of the residue classes 1(5) and 2(5) of Z) U [ 3, 8 ] \ [ -4, 1 ]
gap> ResidueClassUnion(Z_pi([2,3]),8,[3,5]);
<union of 2 residue classes (mod 8) of Z_( 2, 3 )>
gap> ResidueClassUnion(R,x^2,[One(R),x],[Zero(R)],[One(R)]);
<union of 2 residue classes (mod x^2) of GF(2)[x]> U [ 0*Z(2) ] \ [ Z(2)^0 ]
```

For extracting the components of a residue class union as passed as arguments to `ResidueClassUnion` (1.1.2), there are operations `Modulus`, `Residues`, `IncludedElements` and `ExcludedElements`.

The user has the choice between a longer and more descriptive and a shorter and less bulky output format for residue classes and unions thereof:

```
Example
gap> ResidueClassUnionViewingFormat("short");
gap> ResidueClassUnion(Integers,12,[0,1,4,7,8]);
0(4) U 1(6)
gap> ResidueClassUnionViewingFormat("long");
gap> ResidueClassUnion(Integers,12,[0,1,4,7,8]);
Union of the residue classes 0(4) and 1(6) of Z
```

1.1.3 AllResidueClassesModulo (R, m)

◇ `AllResidueClassesModulo( R, m )` (function)
◇ `AllResidueClassesModulo( m )` (function)

Returns: A sorted list of all residue classes (mod `m`) of the ring `R`.

If the argument `R` is omitted it defaults to the default ring of `m` – cp. the documentation of `DefaultRing` in the GAP reference manual. A transversal for the residue classes (mod `m`) can be obtained by the operation `AllResidues(R,m)`, and their number can be determined by the operation `NumberOfResidues(R,m)`.

```
Example
gap> AllResidueClassesModulo(Integers,2);
[ The residue class 0(2) of Z, The residue class 1(2) of Z ]
gap> AllResidueClassesModulo(Z_pi(2),2);
[ The residue class 0(2) of Z_( 2 ), The residue class 1(2) of Z_( 2 ) ]
gap> AllResidueClassesModulo(R,x);
[ The residue class 0*Z(2) ( mod x ) of GF(2)[x],
  The residue class Z(2)^0 ( mod x ) of GF(2)[x] ]
gap> AllResidues(Integers,6);
[ 0 .. 5 ]
gap> AllResidues(R,x^3);
[ 0*Z(2), Z(2)^0, x, x+Z(2)^0, x^2, x^2+Z(2)^0, x^2+x, x^2+x+Z(2)^0 ]
```

1.2 Methods for unions of residue classes

There are methods for `Print`, `String` and `Display` which are applicable to unions of residue classes. There is a method for `in` which tests whether some ring element lies in a given union of residue classes.

Example
```
gap> Print(ResidueClass(1,2),"\n");
ResidueClassUnion( Integers, 2, [ 1 ] )
gap> 1 in ResidueClass(1,2);
true
```

There are methods for `Union`, `Intersection`, `Difference` and `IsSubset` available for unions of residue classes. They also accept finite subsets of the base ring as arguments.

Example
```
gap> S := Union(ResidueClass(0,2),ResidueClass(0,3));
Z \ Union of the residue classes 1(6) and 5(6) of Z
gap> Intersection(S,ResidueClass(0,7));
Union of the residue classes 0(14) and 21(42) of Z
gap> Difference(S,ResidueClass(2,4));
Union of the residue classes 0(4) and 3(6) of Z
gap> IsSubset(ResidueClass(0,2),ResidueClass(4,8));
true
gap> Union(S,[1..10]);
(Union of the residue classes 0(2) and 3(6) of Z) U [ 1, 5, 7 ]
gap> Intersection(S,[1..10]);
[ 2, 3, 4, 6, 8, 9, 10 ]
gap> Difference(S,[1..10]);
(Union of the residue classes 0(2) and 3(6) of Z) \ [ 2, 3, 4, 6, 8, 9, 10 ]
gap> Difference(Integers,[1..10]);
Z \ <set of cardinality 10>
gap> IsSubset(S,[1..10]);
false
```

If the underlying ring has a residue class ring of a given cardinality t, then a residue class can be written as a disjoint union of t residue classes with equal moduli:

1.2.1 SplittedClass (cl, t)

◊ `SplittedClass( cl, t )` (operation)

Returns: A partition of the residue class `cl` into `t` residue classes with equal moduli, provided that such a partition exists. Otherwise `fail`.

Example
```
gap> SplittedClass(ResidueClass(1,2),2);
[ The residue class 1(4) of Z, The residue class 3(4) of Z ]
gap> SplittedClass(ResidueClass(Z_pi(3),3,0),2);
fail
```

Often one needs a partition of a given union of residue classes into "few" residue classes. Ensuring to get always a partition of minimal possible length seems to be algorithmically difficult. The following operation finds usually "reasonably short" partitions:

1.2.2 AsUnionOfFewClasses (U)

◊ `AsUnionOfFewClasses( U )` (operation)

Returns: A set of disjoint residue classes whose union is `U`.

As the name of the operation suggests, it is taken care that the number of residue classes in the returned list is kept "reasonably small". It is not necessarily minimal. No care is taken of `IncludedElements` and `ExcludedElements`.

Example
```
gap> AsUnionOfFewClasses(Difference(Integers,ResidueClass(0,12)));
[ The residue class 1(2) of Z, The residue class 2(4) of Z,
  The residue class 4(12) of Z, The residue class 8(12) of Z ]
```

One can compute the sets of sums, differences, products and quotients of the elements of a union of residue classes and an element of the base ring:

Example
```
gap> ResidueClass(0,2) + 1;
The residue class 1(2) of Z
gap> ResidueClass(0,2) - 2 = ResidueClass(0,2);
true
gap> 3 * ResidueClass(0,2);
The residue class 0(6) of Z
gap> ResidueClass(0,2)/2;
Integers
```

1.2.3 Density (U)

◊ `Density( U )` (operation)

Returns: The natural density of `U` as a subset of the underlying ring.

The *natural density* of a residue class $r(m)$ of a ring R is defined by $1/|R/mR|$, and the *natural density* of a union U of finitely many residue classes is defined by the sum of the densities of the elements of a partition of U into finitely many residue classes.

Example
```
gap> Density(ResidueClass(0,2));
1/2
gap> Density(Difference(Integers,ResidueClass(0,5)));
4/5
```

1.2.4 RandomPartitionIntoResidueClasses (R, length, primes)

◊ `RandomPartitionIntoResidueClasses( R, length, primes )` (operation)

Returns: A “random” partition of the ring `R` into `length` residue classes whose moduli have only prime factors in `primes`, resp. `fail` if no such partition exists.

```
Example
gap> ResidueClassUnionViewingFormat("short");
gap> RandomPartitionIntoResidueClasses(Integers,30,[2,3,5,7]);
[ 2(7), 3(7), 4(7), 5(7), 6(7), 1(35), 8(35), 14(35), 21(35), 22(35), 28(35),
  29(35), 7(105), 15(105), 42(105), 50(105), 77(105), 85(105), 70(175),
  105(175), 35(350), 210(350), 0(525), 140(525), 315(525), 350(525),
  490(525), 175(1575), 700(1575), 1225(1575) ]
gap> Union(last);
Integers
gap> Sum(List(last2,Density));
1
```

For looping over unions of residue classes of the integers, there are methods for the operations `Iterator` and `NextIterator`.

1.3 The categories and families of unions of residue classes

1.3.1 IsUnionOfResidueClasses (U)

◊ `IsUnionOfResidueClasses( U )` (filter)
◊ `IsUnionOfResidueClassesOfZ( U )` (filter)
◊ `IsUnionOfResidueClassesOfZ_pi( U )` (filter)
◊ `IsUnionOfResidueClassesOfGFqx( U )` (filter)

Returns: `true` if `U` is a union of residue classes resp. a union of residue classes of the ring of integers resp. a union of residue classes of a semilocalization of the ring of integers resp. a union of residue classes of a polynomial ring in one variable over a finite field, and `false` otherwise.

Often the same methods can be used for residue classes of the ring of integers and of its semilocalizations. For this reason there is a category `IsUnionOfResidueClassesOfZorZ_pi` which is the union of `IsUnionOfResidueClassesOfZ` and `IsUnionOfResidueClassesOfZ_pi`. The internal representation of unions of residue classes is called `IsResidueClassUnionSparseRep`.

1.3.2 ResidueClassUnionsFamily (R)

◊ `ResidueClassUnionsFamily( R )` (function)
◊ `ResidueClassUnionsFamily( R, fixedreps )` (function)

Returns: The family of unions of residue classes resp. the family of unions of residue classes with fixed representatives of the ring `R`, depending on whether `fixedreps` is present and `true`.

The ring `R` can be retrieved as `UnderlyingRing(ResidueClassUnionsFamily(R))`. Unions of residue classes with fixed representatives are described in the next chapter.

Chapter 2

Unions of Residue Classes with Fixed Representatives

ResClasses supports computations with unions of residue classes which are endowed with distinguished ("fixed") representatives. These unions of residue classes can be viewed as multisets of ring elements. The residue classes forming such a union do not need to be disjoint or even only distinct.

2.1 Entering unions of residue classes with fixed representatives

2.1.1 ResidueClassWithFixedRep (R, m, r)

◊ `ResidueClassWithFixedRep( R, m, r )` (function)
◊ `ResidueClassWithFixedRep( m, r )` (function)

Returns: The residue class `r` mod `m` of the ring `R`, with the fixed representative `r`.

If the argument `R` is omitted, it defaults to `Integers`.

Example

```
gap> ResidueClassWithFixedRep(Integers,2,1);
[1/2]
```

2.1.2 ResidueClassUnionWithFixedReps (R, classes)

◊ `ResidueClassUnionWithFixedReps( R, classes )` (function)
◊ `ResidueClassUnionWithFixedReps( classes )` (function)

Returns: The union of the residue classes `classes`[*i*][2] mod `classes`[*i*][1] of the ring `R`, with fixed representatives `classes`[*i*][2].

The argument `classes` must be a list of pairs of elements of the ring `R`. Their first entries – the moduli – must be nonzero. If the argument `R` is omitted, it defaults to `Integers`.

Example

```
gap> ResidueClassUnionWithFixedReps(Integers,[[2,4],[3,9]]);
[4/2] U [9/3]
```

There is a method for the operation `Modulus` which returns the lcm of the moduli of the residue classes forming such a union. Further there is an operation `Classes` for retrieving the list of classes which has been passed as an argument to `ResidueClassUnionWithFixedReps`. The operation `AsListOfClasses` does the same except that the returned list contains residue classes instead of pairs `[modulus,residue]`. There are methods for `Print`, `String` and `Display` available for unions of residue classes with fixed representatives.

2.1.3 AllResidueClassesWithFixedRepsModulo (R, m)

◊ `AllResidueClassesWithFixedRepsModulo( R, m )` (function)
◊ `AllResidueClassesWithFixedRepsModulo( m )` (function)

Returns: A sorted list of all residue classes (mod `m`) of the ring `R`, with fixed representatives.

If the argument `R` is omitted it defaults to the default ring of `m`, cp. the documentation of `DefaultRing` in the GAP reference manual. The representatives are the same as those chosen by the operation `mod`. See also `AllResidueClassesModulo` (1.1.3).

Example
```
gap> AllResidueClassesWithFixedRepsModulo(Z_pi(2),4);
[ [0/4], [1/4], [2/4], [3/4] ]
gap> AllResidueClassesWithFixedRepsModulo(9);
[ [0/9], [1/9], [2/9], [3/9], [4/9], [5/9], [6/9], [7/9], [8/9] ]
```

2.2 Methods for unions of residue classes with fixed representatives

Throughout this chapter, the argument `R` denotes the underlying ring, and the arguments `U`, `U1` and `U2` denote unions of residue classes of `R` with fixed representatives.

Unions of residue classes with fixed representatives are multisets. Elements and residue classes can be contained with multiplicities:

2.2.1 Multiplicity (x, U)

◊ `Multiplicity( x, U )` (method)
◊ `Multiplicity( cl, U )` (method)

Returns: The multiplicity of `x` in `U` regarded as a multiset of ring elements, resp. the multiplicity of the residue class `cl` in `U` regarded as a multiset of residue classes.

Example
```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> List([0..23],n->Multiplicity(n,U));
[ 2, 0, 1, 1, 1, 0, 2, 0, 1, 1, 1, 0, 2, 0, 1, 1, 1, 0, 2, 0, 1, 1, 1, 0 ]
gap> Multiplicity(ResidueClassWithFixedRep(2,0),U);
1
```

Let `U` be a union of residue classes with fixed representatives. The multiset `U` can have an attribute `Density` which denotes its *natural density* as a multiset, i.e. elements with multiplicity k count k-fold. The multiset `U` has the property `IsOverlappingFree` if it consists of pairwise disjoint residue classes. The set-theoretic union of the residue classes forming `U` can be determined by the operation `AsOrdinaryUnionOfResidueClasses`. The object returned by this operation is an “ordinary” residue class union as described in Chapter 1.

Example
```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> Density(U);
5/6
gap> IsOverlappingFree(U);
false
gap> AsOrdinaryUnionOfResidueClasses(U);
Z \ Union of the residue classes 1(6) and 5(6) of Z
gap> Density(last);
2/3
```

In the sequel we abbreviate the term “the multiset of ring elements endowed with the structure of a union of residue classes with fixed representatives” by “the multiset”.

There are methods for `+` and `-` available for computing the multiset of sums $u+x$, $u \in U$, the multiset of differences $u-x$ resp. $x-u$, $u \in U$ and the multiset of the additive inverses of the elements of U. Further there are methods for `*` and `/` available for computing the multiset of products $x \cdot u$, $u \in U$ and the multiset of quotients u/x, $u \in U$. The division method requires all elements of `U` to be divisible by x. If the underlying ring is the ring of integers, scalar multiplication and division leave δ invariant (→ `Delta` (2.3.1)).

Example
```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> U + 7;
[7/2] U [7/3]
gap> U - 7; 7 - U; -U;
[-7/2] U [-7/3]
[7/-3] U [7/-2]
[0/-3] U [0/-2]
gap> V := 2 * U;
[0/4] U [0/6]
gap> V/2;
[0/2] U [0/3]
```

2.2.2 Union (U1, U2)

◇ `Union( U1, U2 )` (method)

Returns: The union of `U1` and `U2`.

The multiplicity of any ring element resp. residue class in the union is the sum of its multiplicities in the arguments. It holds `Delta(Union(U1,U2)) = Delta(U1) + Delta(U2)`. (→ `Delta` (2.3.1)).

Example
```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> Union(U,U);
[0/2] U [0/2] U [0/3] U [0/3]
```

2.2.3 Intersection (U1, U2)

◇ `Intersection( U1, U2 )` (method)

Returns: The intersection of `U1` and `U2`.

The multiplicity of any residue class in the intersection is the minimum of its multiplicities in the arguments.

Example
```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> Intersection(U,ResidueClassWithFixedRep(2,0));
[0/2]
gap> Intersection(U,ResidueClassWithFixedRep(6,0));
Empty union of residue classes of Z with fixed representatives
```

2.2.4 Difference (U1, U2)

◇ `Difference( U1, U2 )` (method)

Returns: The difference of `U1` and `U2`.

The multiplicity of any residue class in the difference is its multiplicity in `U1` minus its multiplicity in `U2`, if this value is nonnegative. The difference of the empty residue class union with fixed representatives and some residue class $[r/m]$ is set equal to $[(m-r)/m]$. It holds `Delta(Difference(U1,U2)) = Delta(U1) - Delta(U2)`. (→ `Delta` (2.3.1)).

Example
```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> V := ResidueClassUnionWithFixedReps(Integers,[[3,0],[5,2]]);
[0/3] U [2/5]
gap> Difference(U,V);
[0/2] U [3/5]
```

2.3 The invariant Delta

2.3.1 Delta (U)

◊ Delta(U) (attribute)

Returns: The value of the invariant δ of the residue class union U.

For a residue class $[r/m]$ with fixed representative we set $\delta([r/m]) := r/m - 1/2$ and extend this definition additively to unions of such residue classes. If no representatives are fixed, this definition is still unique (mod 1). There is a related invariant ρ which is defined by $e^{\pi i \delta(U)}$. The corresponding attribute is called Rho.

Example

```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,3],[3,4]]);
[3/2] U [4/3]
gap> Delta(U) = (3/2-1/2) + (4/3-1/2);
true
gap> V := RepresentativeStabilizingRefinement(U,3);
[3/6] U [5/6] U [7/6] U [4/9] U [7/9] U [10/9]
gap> Delta(V) = Delta(U);
true
gap> Rho(V);
E(12)^11
```

2.3.2 RepresentativeStabilizingRefinement (U, k)

◊ RepresentativeStabilizingRefinement(U, k) (method)

Returns: The representative stabilizing refinement of U into k parts.

The *representative stabilizing refinement* of a residue class $[r/m]$ of $\mathbb{Z}$ into k parts is defined by $[r/km] \cup [(r+m)/km] \cup \ldots \cup [(r+(k-1)m)/km]$. This definition is extended in the obvious way to unions of residue classes.

If the argument k is zero, the method performs a simplification of U by joining appropriate residue classes, if this is possible.

In any case the value of Delta(U) is invariant under this operation (→ Delta (2.3.1)).

Example

```
gap> U := ResidueClassUnionWithFixedReps(Integers,[[2,0],[3,0]]);
[0/2] U [0/3]
gap> RepresentativeStabilizingRefinement(U,4);
[0/8] U [2/8] U [4/8] U [6/8] U [0/12] U [3/12] U [6/12] U [9/12]
gap> RepresentativeStabilizingRefinement(last,0);
[0/2] U [0/3]
```

2.4 The categories of unions of residue classes with fixed rep's

The names of the categories of unions of residue classes with fixed representatives are derived from the names of those of the "ordinary" unions of residue classes given in Section 1.3 by appending WithFixedRepresentatives.

Chapter 3

Semilocalizations of the Integers

In this package, the semilocalizations $\mathbb{Z}_{(\pi)}$ of the ring of integers are used as base rings for unions of residue classes.

3.1 Entering semilocalizations of the integers

3.1.1 Z_pi (pi)

◊ `Z_pi( pi )` (function)
◊ `Z_pi( p )` (function)

Returns: The ring $\mathbb{Z}_{(\pi)}$ resp. the ring $\mathbb{Z}_{(p)}$.

The returned ring has the property `IsZ_pi`. The set `pi` of noninvertible primes can be retrieved by the operation `NoninvertiblePrimes`.

Example
```
gap> R := Z_pi(2);
Z_( 2 )
gap> S := Z_pi([2,5,7]);
Z_( 2, 5, 7 )
```

3.2 Methods for semilocalizations of the integers

There are methods for the operations `in`, `Intersection`, `IsSubset`, `StandardAssociate`, `Gcd`, `Lcm`, `Factors` and `IsUnit` available for semilocalizations of the integers. For the documentation of these operations, see the GAP reference manual. The standard associate of an element of a ring $\mathbb{Z}_{(\pi)}$ is defined by the product of the noninvertible prime factors of its numerator.

Example
```
gap> 4/7 in R; 3/2 in R;
true
false
gap> Intersection(R,Z_pi([3,11])); IsSubset(R,S);
Z_( 2, 3, 11 )
true
```

Example

```
gap> StandardAssociate(R,-6/7);
2
gap> Gcd(S,90/3,60/17,120/33);
10
gap> Lcm(S,90/3,60/17,120/33);
40
gap> Factors(R,840);
[ 105, 2, 2, 2 ]
gap> Factors(R,-2/3);
[ -1/3, 2 ]
gap> IsUnit(S,3/11);
true
```

Chapter 4

Installation and auxiliary functions

4.1 Requirements

The ResClasses package needs at least GAP 4.4.7 and GAPDoc 0.999 [LN02]. It can be used under UNIX, under Windows and on the MacIntosh. ResClasses is completely written in the GAP language and does neither contain nor require external binaries.

4.2 Installation

Like any other GAP package, ResClasses must be installed in the `pkg` subdirectory of the GAP distribution. This is accomplished by extracting the distribution file in this directory. By default, the package ResClasses is autoloaded. If you have switched autoloading of packages off, you can load ResClasses via `LoadPackage( "resclasses" );`.

4.3 The testing routine

4.3.1 ResClassesTest

◊ `ResClassesTest( )` (function)

Returns: Nothing.

Performs tests of the ResClasses package. Errors, i.e. differences to the correct results of the test computations, are reported. The processed test files are in the directory `pkg/resclasses/tst`.

4.4 Building the manual

4.4.1 ResClassesBuildManual

◊ `ResClassesBuildManual( )` (function)

Returns: Nothing.

This function is a development tool which builds the manual of the ResClasses package in the file formats LaTeX, DVI, Postscript, PDF, HTML and ASCII text. This is accomplished using the GAPDoc package by Frank Lübeck and Max Neunhöffer. Building the manual is possible only on UNIX systems and requires LaTeX, PDFLaTeX and dvips. As all files generated by this function are included in the distribution file anyway, users will not need it.

References

[Koh05] Stefan Kohl. *RCWA - Residue Class-Wise Affine Groups*, 2005. GAP package, available at http://www.gap-system.org/Packages/rcwa.html. 2

[LN02] Frank Lübeck and Max Neunhöffer. *GAPDoc (version 0.99)*. RWTH Aachen, 2002. GAP package, available at http://www.math.rwth-aachen.de/ Frank.Luebeck. 17

Index