ResClasses

Computations with Residue Classes and their Set-Theoretic Unions

(Version 1.1.1)

February 3, 2004

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
Universität Stuttgart
70550 Stuttgart
Germany

Abstract

This package for GAP 4 (at least version 4.4) implements set-theoretic unions of residue classes of the ring $\mathbb{Z}$ of the integers, of its semilocalisations $\mathbb{Z}_{(\pi)}$ at some finite set of primes π and of the polynomial rings $\mathrm{GF}(q)[x]$ for some prime power q as GAP domains and provides basic functionality for computing with these sets (intersection, union, difference etc., any of these also when one of the operands is a finite set of elements). It also implements the above-mentioned rings $\mathbb{Z}_{(\pi)}$ as GAP domains.

Copyright

Contents

Chapter 1

Preface

Although all functionality provided by this package is mathematically trivial, the author thinks that having available a nice and easy-to-use implementation of residue classes as sets which permits to form unions, intersections and differences and which interacts smoothly with finite sets of elements w.r.t. these operations is often useful. The functionality of this package is used in a group theoretical context by the RCWA package written by the same author.

I would be grateful for any bug reports, comments or suggestions.

Stefan Kohl

Chapter 2

Semilocalizations of the Integers

In the following we need the semilocalizations $\mathbb{Z}_{(\pi)}$ of the ring of integers as base rings for unions of residue classes. Since these rings are not already implemented as domains in the GAP library, we had to implement them in this package.

2.1 Defining semilocalizations of the integers

2.1.1 Z_pi

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

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

The function also accepts a single prime p instead of the one-element list pi = [p] as argument.

Example

```
gap> R := Z_pi(2);
Z_( 2 )
gap> S := Z_pi([2,5,7]);
Z_( 2, 5, 7 )
gap> T := Z_pi([3,11]);
Z_( 3, 11 )
```

2.1.2 IsZ_pi

◊ IsZ_pi(R) (property)

Indicates whether R is a ring $\mathbb{Z}_{(\pi)}$ for some set of primes π.

2.1.3 NoninvertiblePrimes

◊ NoninvertiblePrimes(R) (attribute)

The noninvertible primes pi in the semilocalization R of the integers.

2.2 Methods for semilocalizations of the integers

2.2.1 \in

◊ `\in( x, R )` (method)

Checks whether `x` is an element of the semilocalization `R` of the integers.

Example

```
gap> 4/7 in R;
true
gap> 3/2 in R;
false
gap> 17/35 in S;
false
gap> 3/17 in S;
true
```

2.2.2 Intersection

◊ `Intersection( R, S )` (method)

Returns: the intersection of the semilocalizations `R` and `S` of the integers.

Example

```
gap> U := Intersection(R,S,T);
Z_( 2, 3, 5, 7, 11 )
```

2.2.3 IsSubset

◊ `IsSubset( R, S )` (method)

Checks whether `S` is a subring of `R`, where `R` and `S` are semilocalizations of the integers.

Example

```
gap> IsSubset(R,U);
true
gap> IsSubset(T,R);
false
```

2.2.4 StandardAssociate

◊ `StandardAssociate( R, x )` (method)

Returns: the standard associate of `x` in the semilocalization `R` of the integers.

We define the standard associate of an element of $\mathbb{Z}_{(\pi)}$ by the product of the non-invertible prime factors of its numerator.

Example

```
gap> StandardAssociate(R,-6/7);
2
gap> StandardAssociate(R,36/5);
4
gap> StandardAssociate(U,37/13);
1
gap> StandardAssociate(U,36/13);
36
```

2.2.5 GcdOp

◊ `GcdOp( R, x, y )` (method)

◊ `Gcd( R, x, y )` (method)

Returns: the greatest common divisor of `x` and `y` in the semilocalization `R` of the integers.

Example

```
gap> GcdOp(S,-10/3,8/13);
2
gap> Gcd(S,90/3,60/17,120/33);
10
```

2.2.6 LcmOp

◊ `LcmOp( R, x, y )` (method)

◊ `Lcm( R, x, y )` (method)

Returns: the least common multiple of `x` and `y` in the semilocalization `R` of the integers.

Example

```
gap> LcmOp(S,-10/3,8/13);
40
gap> Lcm(S,90/3,60/17,120/33);
40
```

2.2.7 Factors

◊ `Factors( R, x )` (method)

Returns: a prime factorization of `x` in the semilocalization `R` of the integers.

Example

```
gap> Factors(U,840);
[ 2, 2, 2, 3, 5, 7 ]
gap> Factors(R,840);
[ 105, 2, 2, 2 ]
gap> Factors(R,-2/3);
```

```
[ -1/3, 2 ]
gap> Factors(S,60/17);
[ 3/17, 2, 2, 5 ]
```

2.2.8 IsUnit

◊ `IsUnit( R, x )` (method)

Checks whether `x` is a unit in the semilocalization `R` of the integers. The method returns `fail` if `x` is not an element of `R`.

Example

```
gap> IsUnit(S,3/11);
true
gap> IsUnit(T,-2);
true
gap> IsUnit(T,0);
false
gap> IsUnit(T,3);
false
gap> IsUnit(T,3/11);
fail
```

Chapter 3

Unions of Residue Classes

3.1 Defining unions of residue classes

3.1.1 ResidueClass

◊ ResidueClass(R, m, r) (function)

The residue class r mod m of the ring R.

Example

```
gap> A := ResidueClass(Integers,3,2);
The residue class 2(3) of Z
gap> B := ResidueClass(Z_pi([2,5]),2,1);
The residue class 1(2) of Z_( 2, 5 )
gap> B = ResidueClass(Integers,2,1);
false
gap> R := PolynomialRing(GF(7),1);;
gap> x := Indeterminate(GF(7),1);; SetName(x,"x");
gap> C := ResidueClass(R,x+One(R),3*One(R));
The residue class Z(7) ( mod x+Z(7)^0 ) of GF(7)[x]
```

3.1.2 ResidueClassUnion

◊ ResidueClassUnion(R, m, r) (function)

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

The union of the residue classes r[*i*] mod m of the ring R. If the arguments included and excluded are given, they denote lists of elements to be added to resp. to be taken away from the union.

Example

```
gap> D := ResidueClassUnion(Integers,6,[2,4]);
Union of the residue classes 2(6) and 4(6) of Z
gap> F := 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> G := ResidueClassUnion(R,x,[One(R),5*One(R),6*One(R)],[Zero(R)],[One(R)]);
```

```
<union of 3 residue classes (mod x) of GF(7)[x]> U [ 0*Z(7) ] \ [ Z(7)^0 ]
gap> H := ResidueClassUnion(Z_pi([2,3]),8,[3,5]);
<union of 2 residue classes (mod 8) of Z_( 2, 3 )>
```

The components of a residue class union as given as arguments in `ResidueClassUnion` (3.1.2) can be extracted as follows.

3.1.3 Modulus

◇ `Modulus( U )` (method)

Returns: the modulus `m` of `U`.

3.1.4 Residues

◇ `Residues( U )` (operation)

Returns: the set of residues `r` of `U`.

If an element `el` of the underlying ring of `U` is congruent to one of the residues of `U` modulo the modulus of `U`, it is contained in `U`, provided that it is not excluded explicitly (see `ExcludedElements` (3.1.6)).

3.1.5 IncludedElements

◇ `IncludedElements( U )` (operation)

Returns: the set `included` of single elements of `U`.

3.1.6 ExcludedElements

◇ `ExcludedElements( U )` (operation)

Returns: the set `excluded` of single elements explicitly not contained in `U`.

3.2 Methods for unions of residue classes

3.2.1 String

◇ `String( U )` (method)
◇ `String( U, lng )` (method)

Returns: a string representation of the residue class union `U`.

If the argument `lng` is given its absolute value denotes the length of the returned string; if `lng` is negative, the result is flushed left, otherwise it is flushed right.

Example

```
gap> String(C);
"ResidueClassUnion( GF(7)[x], x+Z(7)^0, [ Z(7) ] )"
gap> String(F);
"ResidueClassUnion( Integers, 5, [ 1, 2 ], [ 3, 8 ], [ -4, 1 ] )"
gap> String(H);
"ResidueClassUnion( Z_( 2, 3 ), 8, [ 3, 5 ] )"
```

3.2.2 Print

◊ `Print( U )` (method)

Prints the residue class union `U` in a way similar to what a user has to write in order to create this object.

Example

```
gap> Print(C,"\n");
ResidueClassUnion( GF(7)[x], x+Z(7)^0, [ Z(7) ] )
gap> Print(F,"\n");
ResidueClassUnion( Integers, 5, [ 1, 2 ], [ 3, 8 ], [ -4, 1 ] )
gap> Print(H,"\n");
ResidueClassUnion( Z_( 2, 3 ), 8, [ 3, 5 ] )
```

3.2.3 Display

◊ `Display( U )` (method)

Displays the residue class union `U` in a nice, human-readable form.

Example

```
gap> Display(F);

The union of the residue classes r ( mod 5 ) of Z for r =

 1 2

and the elements

 3 8

without the elements

 -4  1

gap> Display(G);

The union of the residue classes r ( mod x ) of GF(7)[x] for r =

Z(7)^0  -Z(7)^0 Z(7)^5

and the element

0*Z(7)

without the element

Z(7)^0

gap> Display(H);
```

```
The union of the residue classes r ( mod 8 ) of Z_( 2, 3 ) for r =

 3 5

```

3.2.4 \in

◊ `\in( el, U )` (method)

Tests whether `el` is an element of the residue class union `U`.

Example

```
gap> 20 in A;
true
gap> -20 in A;
false
gap> 1/3 in B;
true
gap> 1/5 in B;
false
gap> x in G;
false
gap> Zero(R) in G;
true
```

3.2.5 IsSubset

◊ `IsSubset( U1, U2 )` (method)

Checks whether the residue class union `U2` is a subset of the residue class union `U1`.

Example

```
gap> IsSubset(A,D);
false
gap> IsSubset(H,ResidueClass(Z_pi([2,3]),16,11));
true
```

3.2.6 Density

◊ `Density( U )` (operation)

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

Example

```
gap> G;
<union of 3 residue classes (mod x) of GF(7)[x]> U [ 0*Z(7) ] \ [ Z(7)^0 ]
gap> Density(G);
```

```
3/7
gap> ResidueClassUnion(Integers,12,[3,5,9]);
Union of the residue classes 3(6) and 5(12) of Z
gap> Density(last);
1/4
gap> 2*last2;
Union of the residue classes 6(12) and 10(24) of Z
gap> Density(last);
1/8
```

3.2.7 Union

◊ Union(U1, U2) (method)

◊ Union(U, S) (method)

Returns: the union of two residue class unions U1 and U2 resp. of the residue class union U and the finite set S of elements of the ring U is defined over.

Example

```
gap> I := ResidueClassUnion(Integers,6,[1,5]);
Union of the residue classes 1(6) and 5(6) of Z
gap> J := ResidueClassUnion(Integers,5,[1,2,3,4]);
Union of the residue classes 1(5), 2(5), 3(5) and 4(5) of Z
gap> K := Union(I,J);
<union of 26 residue classes (mod 30) of Z>
gap> Residues(K);
[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24,
  25, 26, 27, 28, 29 ]
gap> Union(K,[0]);
<union of 26 residue classes (mod 30) of Z> U [ 0 ]
gap> Union(D,I);
Union of the residue classes 1(3) and 2(3) of Z
```

3.2.8 Intersection

◊ Intersection(U1, U2) (method)

◊ Intersection(U, S) (method)

Returns: the intersection of two residue class unions U1 and U2 resp. of the residue class union U and the finite set S of elements of the ring U is defined over.

Example

```
gap> L := Intersection(I,J);
<union of 8 residue classes (mod 30) of Z>
gap> Display(L);

The union of the residue classes r ( mod 30 ) of Z for r =

  1  7 11 13 17 19 23 29
```

```
gap> cl := List([1..25],i->ResidueClass(Integers,Primes[i],i));;
gap> cl_int := Intersection(cl);
The residue class 941584379775558526136539054851975983(
2305567963945518424753102147331756070) of Z
gap> List(Primes{[1..25]},p->Representative(cl_int) mod p);
[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
  22, 23, 24, 25 ]
```

3.2.9 Difference

◊ Difference(U1, U2) (method)

◊ Difference(U, S) (method)

Returns: the difference of two residue class unions U1 and U2 resp. of the residue class union U and the finite set S of elements of the ring U is defined over.

Example

```
gap> M := Difference(I,J);
Union of the residue classes 5(30) and 25(30) of Z
gap> N := Difference(J,I);
<union of 16 residue classes (mod 30) of Z>
gap> Display(N);

The union of the residue classes r ( mod 30 ) of Z for r =

  2  3  4  6  8  9 12 14 16 18 21 22 24 26 27 28

gap> Difference(Integers,[1,2,3]);
Z \ [ 1, 2, 3 ]
gap> Difference(Z_pi([2,3,7]),[1/5,1/55]);
Z_( 2, 3, 7 ) \ [ 1/55, 1/5 ]
```

3.2.10 Iterator

◊ Iterator(U) (method)

Returns: an iterator of the residue class union U.

Currently, iterators are only implemented for residue class unions of the integers.

Example

```
gap> it := Iterator(F);
<iterator of a residue class union of Z>
```

3.2.11 NextIterator

◊ NextIterator(iter) (method)

Returns: the next element delivered by the iterator iter of some residue class union.

Currently, iterators are only implemented for residue class unions of the integers.

Example

```
gap> l := List([1..16],i->NextIterator(it));
[ 3, 8, 2, -3, 6, -9, 7, -8, 11, -14, 12, -13, 16, -19, 17, -18 ]
gap> Difference(l,Intersection(F,[-20..20]));
[  ]
```

3.2.12 \+

◊ \+(U, x) (method)

◊ \+(x, U) (method)

Returns: the set of sums $u+x$, $u \in U$.

Example

```
gap> Display(L+1);

The union of the residue classes r ( mod 30 ) of Z for r =

  0  2  8 12 14 18 20 24

gap> L+30 = L;
true
```

3.2.13 \-

◊ \-(U, x) (method)

◊ \-(x, U) (method)

◊ \-(U) (method)

Returns: the set of differences $u-x$, $u \in U$ resp. the set of differences $x-u$, $u \in U$ resp. the set of the additive inverses of the elements of U.

Example

```
gap> F-7;
(Union of the residue classes 0(5) and 4(5) of Z) U [ -4, 1 ] \ [ -11, -6 ]
gap> -L = L;
true
gap> -C;
The residue class Z(7)^4 ( mod x+Z(7)^0 ) of GF(7)[x]
```

3.2.14 *

◊ *(U, x) (method)

◊ *(x, U) (method)

Returns: the set of products $x \cdot u$, $u \in U$.

Example

```
gap> 2*A;
```

```
The residue class 4(6) of Z
gap> D*17;
Union of the residue classes 34(102) and 68(102) of Z
gap> x^3*C;
The residue class Z(7)*x^3 ( mod x^4+x^3 ) of GF(7)[x]
gap> 2*Difference(Integers,[1,2,3]);
(The residue class 0(2) of Z) \ [ 2, 4, 6 ]
```

3.2.15 \/

◊ `\/( U, x )` (method)

Returns: the set of quotients u/x, $u \in U$.

If the result is not a subset of the underlying ring, the method gives up.

Example

```
gap> D/2;
Union of the residue classes 1(3) and 2(3) of Z
gap> M/5;
Union of the residue classes 1(6) and 5(6) of Z
```

3.3 The categories and families of unions of residue classes

3.3.1 IsUnionOfResidueClasses

◊ `IsUnionOfResidueClasses( U )` (filter)

The category of all unions of residue classes.

3.3.2 IsUnionOfResidueClassesOfZ

◊ `IsUnionOfResidueClassesOfZ( U )` (filter)

The category of all unions of residue classes of $\mathbb{Z}$.

3.3.3 IsUnionOfResidueClassesOfZ_pi

◊ `IsUnionOfResidueClassesOfZ_pi( U )` (filter)

The category of all unions of residue classes of some semilocalization $\mathbb{Z}_{(\pi)}$ of the integers.

3.3.4 IsUnionOfResidueClassesOfZorZ_pi

◊ `IsUnionOfResidueClassesOfZorZ_pi( U )` (filter)

The category of all unions of residue classes of $\mathbb{Z}$ or $\mathbb{Z}_{(\pi)}$.

3.3.5 IsUnionOfResidueClassesOfGFqx

◊ `IsUnionOfResidueClassesOfGFqx( U )` (filter)

The category of all unions of residue classes of some polynomial ring GF(*q*)[*x*].

3.3.6 ResidueClassUnionsFamily

◊ `ResidueClassUnionsFamily( R )` (function)

The family of unions of residue classes of the ring `R`.
The ring `R` can be accessed as `UnderlyingRing( ResidueClassUnionsFamily( R ) )`.

Chapter 4

Installation and auxiliary functions

4.1 Installation

Like any other GAP package, ResClasses must be installed in the `pkg` subdirectory of the GAP distribution. This is done by extracting the distribution file in this directory. By default, the package ResClasses is autoloaded, otherwise you can load it via `LoadPackage( "resclasses" );`. The ResClasses Package needs at least version 4.4 of GAP, is completely written in the GAP language and does neither contain nor require external binaries. For the documentation the package GAPDoc [LN02] is needed.

4.2 Building the manual

4.2.1 ResClassesBuildManual

◊ `ResClassesBuildManual( )` (function)

Returns: Nothing.

This function builds the manual of the ResClasses package in the file formats LaTeX, DVI, Postscript, PDF, HTML and ASCII text. This is done using the GAPDoc package by Frank Lübeck and Max Neunhöffer.

4.3 The testing routine

4.3.1 ResClassesTest

◊ `ResClassesTest( )` (function)

Returns: Nothing.

Performs tests of the ResClasses package. This function makes use of an adaptation of the test file `tst/testall.g` of the GAP library to this package.

4.4 Changing the viewing format for residue class unions

4.4.1 ResidueClassUnionViewingFormat

◊ `ResidueClassUnionViewingFormat( format )` (function)

Returns: Nothing.

Switches between a longer and more descriptive (`format` = `"long"`) and a shorter and less bulky (`format` = `"short"`) viewing format for unions of residue classes.

The former is the default and should be used when not many residue classes have to be displayed or residue classes of different rings are used, but the latter is usually preferable if it is always clear which the base ring is and if the printed representation of many residue classes should fit on one screen.

Example

```
gap> ResidueClassUnionViewingFormat("short");
gap> ResidueClassUnion(Integers,12,[1,4,5,7,10,11]);
1(3) U 5(6)
gap> ResidueClassUnionViewingFormat("long");
gap> ResidueClassUnion(Integers,12,[1,4,5,7,10,11]);
Union of the residue classes 1(3) and 5(6) of Z
```

References

[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. 19

Index