

ResClasses

Computations with Residue Classes and their Set-Theoretic Unions

(Version 2.2.2)

March 28, 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
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 $\text{GF}(q)[x]$ for some prime power q as GAP domains.

It provides basic functionality for computing with these sets, like forming intersections, unions and differences. Any of this is also supported in the case that one of the operands is a finite set of elements. The package also implements the mentioned rings $\mathbb{Z}_{(\pi)}$ as GAP domains.

Copyright

© 2003 - 2006 by Stefan Kohl

This package is distributed under the GNU General Public License.

Contents

1	Preface	5
2	Semilocalizations of the Integers	6
2.1	Entering semilocalizations of the integers	6
2.1.1	Z_{π} (π)	6
2.1.2	IsZ_{π} (R)	6
2.1.3	NoninvertiblePrimes (R)	6
2.2	Methods for semilocalizations of the integers	7
3	Unions of Residue Classes	8
3.1	Entering residue classes and unions thereof	8
3.1.1	ResidueClass (R, m, r)	8
3.1.2	IsResidueClass (obj)	8
3.1.3	ResidueClassUnion (R, m, r)	9
3.1.4	AllResidueClassesModulo (R, m)	9
3.1.5	AllResidues (R, m)	10
3.2	Methods for unions of residue classes	10
3.2.1	Union ($U1, U2$)	11
3.2.2	Intersection ($U1, U2$)	11
3.2.3	Difference ($U1, U2$)	12
3.2.4	SplittedClass (cl, t)	12
3.2.5	AsUnionOfFewClasses (U)	13
3.2.6	$\backslash+$ (U, x)	13
3.2.7	$\backslash-$ (U, x)	13
3.2.8	$\backslash*$ (U, x)	14
3.2.9	$\backslash\vee$ (U, x)	14
3.2.10	Density (U)	14
3.3	The categories and families of unions of residue classes	15
3.3.1	IsUnionOfResidueClasses (U)	15
3.3.2	ResidueClassUnionsFamily (R)	15
4	Unions of Residue Classes with Fixed Representatives	16
4.1	About unions of residue classes with fixed representatives	16
4.2	Entering unions of residue classes with fixed representatives	16
4.2.1	ResidueClassWithFixedRepresentative (R, m, r)	16
4.2.2	ResidueClassUnionWithFixedRepresentatives ($R, classes$)	17

4.2.3	AllResidueClassesWithFixedRepresentativesModulo (R, m)	17
4.3	Methods for unions of residue classes with fixed representatives	17
4.3.1	Multiplicity (x, U)	17
4.3.2	IsOverlappingFree (U)	18
4.3.3	AsOrdinaryUnionOfResidueClasses (U)	18
4.3.4	\in (cl, U)	18
4.3.5	AsListOfClasses (U)	19
4.3.6	IsSubset (U1, U2)	19
4.3.7	Density (U)	19
4.3.8	Union (U1, U2)	19
4.3.9	Intersection (U1, U2)	20
4.3.10	Difference (U1, U2)	20
4.3.11	\+ (U, x)	20
4.3.12	\- (U, x)	21
4.3.13	* (U, x)	21
4.3.14	\/ (U, x)	21
4.4	The invariant Delta	22
4.4.1	RepresentativeStabilizingRefinement (U, k)	22
4.4.2	Delta (U)	22
4.4.3	Rho (U)	22
4.5	The categories of unions of residue classes with fixed rep's	23
5	Installation and auxiliary functions	24
5.1	Requirements	24
5.2	Installation	24
5.3	The testing routine	24
5.3.1	ResClassesTest	24
5.4	Changing the viewing format for residue class unions	24
5.4.1	ResidueClassUnionViewingFormat (format)	24
5.5	Building the manual	25
5.5.1	ResClassesBuildManual	25

Chapter 1

Preface

This package provides an 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 ring elements w.r.t. these operations. The functionality of this package is used in a group theoretical context by the RCWA package written by the same author.

The introduction of unions of residue classes with fixed representatives which are implemented in this package since version 2.0 follows a suggestion by Wolfgang Rump.

Chapter 2

Semilocalizations of the Integers

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

2.1 Entering semilocalizations of the integers

2.1.1 Z_pi (π)

◇ $Z_pi(\pi)$ (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 (R)

◇ $IsZ_pi(R)$ (property)

Returns: true if R is a ring $\mathbb{Z}_{(\pi)}$ for some set of primes π and false otherwise.

2.1.3 $NoninvertiblePrimes$ (R)

◇ $NoninvertiblePrimes(R)$ (attribute)

Returns: The set of noninvertible primes π in the semilocalization R of the integers.

2.2 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 non-invertible prime factors of its numerator.

Example

```
gap> 4/7 in R;
true
gap> 3/2 in R;
false
gap> U := Intersection(R,S,T);
Z_( 2, 3, 5, 7, 11 )
gap> IsSubset(R,U);
true
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 3

Unions of Residue Classes

3.1 Entering residue classes and unions thereof

3.1.1 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 $r \bmod m$ of the ring R , and in the two-argument form the residue class $r \bmod m$ 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.

Example

```
gap> A := ResidueClass(2,3);
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> R := PolynomialRing(GF(7),1);; 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 IsResidueClass (obj)

- ◇ IsResidueClass(obj) (property)

Returns: true if obj is a (single) residue class and false otherwise.

Rings are regarded as residue class 0 (mod 1).

Example

```
gap> IsResidueClass(A);
true
gap> IsResidueClass(Z_pi([2,5]));
true
```

3.1.3 ResidueClassUnion (R, m, r)

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

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

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

If the arguments included and excluded are given, they must be sets of elements of R .

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 )>
```

3.1.4 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.

Example

```
gap> AllResidueClassesModulo(3);
[ The residue class 0(3) of Z, The residue class 1(3) of Z,
  The residue class 2(3) of Z ]
gap> AllResidueClassesModulo(Z_pi(2),4);
[ The residue class 0(4) of Z_( 2 ), The residue class 1(4) of Z_( 2 ),
  The residue class 2(4) of Z_( 2 ), The residue class 3(4) of Z_( 2 ) ]
gap> AllResidueClassesModulo(R,x);
[ The residue class 0*Z(7) ( mod x ) of GF(7)[x],
  The residue class Z(7)^0 ( mod x ) of GF(7)[x],
  The residue class Z(7) ( mod x ) of GF(7)[x],
  The residue class Z(7)^2 ( mod x ) of GF(7)[x],
  The residue class -Z(7)^0 ( mod x ) of GF(7)[x],
  The residue class Z(7)^4 ( mod x ) of GF(7)[x],
  The residue class Z(7)^5 ( mod x ) of GF(7)[x] ]
```

A transversal for the set of residue classes (mod m) can be obtained by the following function:

3.1.5 AllResidues (R, m)

◇ AllResidues(R, m)

(function)

Returns: A sorted list of all residues modulo m in the ring R .

Example

```
gap> AllResidues(Integers,6);
[ 0 .. 5 ]
gap> AllResidues(Z_pi([3,5,7]),700);
[ 0 .. 174 ]
gap> x := Indeterminate(GF(2),1); SetName(x,"x");
gap> R := PolynomialRing(GF(2),1); e := One(R); z := Zero(R);
GF(2)[x]
gap> AllResidues(R,x^4+x^2);
[ 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, x^3,
  x^3+Z(2)^0, x^3+x, x^3+x+Z(2)^0, x^3+x^2, x^3+x^2+Z(2)^0, x^3+x^2+x,
  x^3+x^2+x+Z(2)^0 ]
```

For extracting the components of a residue class union as given as arguments in ResidueClassUnion (3.1.3), there are operations Modulus, Residues, IncludedElements and ExcludedElements.

3.2 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 to test whether some ring element lies in a given union of residue classes.

Example

```
gap> 20 in A;
true
gap> 1/3 in B;
true
gap> x in G;
false
gap> Perform( [ C, F, H ], function( U ) Print(U,"\n"); end );
ResidueClassUnion( PolynomialRing( GF(7), ["x"] ), x+Z(7)^0, [ Z(7) ] )
ResidueClassUnion( Integers, 5, [ 1, 2 ], [ 3, 8 ], [ -4, 1 ] )
ResidueClassUnion( Z_pi( [ 2, 3 ] ), 8, [ 3, 5 ] )
```

There is a method for IsSubset available for unions of residue classes. As described in detail in the sequel, there are methods for computing set-theoretic unions, intersections and differences of unions of residue classes:

3.2.1 Union (U1, U2)

◇ 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]);
Z \ The residue class 0(5) of Z
gap> K := Union(I,J);
Z \ Union of the residue classes 0(10) and 15(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);
Z \ The residue class 0(3) of Z
```

3.2.2 Intersection (U1, U2)

◇ 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);
Union of the residue classes 1(30), 7(30), 11(30), 13(30), 17(30), 19(30), 23(
30) and 29(30) of Z
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 ]
```

3.2.3 Difference (U1, U2)

◇ 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);
Union of the residue classes 2(10), 4(10), 6(10), 8(10), 3(30), 9(30), 21(
30) and 27(30) of Z
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 ]
```

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

3.2.4 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(2,3),5);
[ The residue class 2(15) of Z, The residue class 5(15) of Z,
  The residue class 8(15) of Z, The residue class 11(15) of Z,
  The residue class 14(15) of Z ]
gap> SplittedClass(ResidueClass(Z_pi([2,3]),3,2),2);
[ The residue class 2(6) of Z_( 2, 3 ), The residue class 5(6) of Z_( 2, 3 ) ]
gap> SplittedClass(ResidueClass(Z_pi([2,3]),3,2),5);
fail
```

Sometimes one wants to know 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 yields usually “reasonably short” partitions:

3.2.5 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(K);
[ The residue class 1(5) of Z, The residue class 2(5) of Z,
  The residue class 3(5) of Z, The residue class 4(5) of Z,
  The residue class 5(30) of Z, The residue class 25(30) of Z ]
```

One can add / subtract a constant to / from all elements of a union of residue classes, and one can multiply or divide the elements by a constant:

3.2.6 \+ (U, x)

◇ \+(U, x)

(method)

◇ \+(x, U)

(method)

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

Example

```
gap> Display(L+1);
Union of the residue classes 0(30), 2(30), 8(30), 12(30), 14(30), 18(30), 20(
30) and 24(30) of Z
gap> L+30 = L;
true
```

3.2.7 \- (U, x)

◇ \-(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]
```

3.2.8 $\setminus^*(U, x)$

◇ $\setminus^*(U, x)$

(method)

◇ $\setminus^*(x, U)$

(method)

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

Example

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

3.2.9 $\setminus/(U, x)$

◇ $\setminus/(U, x)$

(method)

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

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

Example

```
gap> D/2;
Z \ The residue class 0(3) of Z
gap> M/5;
Union of the residue classes 1(6) and 5(6) of Z
```

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 as the sum of the densities of the elements of a partition of U into finitely many residue classes:

3.2.10 Density (U)

◇ $\text{Density}(U)$

(operation)

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

Example

```
gap> Density(G); G;
3/7
<union of 3 residue classes (mod x) of GF(7)[x]> U [ 0*Z(7) ] \ [ Z(7)^0 ]
gap> ResidueClassUnion(Integers, 12, [3,5,9]);
Union of the residue classes 3(6) and 5(12) of Z
gap> List([last, 2*last], Density);
[ 1/4, 1/8 ]
```

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

3.3 The categories and families of unions of residue classes

3.3.1 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.

3.3.2 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 accessed as `UnderlyingRing(ResidueClassUnionsFamily(R))`. Unions of residue classes with fixed representatives are described in the next chapter.

Chapter 4

Unions of Residue Classes with Fixed Representatives

4.1 About unions of residue classes with fixed representatives

In this chapter, a different kind of unions of residue classes is introduced – namely the one of residue classes which are endowed with a distinguished (“fixed”) representative. These unions of residue classes behave different than the “ordinary” residue class unions which were described in the previous chapter:

- In most situations they behave like lists of single residue classes with fixed representatives rather than like sets of ring elements. There are exceptions from this behaviour, e.g. w.r.t. forming differences, in order to ensure “ δ -additivity” (\rightarrow Delta (4.4.2)).
- They can be viewed as *multisets* of ring elements – the residue classes in such a union are not necessarily disjoint, and not even necessarily distinct.

Throughout this chapter, the argument R denotes the ring whose residue classes are considered, and the arguments U , $U1$ and $U2$ denote unions of residue classes of R with fixed representatives.

Some of the functionality described in this chapter makes only sense if R is the ring of integers – in particular this holds for everything concerning the invariant δ .

4.2 Entering unions of residue classes with fixed representatives

4.2.1 ResidueClassWithFixedRepresentative (R , m , r)

◇ `ResidueClassWithFixedRepresentative(R, m, r)` (function)

◇ `ResidueClassWithFixedRepresentative(m, r)` (function)

Returns: The residue class $r \bmod m$ of the ring R , with fixed representative r .

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

Example

```
gap> c11 := ResidueClassWithFixedRepresentative(Integers,3,2);
[2/3]
gap> c12 := ResidueClassWithFixedRepresentative(Integers,2,1);
[1/2]
```

In all names of functions described in this chapter, `Representative` can be abbreviated by `Rep`. When entering a union of residue classes with fixed representative, for *any* residue class in the union a representative has to be specified:

4.2.2 ResidueClassUnionWithFixedRepresentatives (R, classes)

◇ `ResidueClassUnionWithFixedRepresentatives(R, classes)` (function)

◇ `ResidueClassUnionWithFixedRepresentatives(classes)` (function)

Returns: The union of the residue classes $\text{classes}[i][2] \bmod \text{classes}[i][1]$ of the ring R , with fixed representatives $\text{classes}[i][2]$.

The argument `classes` must be a list of pairs of elements of the ring R , those first elements (the moduli) have to be non-zero. If the argument R is omitted, it defaults to `Integers`.

Example

```
gap> U := ResidueClassUnionWithFixedRepresentatives(Integers, [[2,1],[7,4]]);
[1/2] U [4/7]
```

There is a method for the operation `Modulus`, which returns the lcm of the moduli of the residue classes forming such a union, and there is an operation `Classes` for extracting the list of classes which is passed as an argument to `ResidueClassUnionWithFixedRepresentatives`.

4.2.3 AllResidueClassesWithFixedRepresentativesModulo (R, m)

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

◇ `AllResidueClassesWithFixedRepresentativesModulo(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` (3.1.4).

Example

```
gap> AllResidueClassesWithFixedRepresentativesModulo(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] ]
```

4.3 Methods for unions of residue classes with fixed representatives

There are methods for `Print`, `String` and `Display` which are applicable to unions of residue classes with fixed representatives. Unions of residue classes are multisets, thus elements can be contained with different multiplicities:

4.3.1 Multiplicity (x, U)

◇ `Multiplicity(x, U)` (method)

Returns: The multiplicity of x in U regarded as a multiset of ring elements.

Example

```
gap> List([1,2,11],n->Multiplicity(n,U));
[ 1, 0, 2 ]
```

4.3.2 IsOverlappingFree (U)

◇ IsOverlappingFree(U)

(property)

Returns: true if the residue classes in U are pairwise disjoint and false otherwise.

We call a residue class union U with fixed representatives *overlapping free* if and only if it consists of pairwise disjoint residue classes.

Example

```
gap> IsOverlappingFree(c11);
true
gap> IsOverlappingFree(U);
false
```

4.3.3 AsOrdinaryUnionOfResidueClasses (U)

◇ AsOrdinaryUnionOfResidueClasses(U)

(method)

Returns: The set-theoretic union of the residue classes in U.

The returned object is an ordinary residue class union without fixed representatives as described in Chapter 3 which behaves like a subset of the underlying ring.

Example

```
gap> List([c11,c12,U],AsOrdinaryUnionOfResidueClasses);
[ The residue class 2(3) of Z, The residue class 1(2) of Z,
  Union of the residue classes 1(2) and 4(14) of Z ]
```

4.3.4 \in (c1, U)

◇ \in(c1, U)

(method)

Returns: true if the residue class c1 with a fixed representative is an element of U and false otherwise.

Example

```
gap> c11 in U;
false
gap> c12 in U;
true
```

4.3.5 AsListOfClasses (U)

◇ AsListOfClasses (U)

(method)

Returns: The sorted list of the residue classes in U.

Example

```
gap> AsListOfClasses(U);
[ [1/2], [4/7] ]
```

4.3.6 IsSubset (U1, U2)

◇ IsSubset (U1, U2)

(method)

Returns: true if U2 is a subset of U1 and false otherwise.

We say that U2 is a subset of U1 if the multiplicity of any residue class $[r/m]$ in U1 is greater than or equal to its multiplicity in U2.

Example

```
gap> IsSubset(U,cl1);
false
gap> IsSubset(U,cl2);
true
```

4.3.7 Density (U)

◇ Density (U)

(operation)

Returns: The natural density of U as a multiset (elements with multiplicity k count k -fold).

Example

```
gap> Density(U);
9/14
gap> 1/2+1/7;
9/14
```

4.3.8 Union (U1, U2)

◇ Union (U1, U2)

(method)

Returns: The union of U1 and U2.

It holds $\Delta(\text{Union}(U1, U2)) = \Delta(U1) + \Delta(U2)$. ($\rightarrow \Delta(4.4.2)$).

Example

```
gap> Union(U,cl1);
[1/2] U [2/3] U [4/7]
```

4.3.9 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> Intersection(c11,c12);
Empty union of residue classes of Z with fixed representatives
gap> Intersection(List([c11,c12],AsOrdinaryUnionOfResidueClasses));
The residue class 5(6) of Z
gap> Intersection(c12,U);
[1/2]
```

4.3.10 Difference (U1, U2)

◇ `Difference(U1, U2)` (method)

Returns: The difference of U1 and U2.

It holds $\text{Delta}(\text{Difference}(U1,U2)) = \text{Delta}(U1) - \text{Delta}(U2)$. (\rightarrow Delta (4.4.2)). This is ensured by setting the difference of the empty residue class union with fixed representatives and some residue class $[r/m]$ equal to $[(m-r)/m]$.

Example

```
gap> Difference(U,c11);
[1/2] U [1/3] U [4/7]
gap> Difference(U,c12);
[4/7]
```

Throughout the rest of this section, U is regarded as a multiset of ring elements. For sake of simplicity, the term “the multiset of ring elements endowed with the structure of a union of residue classes with fixed representatives” is abbreviated by “the multiset”.

4.3.11 $\setminus + (U, x)$

◇ `\+(U, x)` (method)

◇ `\+(x, U)` (method)

Returns: The multiset of sums $u+x$, $u \in U$.

Example

```
gap> c11 + 1;
[3/3]
gap> U+23;
[24/2] U [27/7]
```

4.3.12 $\setminus - (\mathbf{U}, \mathbf{x})$

$\diamondsetminus - (\mathbf{U}, \mathbf{x})$ (method)

$\diamondsetminus - (\mathbf{x}, \mathbf{U})$ (method)

$\diamondsetminus - (\mathbf{U})$ (method)

Returns: The multiset 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> c12 - 1;
[0/2]
gap> U - 17;
[-16/2] U [-13/7]
```

4.3.13 $\setminus * (\mathbf{U}, \mathbf{x})$

$\diamondsetminus * (\mathbf{U}, \mathbf{x})$ (method)

$\diamondsetminus * (\mathbf{x}, \mathbf{U})$ (method)

Returns: The multiset of products $x \cdot u$, $u \in U$.

Scalar multiplication leaves δ invariant (\rightarrow Delta (4.4.2)).

Example

```
gap> 3*c11;
[6/9]
gap> 7*U;
[7/14] U [28/49]
```

4.3.14 $\setminus / (\mathbf{U}, \mathbf{x})$

$\diamondsetminus / (\mathbf{U}, \mathbf{x})$ (method)

Returns: The multiset of quotients u/x , $u \in U$.

Scalar division leaves δ invariant (\rightarrow Delta (4.4.2)). If not all elements of all residue classes in U are divisible by x , the method gives up.

Example

```
gap> (2*c11+2)/3;
[2/2]
```

4.4 The invariant Delta

4.4.1 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 $[r/km] \cup [(r+m)/km] \cup \dots \cup [(r+(k-1)m)/km]$. This definition is extended in a natural way to unions of residue classes.

The method tries to perform a simplification of U by joining appropriate residue classes if the argument k is zero.

In any case the value of $\Delta(U)$ is invariant under this operation ($\rightarrow \Delta$ (4.4.2)).

Example

```
gap> cl := ResidueClassUnionWithFixedReps(Integers, [[2,1]]);
[1/2]
gap> S := RepresentativeStabilizingRefinement(cl,3);
[1/6] U [3/6] U [5/6]
gap> cls := AsListOfClasses(S);
[ [1/6], [3/6], [5/6] ]
gap> cls := List([1..3], i->RepresentativeStabilizingRefinement(cls[i],i+1));
[ [1/12] U [7/12], [3/18] U [9/18] U [15/18],
  [5/24] U [11/24] U [17/24] U [23/24] ]
gap> S := Union(cls);
<union of 9 residue classes of Z with fixed representatives>
gap> RepresentativeStabilizingRefinement(S,0);
[1/2]
```

4.4.2 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 additively to unions of such residue classes. If no representatives are fixed, this definition is still unique (mod 1).

Example

```
gap> [ Delta(U), (1/2-1/2)+(4/7-1/2) ];
[ 1/14, 1/14 ]
gap> V := RepresentativeStabilizingRefinement(U,3);
[1/6] U [3/6] U [5/6] U [4/21] U [11/21] U [18/21]
gap> Delta(V) = (1/6-1/2)+(3/6-1/2)+(5/6-1/2)+(4/21-1/2)+(11/21-1/2)+(18/21-1/2);
true
```

4.4.3 Rho (U)

◇ Rho (U) (attribute)

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

For a union $U \subseteq \mathbb{Z}$ of finitely many residue classes, we set $\rho(U) := e^{\pi i \delta(U)}$.

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

The names of the categories of unions of residue classes with fixed representatives can be derived from the names of those of the “ordinary” unions of residue classes given in Section 3.3 by appending `WithFixedRepresentatives`.

Chapter 5

Installation and auxiliary functions

5.1 Requirements

The ResClasses package needs at least GAP 4.4.6 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.

5.2 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");`.

5.3 The testing routine

5.3.1 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`.

5.4 Changing the viewing format for residue class unions

5.4.1 ResidueClassUnionViewingFormat(format)

◇ `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
```

5.5 Building the manual

The following routine is a development tool. As all files it generates are included in the distribution file anyway, users will not need it.

5.5.1 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 accomplished using the GAPDoc package by Frank Lübeck and Max Neunhöffer. Building the manual is possible only on UNIX systems and requires \LaTeX , PDF \LaTeX and dvips.

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>. 24

Index

\backslash^*
 U, x, 14, 21
 x, U, 14, 21
 \backslash^+
 U, x, 13, 20
 x, U, 13, 20
 \backslash^-
 U, 13, 21
 U, x, 13, 21
 x, U, 13, 21
 $\backslash/$
 U, x, 14, 21
 AllResidueClassesModulo
 m, 9
 R, m, 9
 AllResidueClassesWithFixedRepresentativesModulo
 m, 17
 R, m, 17
 AllResidues
 R, m, 10
 AsListOfClasses
 U, 19
 AsOrdinaryUnionOfResidueClasses
 U, 18
 AsUnionOfFewClasses
 U, 13
 Delta
 U, 22
 Density
 U, 14, 19
 Difference
 U, S, 12
 U1, U2, 12, 20
 $\backslash\text{in}$
 cl, U, 18
 Intersection
 U, S, 11
 U1, U2, 11, 20
 U, 18
 IsOverlappingFree
 U, 18
 IsResidueClass
 obj, 8
 IsSubset
 U1, U2, 19
 IsUnionOfResidueClasses
 U, 15
 IsUnionOfResidueClassesOfGFqx
 U, 15
 IsUnionOfResidueClassesOfZ
 U, 15
 IsUnionOfResidueClassesOfZ_pi
 U, 15
 IsZ_pi
 R, 6
 Multiplicity
 x, U, 17
 NoninvertiblePrimes
 R, 6
 RepresentativeStabilizingRefinement
 U, k, 22
 ResClassesBuildManual, 25
 ResClassesTest, 24
 ResidueClass
 m, r, 8
 r, m, 8
 R, m, r, 8
 ResidueClassUnion
 R, m, r, 9
 R, m, r, included, excluded, 9
 ResidueClassUnionsFamily
 R, 15
 R, fixedreps, 15
 ResidueClassUnionViewingFormat
 format, 24

ResidueClassUnionWithFixedRepresentatives
 classes, 17
 R, classes, 17

ResidueClassWithFixedRepresentative
 m, r, 16
 R, m, r, 16

Rho
 U, 22

SplittedClass
 cl, t, 12

Union
 U, S, 11
 U1, U2, 11, 19

Z_pi
 pi, 6