RCWA

Residue Class-Wise Affine Representations of Groups

(Version 1.0)

April 24, 2002

Stefan Kohl

Stefan Kohl — Email: kohl@mathematik.uni-stuttgart.de
— Homepage: http://www.cip.mathematik.uni-stuttgart.de/~kohlsn

Abstract

This package for GAP 4 (at least version 4.3) provides routines for computations with *R*esidue *C*lass-*W*ise *A*ffine mappings of $\mathbb{Z}$, its semilocalizations $\mathbb{Z}_\pi$ and the polynomial rings GF(q)[x], hence it can be used for computations in certain types of infinite permutation groups. It is completely written in the GAP language and contains / requires no external binaries. For the documentation, the package GAPDoc [3] is needed. The package GRAPE[5] is loaded for use by the function `RcwaGraph`, if present. RCWA must be installed in the `pkg` subdirectory of the GAP distribution.

Copyright

Contents

Chapter 1

Preface

This package deals with a certain kind of infinite permutations, namely the “residue class-wise affine” ones. In particular, it provides functionality for doing computations with such mappings over the ring of integers, its semilocalizations and over polynomial rings in one variable over a finite field. In the introduction, I will give brief definitions.

It seems that there is just no literature concerning this topic, but nevertheless I would like to mention the article of Hans Läuchli and Peter M. Neumann [4], which is a little bit related to the subject (but does not deal with any kind of permutation groups over rings or of countable permutation groups).

I got the idea of introducing this kind of permutations resp. representations of groups originally from my investigations concerning the Collatz- ($3n+1$-) conjecture (for an annotated bibliography, see [2]), but now, residue class-wise affine representations of groups seem to be an interesting topic of its own.

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

Chapter 2

Introduction

2.1 Definitions

In this manual, we only give short definitions of the most important terms in this context.

Let R be a commutative principal ideal domain without zero-divisors, such that for all non-zero ideals I of R, the quotient $|R/I|$ is finite.

We call a mapping f from R to itself *residue class-wise affine* if there is a non-zero ideal I_f of R such that f is given on each residue class $r + I_f \in R/I_f$ by

$$n \mapsto \frac{a_r \cdot n + b_r}{c_r}$$

for some coefficients $a_r, b_r, c_r \in R$. In this case, we say that f is an *rcwa mapping*.

We always assume that all fractions are reduced, i.e. that $\gcd(a_r, b_r, c_r) = 1$, and that I_f is the largest ideal having the described property.

We define the *modulus* $\mathrm{Mod}(f)$ of f as the (up to multiplication by units uniquely determined) element m_f generating the ideal I_f.

We define the *multiplier* $\mathrm{Mult}(f)$ of f as the standard associate of the least common multiple of the coefficients a_r in the numerators.

We define the *divisor* $\mathrm{Div}(f)$ of f as the standard associate of the least common multiple of the coefficients c_r in the denominators.

We say that an rcwa mapping is *flat* in case its multiplier and divisor are both equal to 1.

In case that the underlying ring R is the ring of integers, we call an rcwa mapping f an *integral rcwa mapping*, in case $R = \mathbb{Z}_\pi$ for a set of primes π we call f a *semilocal integral rcwa mapping* and in case R is a polynomial ring in one variable over a finite field we call f a *modular rcwa mapping*. Since integral and semilocal integral rcwa mappings share many important properties, we make use of the generic term *rational-based rcwa mapping*.

We call a rational-based rcwa mapping *class-wise order-preserving* if its restriction to any residue class modulo its modulus is order-preserving.

We define the *graph* Γ_f associated to an rcwa mapping f with modulus m as follows:

1. The vertices are the residue classes (mod m).
2. There is an edge from $r_1(m)$ to $r_2(m)$ if and only if there is some $n_1 \in r_1(m)$ such that $n_1^f \in r_2(m)$.

We define the *transition matrix* M of degree d of the integral rcwa mapping f by $M_{i+1,j+1} = 1$ if there is an $n \equiv i \pmod d$ such that $n^f \equiv j \pmod d$, and 0 if not. Their rank (and in case it is invertible the

absolute value of its determinant) does not depend on the particular assignment of the residue classes (mod d) to rows/columns, hence accordingly, we can define the *transitional rank* resp. the *transitional determinant* of f of degree d for any rcwa mapping.

We define the *prime set* of an rcwa mapping f as the set of all prime elements dividing the modulus of f or some coefficient a_r or c_r (in the notation used above).

We set $\mathrm{RCWA}(R) := \{ g \in \mathrm{Sym}(R) \mid g \text{ is residue class-wise affine} \}$.

The set RCWA(R) is closed under multiplication and taking inverses (this can be verified easily), hence forms a subgroup of Sym(R). Since R contains no zero-divisors and the quotients R/I are all finite, this subgroup is proper.

We call a subgroup of RCWA(R) a residue class-wise affine group, or shortly, an *rcwa group*.

We call an rcwa group *flat* if all of its elements are.

We call an integral rcwa group *class-wise order-preserving* if all of its elements are.

We define the *prime set* of an rcwa group as the union of the prime sets of its elements.

We define the *modulus* of an rcwa group G as the least common multiple of the moduli of its elements in case this is finite, and zero otherwise.

We say that an R-rcwa mapping f is *tame* if and only if the moduli of its powers are bounded, and *wild* otherwise. Furthermore, we say that an R-rcwa group is tame if and only if its modulus is strictly positive, and wild otherwise.

We define an (R-) *rcwa representation* of a group G as an homomorphism from G to RCWA(R).

For an introduction to this topic, see [1].

Chapter 3

Semilocalizations of the integers

Since in the following we need the semilocalizations $\mathbb{Z}_\pi$ of the ring of integers (which are not already implemented as domains in the GAP library) as sources and ranges of rcwa mappings, we have to do this here.

3.1 Computing with semilocalizations of the integers

3.1.1 Z_pi

◊ `Z_pi( pi )` (function)

The ring $\mathbb{Z}_\pi$.

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 ]
```

3.1.2 IsZ_pi

◊ `IsZ_pi( R )` (property)

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

3.1.3 NoninvertiblePrimes

◊ `NoninvertiblePrimes( R )` (attribute)

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

3.1.4 \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
```

3.1.5 Intersection

◊ Intersection(R, S) (method)

The intersection of the semilocalizations R and S of the integers.

Example

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

3.1.6 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
```

3.1.7 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 $\mathbb{Z}_\pi$ as the product of the noninvertible 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
```

3.1.8 GcdOp

◊ GcdOp(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
```

3.1.9 LcmOp

◊ LcmOp(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
```

3.1.10 Factors

◊ Factors(R, x) (method)

Computes a prime factorization of x in the semilocalization R of the integers.

This method returns a list l, where the noninvertible prime factors of the numerator of x are stored in l{[2..Length(l)]} in ascending order, and l[1] is a suitable unit.

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 ]
```

3.1.11 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 4

Residue Class-Wise Affine Mappings

This chapter describes the functionality available for calculating with rcwa mappings.

4.1 The categories and families of rcwa mappings

4.1.1 IsRcwaMapping

◊ `IsRcwaMapping( f )` (filter)

The category of all rcwa mappings.

4.1.2 IsRationalBasedRcwaMapping

◊ `IsRationalBasedRcwaMapping( f )` (filter)

The category of all “rational-based” (hence all integral and all semilocal integral) rcwa mappings.

4.1.3 IsIntegralRcwaMapping

◊ `IsIntegralRcwaMapping( f )` (filter)

The category of all integral rcwa mappings.

4.1.4 IntegralRcwaMappingsFamily

◊ `IntegralRcwaMappingsFamily` (family)

The family of all integral rcwa mappings.

4.1.5 IsSemilocalIntegralRcwaMapping

◊ `IsSemilocalIntegralRcwaMapping( f )` (filter)

The category of all semilocal integral rcwa mappings.

4.1.6 SemilocalIntegralRcwaMappingsFamily

◊ SemilocalIntegralRcwaMappingsFamily(primes) (function)

The family of rcwa mappings over $\mathbb{Z}_\pi$, where π is the set of primes given as argument primes.

4.1.7 IsModularRcwaMapping

◊ IsModularRcwaMapping(f) (filter)

The category of all modular rcwa mappings.

4.1.8 ModularRcwaMappingsFamily

◊ ModularRcwaMappingsFamily(q) (function)

The family of rcwa mappings over the ring GF(q)[x].

4.2 Constructing rcwa mappings

4.2.1 IntegralRcwaMapping

◊ IntegralRcwaMapping(coeffs) (function)
◊ IntegralRcwaMapping(perm, range) (function)
◊ IntegralRcwaMapping(modulus, val) (function)
◊ RcwaMapping(coeffs) (function)
◊ RcwaMapping(perm, range) (function)
◊ RcwaMapping(modulus, val) (function)

Construction of the integral rcwa mapping

(a) with coefficients coeffs resp.

(b) acting on the translates of range by integral multiples of the length of range as the translates of the action of the finite permutation perm on range to the respective intervals, where moved points of perm outside range are ignored (range must entirely consist out of positive integers less than 2^{28}, since GAP permutations can only move these), resp.

(c) with modulus modulus and with values prescribed by the list val, which consists of 2·modulus pairs giving preimage and image for 2 points per residue class (mod modulus).

In case (a), coeffs is a list of m lists of 3 integers each (where m is the modulus of the mapping), giving the coefficients a_r, b_r and c_r for $r = 0, \ldots, m-1$ as described in the introduction.

Example

```
gap> f := RcwaMapping([[1,1,1],[1,-1,1],[1,1,1],[1,-1,1]]);
<integral rcwa mapping with modulus 2>
gap> f = RcwaMapping((2,3),[2..3]);
true
gap> g := RcwaMapping((1,2,3)(8,9),[4..20]);
```

```
<integral rcwa mapping with modulus 17>
gap> Action(Group(g),[4..20]);
Group([ ( 5, 6) ])
gap> T := RcwaMapping([[1,0,2],[3,1,2]]); # The Collatz mapping.
<integral rcwa mapping with modulus 2>
gap> T := RcwaMapping(2,[[1,2],[2,1],[3,5],[4,2]]); # The same, by mod. and values.
<integral rcwa mapping with modulus 2>
gap> t := RcwaMapping(1,[[-1,1],[1,-1]]); # The involution n -> -n.
Integral rcwa mapping: n -> -n
```

4.2.2 SemilocalIntegralRcwaMapping

◊ SemilocalIntegralRcwaMapping(pi, coeffs) (function)
◊ RcwaMapping(pi, coeffs) (function)

Construction of the rcwa mapping of $\mathbb{Z}_\pi$ with coefficients `coeffs`, where the set of primes π is given as argument `primes`.

Example

```
gap> d := RcwaMapping([2],[[1/3,0,1]]);
Rcwa mapping of Z_[ 2 ]: n -> 1/3 n
gap> RcwaMapping([2,3],ShallowCopy(Coefficients(T)));
<rcwa mapping of Z_[ 2, 3 ] with modulus 2>
```

4.2.3 ModularRcwaMapping

◊ ModularRcwaMapping(q, modulus, coeffs) (function)
◊ RcwaMapping(q, modulus, coeffs) (function)

Construction of the rcwa mapping of GF(`q`)[`x`] with modulus `modulus` and coefficients `coeffs`.

Example

```
gap> R := PolynomialRing(GF(2),1);;
gap> x := IndeterminatesOfPolynomialRing(R)[1];; SetName(x,"x");;
gap> e := One(GF(2));; z := Zero(R);;
gap> r := ModularRcwaMapping( 2, x^2 + e,
>                             [ [ x^2 + x + e, z      , x^2 + e ],
>                               [ x^2 + x + e, x      , x^2 + e ],
>                               [ x^2 + x + e, x^2    , x^2 + e ],
>                               [ x^2 + x + e, x^2 + x, x^2 + e ] ] );
<rcwa mapping of GF(2)[x] with modulus Z(2)^0+x^2>
```

There is an auxiliary function for determining the correct order of the coefficient triples of a modular rcwa mapping:

4.2.4 AllGFqPolynomialsModDegree

◇ AllGFqPolynomialsModDegree(q, d, x) (function)

Returns a sorted list of all residues modulo a polynomial of degree d over GF(q) in the variable x. This gives also the ordering in which the coefficients of a modular rcwa mapping are stored; thus, if f is a modular rcwa mapping over GF(q)[x] with coefficients list c, whose modulus m has degree d, then f maps a polynomial P with P mod m = r to

(c[Position(res, r)][1] * P + c[Position(res, r)][2]) / c[Position(res, r)][3]

where res denotes the list of residues returned by this function.

Example

```
gap> AllGFqPolynomialsModDegree(2,4,x);
[ 0*Z(2), Z(2)^0, x, Z(2)^0+x, x^2, Z(2)^0+x^2, x+x^2, Z(2)^0+x+x^2, x^3,
  Z(2)^0+x^3, x+x^3, Z(2)^0+x+x^3, x^2+x^3, Z(2)^0+x^2+x^3, x+x^2+x^3,
  Z(2)^0+x+x^2+x^3 ]
```

The internal representation of an rcwa mapping of any kind is always converted to a normalized (reduced) form, i.e. for all r, the coefficients a_r, b_r and c_r are divided by their gcd and then multiplied by a suitable unit such that c_r equals its standard conjugate, and the mapping is reduced to the smallest possible modulus.

4.3 Extracting the components of rcwa mappings

4.3.1 Coefficients

◇ Coefficients(f) (method)

The coefficients of the rcwa mapping f.

Example

```
gap> Coefficients(T);
[ [ 1, 0, 2 ], [ 3, 1, 2 ] ]
gap> Coefficients(r);
[ [ Z(2)^0+x+x^2, 0*Z(2), Z(2)^0+x^2 ], [ Z(2)^0+x+x^2, x, Z(2)^0+x^2 ],
  [ Z(2)^0+x+x^2, x^2, Z(2)^0+x^2 ], [ Z(2)^0+x+x^2, x+x^2, Z(2)^0+x^2 ] ]
```

4.3.2 Modulus (Modulus of an rcwa mapping)

◇ Modulus(f) (operation)

The modulus of the rcwa mapping f. See also Modulus (5.3.6) for rcwa groups, and IsTame (4.10.2).

Example

```
gap> u := RcwaMapping([[3,0,5],[9,1,5],[3,-1,5],[9,-2,5],[9,4,5]]);;
gap> Modulus(u);
5
gap> Modulus(r);
Z(2)^0+x^2
```

4.4 Flat and order-preserving mappings, multiplier and divisor

4.4.1 Multiplier

◊ Multiplier(f) (attribute)

The multiplier of the rcwa mapping f.

Example

```
gap> Multiplier(g);
1
gap> Multiplier(u);
9
gap> Multiplier(T);
3
gap> Multiplier(d);
1
gap> Multiplier(r);
Z(2)^0+x+x^2
```

4.4.2 Divisor

◊ Divisor(f) (attribute)

The divisor of the rcwa mapping f.

Example

```
gap> Divisor(g);
1
gap> Divisor(u);
5
gap> Divisor(T);
2
gap> Divisor(d);
1
gap> Divisor(r);
Z(2)^0+x^2
```

4.4.3 IsFlat (Flat rcwa mapping)

◊ IsFlat(f) (property)

Indicates whether the rcwa mapping f is flat or not. See also IsFlat (5.3.8) for rcwa groups.

Example

```
gap> IsFlat(g);
true
gap> IsFlat(u);
false
gap> IsFlat(T);
false
gap> IsFlat(d);
true
```

```
gap> IsFlat(r);
false
```

4.4.4 IsClassWiseOrderPreserving (Class-wise order-preserving rcwa mapping)

◊ IsClassWiseOrderPreserving(f) (property)

Indicates whether the rational-based rcwa mapping f is class-wise order-preserving or not. See also IsClassWiseOrderPreserving (5.3.9) for rcwa groups.

Example

```
gap> IsClassWiseOrderPreserving(g);
true
gap> IsClassWiseOrderPreserving(u);
true
gap> IsClassWiseOrderPreserving(T);
true
gap> IsClassWiseOrderPreserving(t);
false
gap> IsClassWiseOrderPreserving(d);
true
```

4.5 Checking for equality

4.5.1 \=

◊ \=(f, g) (method)

Tests whether the rcwa mappings f and g are equal. Since rcwa mappings are stored in a normalized form, this requires only comparing their coefficients.

Example

```
gap> RcwaMapping([[1,1,1],[2,-2,2],[3,3,3],[4,-4,4]])
>  = RcwaMapping([[1,1,1],[1,-1,1]]);
true
```

4.6 Printing and displaying rcwa mappings

4.6.1 Print

◊ Print(f) (method)

Prints the rcwa mapping f in GAP-readable format.

Example

```
gap> Print(T,"\n");
IntegralRcwaMapping( [ [ 1, 0, 2 ], [ 3, 1, 2 ] ] )
gap> Print(d,"\n");
SemilocalIntegralRcwaMapping( [ 2 ], [ [ 1/3, 0, 1 ] ] )
gap> Print(r,"\n");
ModularRcwaMapping( 2, Z(2)^0+x^2, [ [ Z(2)^0+x+x^2, 0*Z(2), Z(2)^0+x^2 ],
```

```
    [ Z(2)^0+x+x^2, x, Z(2)^0+x^2 ], [ Z(2)^0+x+x^2, x^2, Z(2)^0+x^2 ],
    [ Z(2)^0+x+x^2, x+x^2, Z(2)^0+x^2 ] ] )
```

The string printed by this method may be obtained with

4.6.2 String

◇ `String( f )` (method)

Converts the rcwa mapping `f` to a string. The string is the same as the one which is produced by `Print`.

Example

```
gap> String(d);
"SemilocalIntegralRcwaMapping( [ 2 ], [ [ 1/3, 0, 1 ] ] )"
```

4.6.3 Display

◇ `Display( f )` (method)

Displays the rcwa mapping `f` in a nice human-readable form.

Example

```
gap> SetName(u,"u");
gap> Display(u);

Integral rcwa mapping with modulus 5

                n mod 5                |                  n^u
---------------------------------------+---------------------------------------
  0                                    | 3n/5
  1                                    | (9n + 1)/5
  2                                    | (3n - 1)/5
  3                                    | (9n - 2)/5
  4                                    | (9n + 4)/5

gap> SetName(r,"r");
gap> Display(r);

Rcwa mapping of GF(2)[x] with modulus Z(2)^0+x^2

      P mod Z(2)+x^2       |                        P^r
---------------------------+-------------------------------------------------
 0*Z(2)                    | (Z(2)+x+x^2)*P/(Z(2)+x^2)
 Z(2)                      | ((Z(2)+x+x^2)*P + x)/(Z(2)+x^2)
 x                         | ((Z(2)+x+x^2)*P + x^2)/(Z(2)+x^2)
 Z(2)+x                    | ((Z(2)+x+x^2)*P + x+x^2)/(Z(2)+x^2)
```

4.7 Images and preimages under rcwa mappings

4.7.1 ImageElm

◊ ImageElm(f, n) (method)
◊ \^(f, n) (method)

Computes the image of the integer n under the rcwa mapping f.

Example

```
gap> 15^T;
23
gap> 7^d;
7/3
gap> p := (x^3+x^2+x+One(R))^r;
Z(2)^0+x^3
```

4.7.2 PreImageElm

◊ PreImageElm(f, n) (method)

Computes the preimage of the integer n under the bijective rcwa mapping f.

Example

```
gap> PreImageElm(u,8);
4
gap> PreImageElm(d,37/17);
111/17
gap> PreImageElm(r,p);
Z(2)^0+x+x^2+x^3
```

If the mapping f is not bijective, we can use

4.7.3 PreImagesElm

◊ PreImagesElm(f, n) (method)

Computes the set of preimages of the integer n under the rcwa mapping f.

If the preimage contains a full residue class but is not equal to the full source of the mapping, it is not representable as a set in GAP and the method will give up. In this case, we can get at least a representative of the preimage by PreImagesRepresentative (4.7.4).

Example

```
gap> PreImagesElm(T,8);
[ 16, 5 ]
gap> PreImagesElm(ZeroIntegralRcwaMapping,0);
Integers
```

4.7.4 PreImagesRepresentative

◊ PreImagesRepresentative(f, n) (method)

A representative of the set of preimages of the element n under the rcwa mapping f.

Example

```
gap> ZeroOne := RcwaMapping([[0,0,1],[0,1,1]]);;
gap> PreImagesRepresentative(ZeroOne,1);
1
```

4.8 Testing for injectivity, surjectivity and bijectivity

4.8.1 IsInjective

◊ IsInjective(f) (method)
◊ IsSurjective(f) (method)
◊ IsBijective(f) (method)

Tests whether the rcwa mapping f is injective, surjective resp. bijective.

Example

```
gap> IsInjective(T);
false
gap> IsSurjective(T);
true
gap> IsBijective(u);
true
gap> a_2 := RcwaMapping([2],[[3,0,2],[3,1,4],[3,0,2],[3,-1,4]]);
<rcwa mapping of Z_[ 2 ] with modulus 4>
gap> a_23 := RcwaMapping([2,3],[[3,0,2],[3,1,4],[3,0,2],[3,-1,4]]);
<rcwa mapping of Z_[ 2, 3 ] with modulus 4>
gap> IsInjective(a_2);
false
gap> IsSurjective(a_2);
true
gap> IsBijective(a_23);
true
gap> IsBijective(r);
true
```

4.9 Arithmetical operations and neutral elements

4.9.1 \+

◊ \+(f, g) (method)
◊ \-(f, g) (method)

Computes the (pointwise) sum resp. difference of the rcwa mappings f and g.

Example

```
gap> a := RcwaMapping([[3,0,2],[3, 1,4],[3,0,2],[3,-1,4]]);;
gap> b := RcwaMapping([[3,0,2],[3,13,4],[3,0,2],[3,-1,4]]);;
gap> Display(a + b);

Integral rcwa mapping with modulus 4

                n mod 4                |                 n^f
---------------------------------------+---------------------------------------
  0 2                                  | 3n
  1                                    | (3n + 7)/2
  3                                    | (3n - 1)/2

gap> Display(a - b);

Integral rcwa mapping with modulus 4

                n mod 4                |                 n^f
---------------------------------------+---------------------------------------
  0 2 3                                | 0
  1                                    | -3

gap> d+d+d;
IdentityMapping( Z_[ 2 ] )
gap> e := One(r);;
gap> e+e;
ZeroMapping( GF(2)[x], GF(2)[x] )
gap> p^(r+e) = p^r + p;
true
```

4.9.2 *

◇ *(f, g) (method)

Computes the product (composition) of the rcwa mappings `f` and `g`. The mapping `f` is applied first.

Example

```
gap> Display(a*b);

Integral rcwa mapping with modulus 16

               n mod 16                |                 n^f
---------------------------------------+---------------------------------------
   0  4  8 12                          | 9n/4
   1                                    | (9n + 55)/16
   2 10                                 | (9n - 2)/8
   3 11                                 | (9n - 3)/8
   5 13                                 | (9n + 3)/8
```

```
   6 14                                  | (9n + 26)/8
   7                                     | (9n + 49)/16
   9                                     | (9n - 1)/16
  15                                     | (9n - 7)/16

gap> s := RcwaMapping(2,x,[[One(R),Zero(R),One(R)],[x,Zero(R),One(R)]]);
<rcwa mapping of GF(2)[x] with modulus x>
gap> s^2=s;
true   # a non-trivial idempotent
gap> IsInjective(s) or IsSurjective(s);
false  # certainly not a group element ...
gap> Display(s*r-r*s);

Rcwa mapping of GF(2)[x] with modulus x+x^3

       P mod x+x^3        |                          P^f
--------------------------+-------------------------------------------------
 0*Z(2)     Z(2)          |
 x          x^2           |
 Z(2)+x^2   x+x^2         | 0*Z(2)
 Z(2)+x     Z(2)+x+x^2    | x
```

Multiplying rational-based rcwa mappings and (finite) permutations is forbidden, since GAP-permutations can only move positive integers less than 2^{28}; more precisely, one would encounter problems like the following:

Example

```
gap> (1,2)^RcwaMapping([[-1,0,1]]);
(-2,-1)
gap> (1,2)^RcwaMapping([[1,2^28,1]]);
(268435457,268435458)
```

4.9.3 Inverse

◊ Inverse(f) (method)

Computes the inverse mapping of the bijective rcwa mapping `f`.

Example

```
gap> Display(Inverse(u));

Bijective integral rcwa mapping with modulus 9

                n mod 9                 |                  n^f
----------------------------------------+----------------------------------------
  0 3 6                                 | 5n/3
  1 4 7                                 | (5n + 1)/3
  2                                     | (5n - 1)/9
  5                                     | (5n + 2)/9
  8                                     | (5n - 4)/9
```

```
gap> Display(Inverse(r));

Bijective rcwa mapping of GF(2)[x] with modulus Z(2)^0+x+x^2

     P mod Z(2)+x+x^2     |                          P^f
--------------------------+-------------------------------------------------------
 0*Z(2)                   | (Z(2)+x^2)*P/(Z(2)+x+x^2)
 Z(2)                     | ((Z(2)+x^2)*P + x)/(Z(2)+x+x^2)
 x                        | ((Z(2)+x^2)*P + x^2)/(Z(2)+x+x^2)
 Z(2)+x                   | ((Z(2)+x^2)*P + x+x^2)/(Z(2)+x+x^2)
```

4.9.4 \^

◊ \^(f, g) (method)

Computes the conjugate $f^g = g^{-1}fg$ of f under g.

Example

```
gap> T^u;
<surjective integral rcwa mapping with modulus 18>
gap> Display(s^r);

Rcwa mapping of GF(2)[x] with modulus x+x^2+x^3

     P mod x+x^2+x^3      |                          P^f
--------------------------+-------------------------------------------------------
 0*Z(2)      x            |
 x^2         x+x^2        | P
 Z(2)        Z(2)+x+x^2   | (x)*P
 Z(2)+x      Z(2)+x^2     | (x)*P + x
```

4.9.5 ZeroIntegralRcwaMapping

◊ ZeroIntegralRcwaMapping (global variable)
◊ Zero(f) (method)

The zero integral rcwa mapping, resp. the zero mapping in the family of rcwa mappings f belongs to.

Example

```
gap> Zero(a);
ZeroMapping( Integers, Integers )
gap> Zero(r);
ZeroMapping( GF(2)[x], GF(2)[x] )
```

4.9.6 IdentityIntegralRcwaMapping

◊ IdentityIntegralRcwaMapping (global variable)
◊ One(f) (method)

The identity integral rcwa mapping, resp. the identity mapping in the family of rcwa mappings f belongs to.

Example

```
gap> One(a);
IdentityMapping( Integers )
gap> One(d);
IdentityMapping( Z_[ 2 ] )
```

Sometimes this is also printed as IdentityMapping(R), where R denotes the ring f is acting on.

Caution: The set of rcwa mappings over a ring does not form a ring under addition and multiplication – it holds only the left distibutive law ($a \cdot (b+c) = a \cdot b + a \cdot c$, but not necessary $(a+b) \cdot c = a \cdot c + b \cdot c$), and the zero mapping multiplicatively is only a right zero element ($a \cdot 0 = 0$ for all a, but $0 \cdot a = 0$ if and only if $0^a = 0$).

4.10 Computing the order of an rcwa mapping

4.10.1 Order

◊ Order(f) (method)

Computes the multiplicative order of the bijective rcwa mapping f.

One of the two sufficient criteria for f having infinite order which are used by this package relies on a currently unproved hypothesis.

Example

```
gap> Order(Comm(a,b));
6
gap> Order(u);
infinity
```

The other criterium applies in case that f is tame:

4.10.2 IsTame (Tame rcwa mapping)

◊ IsTame(f) (property)

Determines whether or not the rcwa mapping f is tame.

See also IsTame (5.3.10) for rcwa groups.

Example

```
gap> IsTame(T);
false
gap> IsTame(a) or IsTame(b);
```

```
false
gap> IsTame(Comm(a,b));
true
```

4.11 Graph, transition matrix and prime set

4.11.1 RcwaGraph

◊ RcwaGraph(f) (operation)

Computes the graph associated to the rcwa mapping f.

For technical reasons, the residue classes 0(m) .. m-1(m) modulo the modulus m of f are identified with vertices named 1 .. m, in this order.

The result is returned as a GRAPE-graph, hence the package GRAPE has to be present; otherwise, fail is returned after issueing a warning.

Example

```
gap> RcwaGraph(a);
rec( isGraph := true, order := 4, group := Group(()),
  schreierVector := [ -1, -2, -3, -4 ],
  adjacencies := [ [ 1, 3 ], [ 1, 2, 3, 4 ], [ 2, 4 ], [ 1, 2, 3, 4 ] ],
  representatives := [ 1, 2, 3, 4 ], names := [ 1, 2, 3, 4 ] )
```

4.11.2 TransitionMatrix

◊ TransitionMatrix(f, deg) (function)

Computes the transition matrix of degree deg of the rcwa mapping f.

Example

```
gap> M := TransitionMatrix(a,7);;
gap> Display(M);
[ [  1,  0,  1,  0,  0,  1,  0 ],
  [  0,  1,  0,  0,  1,  1,  0 ],
  [  1,  0,  0,  1,  0,  0,  0 ],
  [  0,  1,  1,  0,  0,  0,  1 ],
  [  0,  1,  0,  0,  0,  1,  1 ],
  [  1,  0,  0,  0,  1,  0,  0 ],
  [  0,  0,  1,  1,  0,  0,  1 ] ]
gap> DeterminantMat(M);
-5
gap> M := TransitionMatrix(T,13);;
gap> Display(M*One(GF(2)));
 1 . . . . . 1 . . . . .
 . . 1 . . . . 1 . . . .
 . 1 . . . . . . . 1 . .
 . . . . . 1 . . 1 . . . .
 1 . 1 . . . . . . . . .
 . . . . . . . . 1 1 . . .
 . . . 1 . . . . . . . .
 . . . . . . . . . 1 1 .
```

```
 . . . . 1 . 1 . . . . . .
 . 1 . . . . . . . . . 1 .
 . . . . . 1 . . . 1 . . .
 . . . . 1 . . . . . . . 1
 . . . . . . 1 . . . . . 1
gap> DeterminantMat(M);
-16
```

4.11.3 PrimeSet (Prime set of an rcwa mapping)

◊ PrimeSet(f) (operation)

Computes the prime set of the rcwa mapping f.
See also PrimeSet (5.3.7) for rcwa groups.

Example

```
gap> PrimeSet(T);
[ 2, 3 ]
gap> PrimeSet(u);
[ 3, 5 ]
gap> PrimeSet(T^u);
[ 2, 3 ]
gap> PrimeSet(T^(u^-1));
[ 2, 3, 5 ]
gap> PrimeSet(r);
[ Z(2)^0+x, Z(2)^0+x+x^2 ]
```

4.12 The normal form of an rcwa mapping

4.12.1 ShortCycles

◊ ShortCycles(f, maxlng) (operation)

Computes all "single" finite cycles, hence all finite cycles not belonging to an infinite series, of the rcwa mapping f of length at most maxlng. Since in GAP, permutations cannot move negative integers, rationals or even polynomials, the cycles are returned as lists, for example, the list [-3,1,2,-2] denotes the cycle (-3,1,2,-2).

Example

```
gap> ShortCycles(a,2);
[ [ 0 ], [ 1 ], [ -1 ], [ 2, 3 ], [ -3, -2 ] ]
gap> ShortCycles(a,5);
[ [ 0 ], [ 1 ], [ -1 ], [ 2, 3 ], [ -3, -2 ], [ 4, 6, 9, 7, 5 ],
  [ -9, -7, -5, -4, -6 ] ]
gap> ShortCycles(u,2);
[ [ 0 ], [ -1 ], [ 1, 2 ], [ 3, 5 ], [ -10, -6 ] ]
gap> ShortCycles(Comm(a,b),10);
[  ]
gap> ShortCycles(a*b,2);
[ [ 0 ], [ 2 ], [ 3 ], [ -26 ], [ 7 ], [ -3 ], [ -1 ] ]
gap> v := RcwaMapping([[-1,2,1],[1,-1,1],[1,-1,1]]);;
```

```
gap> w := RcwaMapping([[-1,3,1],[1,-1,1],[1,-1,1],[1,-1,1]]);;
gap> Order(v);
6
gap> Order(w);
8
gap> ShortCycles(v,10);
[ [ 0, 2, 1 ] ]
gap> ShortCycles(w,10);
[ [ 0, 3, 2, 1 ] ]
```

4.12.2 CycleType

◊ CycleType(f) (attribute)

The *cycle type* of a tame integral rcwa mapping f is denoted by a list of two lists, where the first list contains the lengths of "halved" cycles, hence the cycles belonging to a series producing a cycle of only half of the "usual" length at some point,with the respective multiplicities, and the second list is the set of the lengths of all other cycles, sortedby increasing length.

Example

```
gap> CycleType(t);
[ [ 2 ], [  ] ]
gap> CycleType(RcwaMapping([[1,1,1],[1,-1,1]]));
[ [  ], [ 2 ] ]
gap> CycleType(v);
[ [ 6 ], [  ] ]
gap> CycleType(w);
[ [ 8 ], [  ] ]
gap> g := RcwaMapping([[-1,0,1],[1,2,1],[1,0,1],[1,-2,1]]);;
gap> CycleType(g);
[ [ 2 ], [ 1, 2 ] ]
gap> CycleType(Comm(a,b));
[ [  ], [ 1, 6 ] ]
```

4.12.3 StandardConjugate

◊ StandardConjugate(f) (attribute)

Some "nice" canonical representative of the conjugacy class of the bijective integral rcwa mapping f in the whole group RCWA(ℤ). Two integral rcwa mappings are conjugate in RCWA(ℤ) if and only if their "standard conjugates" are equal.

Example

```
gap> w_std := StandardConjugate(w);
<integral rcwa mapping with modulus 4>
gap> Order(w_std);
8
gap> ShortCycles(w_std,8);
[ [ 0, 1, 2, 3 ] ]
gap> ab := Comm(a,b);
<bijective integral rcwa mapping with modulus 18>
```

```
gap> ab_std := StandardConjugate(ab);
<integral rcwa mapping with modulus 7>
gap> Display(ab_std);

Integral rcwa mapping with modulus 7

                n mod 7                |                  n^f
---------------------------------------+---------------------------------------
  0                                    | n
  1 2 3 4 5                            | n + 1
  6                                    | n - 5

gap> f := RcwaMapping([[1,1,1],[1, 4,1],[1,1,1],[2,-2,1],
>                      [1,0,2],[1,-5,1],[1,1,1],[2,-2,1]]);
<integral rcwa mapping with modulus 8>
gap> f_std := StandardConjugate(f);
<integral rcwa mapping with modulus 3>
gap> Display(f_std);

Integral rcwa mapping with modulus 3

                n mod 3                |                  n^f
---------------------------------------+---------------------------------------
  0 1                                  | n + 1
  2                                    | n - 2
```

4.12.4 StandardizingConjugator

◊ StandardizingConjugator(f) (attribute)

An rcwa mapping mapping x, such that f^x is the "standard" representative of the conjugacy class of the bijective integral rcwa mapping f in the whole integral residue class-wise affine group.

Example

```
gap> ab_tostd := StandardizingConjugator(ab);
<integral rcwa mapping with modulus 18>
gap> Display(ab_tostd);

Integral rcwa mapping with modulus 18

               n mod 18                |                  n^f
---------------------------------------+---------------------------------------
   0  9                                | 28n/9
   1 10                                | (7n + 2)/9
   2 11                                | (28n + 7)/9
   3 12                                | (28n + 42)/9
   4 13                                | (7n + 80)/9
   5 14                                | (7n + 55)/9
   6                                   | (7n + 156)/18
   7 16                                | (7n + 68)/9
```

```
   8 17                                  | (28n - 35)/9
  15                                     | (7n + 57)/18

gap> ab^ab_tostd = ab_std;
true
gap> f_tostd := StandardizingConjugator(f);
<integral rcwa mapping with modulus 16>
gap> Display(f_tostd);

Integral rcwa mapping with modulus 16

               n mod 16                  |                  n^f
-----------------------------------------+---------------------------------------
   0  8                                  | 9n/8
   1  9                                  | (9n - 1)/8
   2 10                                  | (9n + 6)/8
   3 11                                  | (9n + 5)/8
   4                                     | (9n + 44)/16
   5 13                                  | (9n - 29)/8
   6 14                                  | (9n - 6)/8
   7 15                                  | (9n - 7)/8
  12                                     | (9n + 20)/16

gap> f^f_tostd = f_std;
true
```

4.12.5 IsConjugate

◊ `IsConjugate( R, f, g )` (method)

Checks whether the bijective integral rcwa mappings `f` and `g` are conjugate in the whole group RCWA(ℤ), e.g. via comparing their "standard conjugates". This may fail or run into an infinite loop. The argument `R` has to be `RCWA( Integers )`.

Example

```
gap> IsConjugate(RCWA(Integers),w,w_std);
true
gap> IsConjugate(RCWA(Integers),a,b);
false
gap> IsConjugate(RCWA(Integers),ab,ab_std);
true
```

Chapter 5

Residue Class-Wise Affine Groups

This chapter describes the functionality available for calculating with rcwa groups.

5.1 The categories of rcwa groups

5.1.1 IsRcwaGroup

◊ `IsRcwaGroup( G )` (filter)

The category of all rcwa groups.

5.1.2 IsRationalBasedRcwaGroup

◊ `IsRationalBasedRcwaGroup( G )` (filter)

The category of all “rational-based” (hence all integral and all semilocal integral) rcwa groups.

5.1.3 IsIntegralRcwaGroup

◊ `IsIntegralRcwaGroup( G )` (filter)

The category of all integral rcwa groups.

5.1.4 IntegralRcwaGroupsFamily

◊ `IntegralRcwaGroupsFamily` (family)

The family of all integral rcwa groups.
All integral rcwa groups are supergroups of

5.1.5 TrivialIntegralRcwaGroup

◊ `TrivialIntegralRcwaGroup` (global variable)

The trivial integral rcwa group.

5.1.6 IsSemilocalIntegralRcwaGroup

◊ IsSemilocalIntegralRcwaGroup(G) (filter)

The category of all semilocal integral rcwa groups.

5.1.7 IsModularRcwaGroup

◊ IsModularRcwaGroup(G) (filter)

The category of all modular rcwa groups.
All rcwa groups over the ring R are subgroups of

5.1.8 RCWA

◊ RCWA(R) (function)

The group RCWA(R) of all bijective rcwa mappings over the ring R. This group is (at least in general) not finitely generated, hence no generators are stored.

Example

```
gap> RCWA_Z := RCWA(Integers);
RCWA(Z)
gap> Size(RCWA_Z);
infinity
gap> IsFinitelyGeneratedGroup(RCWA_Z);
false
gap> One(RCWA_Z);
IdentityMapping( Integers )
gap> IsSubgroup(RCWA_Z, Group(RcwaMapping([[-1,0,1]]),
>                             RcwaMapping((1,2,3),[1..4]),
>                             RcwaMapping(2,[[0,1],[1,0],[2,3],[3,2]])));
true
```

5.2 Constructing rcwa groups

Rcwa groups can be constructed using either Group, GroupByGenerators or GroupWithGenerators, as usual (see reference manual). Note that currently except of the whole groups RCWA(R) only finitely generated rcwa groups are supported.

Example

```
gap> g := RcwaMapping([[1,0,1],[1,1,1],[3,6,1],
>                      [1,0,3],[1,1,1],[3,6,1],
>                      [1,0,1],[1,1,1],[3,-21,1]]);;
gap> h := RcwaMapping([[1,0,1],[1,1,1],[3,6,1],
>                      [1,0,3],[1,1,1],[3,-21,1],
>                      [1,0,1],[1,1,1],[3,6,1]]);;
gap> Order(g);
9
```

```
gap> Order(h);
9
gap> G := Group(g,h);
<integral rcwa group with 2 generators>
gap> Size(G);
infinity
```

Another possible way to get an rcwa group is by “translating” a permutation group, or by taking the image of an rcwa representation.

5.2.1 IntegralRcwaGroupByPermGroup

◊ IntegralRcwaGroupByPermGroup(G) (function)
◊ RcwaGroupByPermGroup(G) (function)

Constructs an integral rcwa group isomorphic to the (finite) permutation group G, which acts on the range [1 .. LargestMovedPoint(G)] as G does.

Example

```
gap> H := RcwaGroupByPermGroup(Group((1,2),(3,4),(5,6),(7,8),
>                                    (1,3)(2,4),(1,3,5,7)(2,4,6,8)));
<integral rcwa group with 6 generators>
gap> Size(H);
384
gap> IsSolvable(H);
true
gap> List(DerivedSeries(H),Size);
[ 384, 96, 32, 2, 1 ]
```

5.2.2 IsomorphismIntegralRcwaGroup

◊ IsomorphismIntegralRcwaGroup(G) (attribute)
◊ IsomorphismRcwaGroup(G) (attribute)

A faithful integral rcwa representation of the group G. Currently only supported for finite groups.

Example

```
gap> G := GL(2,5);
GL(2,5)
gap> IsomorphismRcwaGroup(G);
CompositionMapping(
[ (1,2,4,8)(3,6,12,18)(5,10,17,22)(9,15,21,24)(13,14,19,23),
  (1,3,7)(2,5,11)(4,9,16)(6,13,12)(8,14,20)(10,18,17)(15,22,21)(19,24,23)
 ] -> [ <integral rcwa mapping with modulus 24>,
  <integral rcwa mapping with modulus 24> ], <action isomorphism> )
```

5.2.3 Display

◇ Display(G) (method)

Displays the rcwa group G.

Example

```
gap> a := RcwaMapping([[3,0,2],[3, 1,4],[3,0,2],[3,-1,4]]);;
gap> b := RcwaMapping([[3,0,2],[3,13,4],[3,0,2],[3,-1,4]]);;
gap> c := RcwaMapping([[3,0,2],[3, 1,4],[3,0,2],[3,11,4]]);;
gap> H := Group(a,b);
<integral rcwa group with 2 generators>
gap> Display(H);

Integral rcwa group, generated by

[

Bijective integral rcwa mapping with modulus 4

                n mod 4                 |                  n^f
----------------------------------------+---------------------------------------
  0 2                                   | 3n/2
  1                                     | (3n + 1)/4
  3                                     | (3n - 1)/4


Bijective integral rcwa mapping with modulus 4

                n mod 4                 |                  n^f
----------------------------------------+---------------------------------------
  0 2                                   | 3n/2
  1                                     | (3n + 13)/4
  3                                     | (3n - 1)/4

]
```

5.3 Computing with rcwa groups

5.3.1 \in

◇ \in(g, G) (method)

Tries to figure out whether g is an element of G or not. In case G is finite, the result is correct provided that the result given by IsomorphismPermGroup(G) is. If G is infinite and g is not in G, this method may run into an infinite loop.

Example

```
gap> u := RcwaMapping([[3,0,5],[9,1,5],[3,-1,5],[9,-2,5],[9,4,5]]);;
gap> u in H;
```

```
false
```

5.3.2 Size

◊ Size(G) (method)

Tries to compute the order of the rcwa group G.

This is a probabilistic method. It may return the size of a proper factor group of G, or run in an infinite loop. You can increase the option value Steps to decrease the probability of getting a wrong result. The default value for Steps is 10.

Example

```
gap> g1 := RcwaMapping((1,2),[1..2]);
<integral rcwa mapping with modulus 2>
gap> g2 := RcwaMapping((1,2,3),[1..3]);
<integral rcwa mapping with modulus 3>
gap> g3 := RcwaMapping((1,2,3,4,5),[1..5]);
<integral rcwa mapping with modulus 5>
gap> G := Group(g1,g2,g3);
<integral rcwa group with 3 generators>
gap> Size(G);
265252859812191058636308480000000
```

5.3.3 IsomorphismPermGroup

◊ IsomorphismPermGroup(G) (method)

Tries to compute an isomorphism from a finite rcwa group G to some permutation group.

This is a probabilistic method, also. It may return a homomorphism to a permutation group isomorphic to a proper factor group of G. You can increase the option value Steps to decrease the probability of getting a wrong result. The default value for Steps is 10.

Example

```
gap> H := Group(g1,g2);
<integral rcwa group with 2 generators>
gap> phi := IsomorphismPermGroup(H);
[ <bijective integral rcwa mapping with modulus 2, of order 2>,
  <bijective integral rcwa mapping with modulus 3, of order 3> ] ->
[ (1,2)(3,4)(5,6), (1,2,3)(4,5,6) ]
```

5.3.4 NiceMonomorphism

◊ NiceMonomorphism(G) (method)

Returns the result of IsomorphismPermGroup in case the group is supposed to be finite by Size, and IdentityMapping(G) otherwise.

5.3.5 NiceObject

◊ NiceObject(G) (method)

Returns the image of NiceMonomorphism(G).

Example

```
gap> NiceObject(G);
Group([ (1,2)(3,4)(5,6)(7,8)(9,10)(11,12)(13,14)(15,16)(17,18)(19,20)(21,
    22)(23,24)(25,26)(27,28)(29,30), (1,2,3)(4,5,6)(7,8,9)(10,11,12)(13,14,
    15)(16,17,18)(19,20,21)(22,23,24)(25,26,27)(28,29,30),
  (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)(26,
    27,28,29,30) ])
```

5.3.6 Modulus (Modulus of an rcwa group)

◊ Modulus(G) (method)

Computes the modulus of the rcwa group G.
See also Modulus (4.3.2) for rcwa mappings, and IsTame (5.3.10).

Example

```
gap> Modulus(G);
30
gap> Modulus(Group(a,b));
0
```

5.3.7 PrimeSet (Prime set of an rcwa group)

◊ PrimeSet(G) (operation)

Computes the prime set of the rcwa group G.
See also PrimeSet (4.11.3) for rcwa mappings.

Example

```
gap> PrimeSet(G);
[ 2, 3, 5 ]
gap> PrimeSet(H);
[ 2, 3 ]
```

5.3.8 IsFlat (Flat rcwa group)

◊ IsFlat(G) (property)

Determines whether or not the rcwa group G is flat.
See also IsFlat (4.4.3) for rcwa mappings.

Example

```
gap> IsFlat(AlternatingGroup(IsIntegralRcwaGroup,5));
true
gap> IsFlat(Group(a,b));
false
gap> IsFlat(Group(g));
false
```

5.3.9 IsClassWiseOrderPreserving (Class-wise order-preserving rcwa group)

◊ IsClassWiseOrderPreserving(G) (property)

Indicates whether the integral rcwa group G is class-wise order-preserving or not.
See also IsClassWiseOrderPreserving (4.4.4) for rcwa mappings.

Example

```
gap> IsClassWiseOrderPreserving(Group(a,b));
true
gap> IsClassWiseOrderPreserving(G);
true
gap> t := RcwaMapping([[-1,0,1]]);;
gap> IsClassWiseOrderPreserving(Group(t,g,h));
false
```

5.3.10 IsTame (Tame rcwa group)

◊ IsTame(G) (property)

Determines whether or not the rcwa group G is tame.
See also IsTame (4.10.2) for rcwa mappings.

Example

```
gap> IsTame(G);
true
gap> IsTame(Group(a,b));
false
gap> IsTame(Group(Comm(a,b),Comm(a,c)));
true
```

5.3.11 ShortOrbits

◊ ShortOrbits(G, S, maxlng) (function)

Computes all finite orbits of the rcwa group G of maximal length maxlng, which intersect non-trivially with the set S.

Example

```
gap> A5 := AlternatingGroup(IsIntegralRcwaGroup,5);;
gap> ShortOrbits(A5,[-10..10],100);
[ [ -14, -13, -12, -11, -10 ], [ -9, -8, -7, -6, -5 ], [ -4, -3, -2, -1, 0 ],
  [ 1, 2, 3, 4, 5 ], [ 6, 7, 8, 9, 10 ] ]
gap> Action(A5,last[2]);
Group([ (1,2,3,4,5), (3,4,5) ])
gap> ab := Comm(a,b);; ac := Comm(a,c);;
gap> G := Group(ab,ac);;
gap> orb := ShortOrbits(G,[-20..20],100);
[ [ -51, -48, -42, -39, -25, -23, -22, -20, -19, -17 ], [ -18 ],
  [ -33, -30, -24, -21, -16, -14, -13, -11, -10, -8 ],
  [ -15, -12, -7, -6, -5, -4, -3, -2, -1, 1 ], [ -9 ], [ 0 ],
  [ 2, 3, 4, 5, 6, 7, 8, 10, 12, 15 ], [ 9 ],
  [ 11, 13, 14, 16, 17, 19, 21, 24, 30, 33 ], [ 18 ],
```

```
  [ 20, 22, 23, 25, 26, 28, 39, 42, 48, 51 ] ]
gap> Action(G,orb[1]);
Group([ (2,6,8,10,4,7), (1,5,7,9,3,6) ])
gap> ShortOrbits(Group(u),[-30..30],100);
[ [ -13, -8, -7, -5, -4, -3, -2 ], [ -10, -6 ], [ -1 ], [ 0 ], [ 1, 2 ],
  [ 3, 5 ], [ 24, 36, 39, 40, 44, 48, 60, 65, 67, 71, 80, 86, 93, 100, 112,
      128, 138, 155, 167, 187, 230, 248, 312, 446, 520, 803, 867, 1445 ] ]
```

5.4 Properties of RCWA(Z)

5.4.1 NrConjugacyClassesOfRCWAZOfOrder

◊ NrConjugacyClassesOfRCWAZOfOrder(ord) (function)

Computes the number of conjugacy classes of the whole group RCWA(ℤ) of elements of order ord.

Example

```
gap> NrConjugacyClassesOfRCWAZOfOrder(2);
infinity
gap> NrConjugacyClassesOfRCWAZOfOrder(105);
218
```

5.5 Predefined rcwa groups

There are methods for constructing various types of groups as integral rcwa groups. They are just using the ad hoc-translation provided by IntegralRcwaGroupByPermGroup, for the matrix groups after turning to the image under IsomorphismPermGroup. So far, the provided methods cover all groups listed in the sections concerning basic groups and classical groups in the chapter describing the group libraries in the reference manual.

Example

```
gap> C2 := CyclicGroup(IsIntegralRcwaGroup,2);
<integral rcwa group with 1 generator>
gap> G := ExtraspecialGroup(IsIntegralRcwaGroup,27,3);
<integral rcwa group with 3 generators>
gap> IsAbelian(G);
false
gap> Exponent(G);
3
gap> S4 := SymmetricGroup(IsIntegralRcwaGroup,4);
<integral rcwa group with 2 generators>
gap> Size(S4);
24
gap> G := SylowSubgroup(S4,2);
<integral rcwa group with 3 generators, of size 8>
gap> IdGroup(G);
[ 8, 3 ]
gap> A5 := AlternatingGroup(IsIntegralRcwaGroup,5);
<integral rcwa group with 2 generators>
```

```
gap> Size(A5);
60
gap> IsSimple(A5);
true
gap> G := GL(IsIntegralRcwaGroup,2,3);
<integral rcwa group with 2 generators>
gap> Size(G);
48
```

Chapter 6

Getting Information about Computations

6.1 The Info class of the package

6.1.1 RcwaInfo

◊ RcwaInfo (info class)

This is the Info class of the RCWA package (see section *Info Functions* in the GAP Reference Manual for a description of the Info mechanism).

6.1.2 RCWAInfo

◊ RCWAInfo(n) (function)

Returns: Nothing.

For convenience: `RCWAInfo(n)` is a shorthand for `SetInfoLevel(RcwaInfo,n)`.

Example

```
gap> RCWAInfo(3);
gap> a := RcwaMapping([[3,0,2],[3, 1,4],[3,0,2],[3,-1,4]]);;
gap> T := RcwaMapping([[1,0,2],[3,1,2]]);;
gap> v := RcwaMapping([[-1,2,1],[1,-1,1],[1,-1,1]]);;
gap> Order(a);
#I  The mapping IntegralRcwaMapping( [ [ 3, 0, 2 ], [ 3, 1, 4 ], [ 3, 0, 2 ],
  [ 3, -1, 4 ]
 ] ) has a cycle longer than 2 times the square of its modulus, hence we claim\
 its order is infinite, although the validity of this criterium has not been p\
roved so far.
infinity
gap> IsTame(T);
#I  The 4th power of IntegralRcwaMapping( [ [ 1, 0, 2 ], [ 3, 1, 2 ]
 ] ) has Modulus
16; this is larger than the square of the modulus of the base, so we claim the\
 mapping is wild, although the validity of this criterium has not yet been pro\
ved.
false
gap> StandardizingConjugator(v);
#I  StandardConjugate for IntegralRcwaMapping(
[ [ -1, 2, 1 ], [ 1, -1, 1 ], [ 1, -1, 1 ] ] )
```

```
#I  A set of representatives for the series of 'halved' cycles is [ rec(
      pts := [ 3, -1, -2, -3, 5, 4 ],
      HalvedAt := 0 ) ].
#I  A set of representatives for the series of 'not halved' cycles is [  ].
#I  The cycle type is [ [ 6 ], [  ] ].
#I  [Preimage, image] - pairs defining a standardizing conjugator are
[ [ 0, 0 ], [ 2, 1 ], [ 1, 2 ], [ 0, 0 ], [ 2, 1 ], [ 1, 2 ], [ 3, -3 ],
  [ -1, -2 ], [ -2, -1 ], [ -3, 3 ], [ 5, 4 ], [ 4, 5 ] ].
<integral rcwa mapping with modulus 3>
```

Chapter 7

Auxiliary functions

7.1 Building the manual

7.1.1 BuildRCWAManual

◊ `BuildRCWAManual( )` (function)

Returns: Nothing.

This function builds the manual of the RCWA package in the file formats LaTeX, DVI, Postscript, PDF and HTML.

This is done using the GAPDoc package by Frank Lbeck and Max Neunh÷ffer.

7.2 The testing routine

7.2.1 RCWATest

◊ `RCWATest( [, test1[, test2[, ...]]] )` (function)

Returns: Nothing.

Performs tests of the RCWA package.

The arguments `test1`, `test2`, ..., if present, are names of tests.

Available tests are:

`"z_pi"` Arithmetic in the rings $\mathbb{Z}_\pi$.

`"integral"` Computations with integral rcwa mappings and integral rcwa groups.

`"semilocal"` Computations with semilocal integral rcwa mappings and semilocal integral rcwa groups.

`"modular"` Computations with modular rcwa mappings and modular rcwa groups.

`"all"` All tests.

The default (if no argument is given) is `"all"`. In case that all tests are to be performed, the function makes use of an adaptation of the test file `tst/testall.g` of the GAP library to this package.

7.3 The color markup

By default, RCWA switches on coloring of GAP prompts and user input and uses colored markup for displaying its online help texts.

If you do not like this or if your system does not support it, you can switch this off by setting the option value `BlackAndWhite` to `true` when loading the package: `RequirePackage( "rcwa" : BlackAndWhite );`.

Chapter 8

Examples

In this chapter, we would like to give some "nice" examples of rcwa mappings and groups generated by them. The rcwa mappings used in this chapter can be found in the file `rcwa/examples/examples.gap`, so there is no need to extract them from the manual files.

8.1 Replacing the Collatz mapping by conjugates

This is probably not the most interesting application of this package, but we can turn the Collatz-($3n+1$-) problem into an equivalent question by replacing the Collatz mapping `T` by one of its conjugates under an element of the pointwise stabilizer of 1 in the setwise stabilizer of the positive integers of the integral residue class-wise affine group.

Define the Collatz mapping:

Example

```
gap> T := RcwaMapping([[1,0,2],[3,1,2]]);
<integral rcwa mapping with modulus 2>
```

A suitable bijection:

Example

```
gap> a := RcwaMapping([[3,0,2],[3,1,4],[3,0,2],[3,-1,4]]);
<integral rcwa mapping with modulus 4>
gap> IsBijective(a);
true
gap> 1^a; # The mapping a stabilizes 1
1
```

Some evidence that `a` stabilizes the set of positive integers (this certainly can be proved easily ...):

Example

```
gap> List([1..50],n->n^a);
[ 1, 3, 2, 6, 4, 9, 5, 12, 7, 15, 8, 18, 10, 21, 11, 24, 13, 27, 14, 30, 16,
  33, 17, 36, 19, 39, 20, 42, 22, 45, 23, 48, 25, 51, 26, 54, 28, 57, 29, 60,
  31, 63, 32, 66, 34, 69, 35, 72, 37, 75 ]
```

Compute T^a:

Example

```
gap> f := T^a;;
gap> Display(f);

Integral rcwa mapping with modulus 12

               n mod 12                |                  n^f
---------------------------------------+--------------------------------------
   0  6                                | n/2
   1  4  7 10                         | 3n
   2  8                                | (3n + 2)/2
   3                                   | (n + 1)/4
   5 11                                | (3n + 1)/2
   9                                   | (n - 1)/4
```

Have a look at resulting integer sequences:

Example

```
gap> seq := function(n)
>             repeat Print(n,","); n := n^f; until n = 1; Print("1.\n");
>           end;
function( n ) ... end
gap> seq(7);
7,21,5,8,13,39,10,30,15,4,12,6,3,1.
gap> seq(10);
10,30,15,4,12,6,3,1.
gap> seq(20);
20,31,93,23,35,53,80,121,363,91,273,68,103,309,77,116,175,525,131,197,296,445,
1335,334,1002,501,125,188,283,849,212,319,957,239,359,539,809,1214,1822,5466,
2733,683,1025,1538,2308,6924,3462,1731,433,1299,325,975,244,732,366,183,46,
138,69,17,26,40,120,60,30,15,4,12,6,3,1.
```

It seems that they all reach 1, as the original $3n+1$ sequences ... .

8.2 An rcwa representation of a small group

We give an rcwa representation of the 3-Sylow-subgroup of the symmetric group on 9 points. Certainly, this group has a very nice permutation representation, hence for computational purposes, we cannot do better here.

Example

```
gap> r := RcwaMapping([[1,0,1],[1,1,1],[3,-3,1],
>                      [1,0,3],[1,1,1],[3,-3,1],
>                      [1,0,1],[1,1,1],[3,-3,1]]);;
gap> s := RcwaMapping([[1,0,1],[1,1,1],[3,6,1],
>                      [1,0,3],[1,1,1],[3,6,1],
```

```
>                      [1,0,1],[1,1,1],[3,-21,1]]);;
gap> Display(r);

Integral rcwa mapping with modulus 9

               n mod 9                |                 n^f
--------------------------------------+--------------------------------------
  0 6                                 | n
  1 4 7                               | n + 1
  2 5 8                               | 3n - 3
  3                                   | n/3

gap> Display(s);

Integral rcwa mapping with modulus 9

               n mod 9                |                 n^f
--------------------------------------+--------------------------------------
  0 6                                 | n
  1 4 7                               | n + 1
  2 5                                 | 3n + 6
  3                                   | n/3
  8                                   | 3n - 21

gap> G := Group(r,s);
<integral rcwa group with 2 generators>
gap> H := SylowSubgroup(SymmetricGroup(9),3);
Group([ (1,2,3), (4,5,6), (7,8,9), (1,4,7)(2,5,8)(3,6,9) ])
gap> phi := InverseGeneralMapping(IsomorphismGroups(G,H));
[ (1,5,8)(2,6,9)(3,4,7), (1,6,9,2,4,7,3,5,8) ] ->
[ <bijective integral rcwa mapping with modulus 9, of order 3>,
  <bijective integral rcwa mapping with modulus 9, of order 9> ]
```

8.3 An rcwa representation of the symmetric group on 10 points

Firstly, we define some bijections of infinite order and compute commutators:

Example

```
gap> a := RcwaMapping([[3,0,2],[3, 1,4],[3,0,2],[3,-1,4]]);;
gap> b := RcwaMapping([[3,0,2],[3,13,4],[3,0,2],[3,-1,4]]);;
gap> c := RcwaMapping([[3,0,2],[3, 1,4],[3,0,2],[3,11,4]]);;
gap> Order(a);
infinity
gap> Order(b);
infinity
gap> Order(c);
infinity
gap> ab := Comm(a,b);;
gap> ac := Comm(a,c);;
gap> bc := Comm(b,c);;
```

```
gap> Order(ab);
6
gap> Order(ac);
6
gap> Order(bc);
12
```

Now we would like to have a look at [a, b].

Example

```
gap> Display(ab);

Bijective integral rcwa mapping with modulus 18, of order 6

               n mod 18                |                 n^f
---------------------------------------+--------------------------------------
   0  2  3  8  9 11 12 17              | n
   1 10                                | 2n - 5
   4  7 13 16                          | n + 3
   5 14                                | 2n - 4
   6                                   | (n + 2)/2
  15                                   | (n - 5)/2
```

Afterwards, we form the group generated by [a, b] and [a, c] and compute its action on one of its orbits ...

Example

```
gap> G := Group(ab,ac);
<integral rcwa group with 2 generators>
gap> orb := Orbit(G,1);
[ 1, -3, -4, -12, -1, -5, -6, -2, -15, -7 ]
gap> H := Action(G,orb);
Group([ (1,2,3,4,6,8), (3,5,7,6,9,10) ])
gap> Size(H);
3628800
gap> Size(G);
3628800
gap> H := NiceObject(G);
Group([ (2,4,6,10,3,8), (1,3,5,9,2,7) ])
```

Hence, G is isomorphic to the symmetric group on 10 points and acts faithfully on the orbit containing 1. We also would like to know which groups arise if we take as generators either ab, ac or bc and the mapping t, which maps each integer to its additive inverse:

Example

```
gap> t := RcwaMapping([[-1,0,1]]);
Integral rcwa mapping: n -> -n
gap> Order(t);
```

```
2
gap> G := Group(ab,t);
<integral rcwa group with 2 generators>
gap> Size(G);
7257600
gap> H := NiceObject(G);
Group([ (2,7,9,10,4,8)(13,15,16,20,14,18), (1,20)(2,19)(3,18)(4,17)(5,16)(6,
    15)(7,14)(8,13)(9,12)(10,11) ])
gap> H2 := Group((1,2),(1,2,3,4,5,6,7,8,9,10),(11,12));
Group([ (1,2), (1,2,3,4,5,6,7,8,9,10), (11,12) ])
gap> phi := IsomorphismGroups(H,H2);
[ (2,7,9,10,4,8)(13,15,16,20,14,18), (1,20)(2,19)(3,18)(4,17)(5,16)(6,15)(7,
    14)(8,13)(9,12)(10,11) ] ->
[ (3,8,9,4,10,7), (1,3)(2,8)(4,5)(6,7)(9,10)(11,12) ]
```

Hence, the group generated by `ab` and `t` is isomorphic to $C_2 \times S_{10}$. The next group is an extension of a perfect group of order 960:

Example

```
gap> G := Group(ac,t);;
gap> Size(G);
3840
gap> H := NiceObject(G);
Group([ (1,6,8,10,3,7)(12,14,16,19,13,17), (1,20)(2,19)(3,18)(4,17)(5,16)(6,
    15)(7,14)(8,13)(9,12)(10,11) ])
gap> P := DerivedSubgroup(H);
Group([ (1,6,8,2,5)(3,7,9,4,10)(11,17,12,14,18)(13,15,20,16,19),
  (1,2,8,4,6)(3,10,9,7,5)(11,19,16,15,13)(12,17,18,14,20) ])
gap> Size(P);
960
gap> IsPerfect(P);
true
gap> IdGroup(PerfectGroup(960,1));
[ 960, 11357 ]
gap> IdGroup(PerfectGroup(960,2));
[ 960, 11358 ]
gap> IdGroup(P);
[ 960, 11358 ]
```

The last group is infinite:

Example

```
gap> G := Group(bc,t);;
gap> Size(G);
infinity
gap> Order(bc*t);
infinity
gap> Modulus(G);
18
```

8.4 Twisting 257-cycles into an rcwa mapping with modulus 32

We define an rcwa mapping `x` of order 257 with modulus 32 (the easiest way to construct such a mapping is to prescribe the graph and then assign suitable affine mappings to its vertices).

Example

```
gap> x := RcwaMapping([[ 16,  2,  1], [ 16, 18,  1],
>                      [  1, 16,  1], [ 16, 18,  1],
>                      [  1, 16,  1], [ 16, 18,  1],
>                      [  1, 16,  1], [ 16, 18,  1],
>                      [  1, 16,  1], [ 16, 18,  1],
>                      [  1, 16,  1], [ 16, 18,  1],
>                      [  1, 16,  1], [ 16, 18,  1],
>                      [  1,  0, 16], [ 16, 18,  1],
>                      [  1,-14,  1], [ 16, 18,  1],
>                      [  1,-14,  1], [ 16, 18,  1],
>                      [  1,-14,  1], [ 16, 18,  1],
>                      [  1,-14,  1], [ 16, 18,  1],
>                      [  1,-14,  1], [ 16, 18,  1],
>                      [  1,-14,  1], [ 16, 18,  1],
>                      [  1,-14,  1], [  1,-31,  1]]);;
gap> Display(x);

Integral rcwa mapping with modulus 32

               n mod 32                |                 n^f
---------------------------------------+--------------------------------------
   0                                   | 16n + 2
   1  3  5  7  9 11 13 15 17 19 21 23  |
  25 27 29                             | 16n + 18
   2  4  6  8 10 12 14                 | n + 16
  16                                   | n/16
  18 20 22 24 26 28 30                 | n - 14
  31                                   | n - 31

gap> Order(x);
257
gap> List([-20..20],n->n^x);
[ -4, -286, -2, -254, -1, -222, -28, -190, -26, -158, -24, -126, -22, -94,
  -20, -62, -18, -30, -16, -32, 2, 34, 18, 66, 20, 98, 22, 130, 24, 162, 26,
  194, 28, 226, 30, 258, 1, 290, 4, 322, 6 ]
```

Certainly, we would like to know how a cycle of this permutation looks like.

Example

```
gap> Cycle(x,[1],0);
[ 0, 2, 18, 4, 20, 6, 22, 8, 24, 10, 26, 12, 28, 14, 30, 16, 1, 34, 50, 36,
  52, 38, 54, 40, 56, 42, 58, 44, 60, 46, 62, 48, 3, 66, 82, 68, 84, 70, 86,
  72, 88, 74, 90, 76, 92, 78, 94, 80, 5, 98, 114, 100, 116, 102, 118, 104,
  120, 106, 122, 108, 124, 110, 126, 112, 7, 130, 146, 132, 148, 134, 150,
  136, 152, 138, 154, 140, 156, 142, 158, 144, 9, 162, 178, 164, 180, 166,
```

```
  182, 168, 184, 170, 186, 172, 188, 174, 190, 176, 11, 194, 210, 196, 212,
  198, 214, 200, 216, 202, 218, 204, 220, 206, 222, 208, 13, 226, 242, 228,
  244, 230, 246, 232, 248, 234, 250, 236, 252, 238, 254, 240, 15, 258, 274,
  260, 276, 262, 278, 264, 280, 266, 282, 268, 284, 270, 286, 272, 17, 290,
  306, 292, 308, 294, 310, 296, 312, 298, 314, 300, 316, 302, 318, 304, 19,
  322, 338, 324, 340, 326, 342, 328, 344, 330, 346, 332, 348, 334, 350, 336,
  21, 354, 370, 356, 372, 358, 374, 360, 376, 362, 378, 364, 380, 366, 382,
  368, 23, 386, 402, 388, 404, 390, 406, 392, 408, 394, 410, 396, 412, 398,
  414, 400, 25, 418, 434, 420, 436, 422, 438, 424, 440, 426, 442, 428, 444,
  430, 446, 432, 27, 450, 466, 452, 468, 454, 470, 456, 472, 458, 474, 460,
  476, 462, 478, 464, 29, 482, 498, 484, 500, 486, 502, 488, 504, 490, 506,
  492, 508, 494, 510, 496, 31 ]
gap> Length(last);
257
```

8.5 Two mappings with isomorphic graphs, but different orders

We define two very similar rcwa mappings with modulus 6, and compute their orders:

Example

```
gap> g := RcwaMapping([[2,2,1],[1, 4,1],[1,0,2],[2,2,1],[1,-4,1],[1,-2,1]]);
<integral rcwa mapping with modulus 6>
gap> h := RcwaMapping([[2,2,1],[1,-2,1],[1,0,2],[2,2,1],[1,-1,1],[1, 1,1]]);
<integral rcwa mapping with modulus 6>
gap> Order(g);
7
gap> Order(h);
12
gap> Display(g);

Bijective integral rcwa mapping with modulus 6, of order 7

                n mod 6                |                 n^f
---------------------------------------+---------------------------------------
  0 3                                  | 2n + 2
  1                                    | n + 4
  2                                    | n/2
  4                                    | n - 4
  5                                    | n - 2
```

Example

```
gap> Display(h);

Bijective integral rcwa mapping with modulus 6, of order 12

                n mod 6                |                 n^f
---------------------------------------+---------------------------------------
  0 3                                  | 2n + 2
```

```
  1                                   | n - 2
  2                                   | n/2
  4                                   | n - 1
  5                                   | n + 1
```

Example

```
gap> Display(g - h);

Integral rcwa mapping with modulus 6

                n mod 6               |                 n^f
--------------------------------------+--------------------------------------
  0 2 3                               | 0
  1                                   | 6
  4 5                                 | -3

gap> Display(Comm(g,h));

Bijective integral rcwa mapping with modulus 6

                n mod 6               |                 n^f
--------------------------------------+--------------------------------------
  0                                   | n + 3
  1 4                                 | n
  2 5                                 | n - 6
  3                                   | n + 9

gap> Order(Comm(g,h));
infinity
```

Both of these mappings have an rcwa graph consisting of one cycle of length 3 and one of length 4. The mapping `g` has order $3+4=7$, because the two cycles are always passed consecutively, since running through one of them results in an addition of 6, and running through the other in a subtraction of 6, while `h` has order $3\cdot 4=12$, because here, the two cycles are passed independant from each other, since running through one of them always ends up in the starting point.

8.6 A group with a free abelian normal subgroup of rank 12

Firstly, we define our group `G`:

Example

```
gap> v := RcwaMapping([[-1,2,1],[1,-1,1],[1,-1,1]]);;
gap> w := RcwaMapping([[-1,3,1],[1,-1,1],[1,-1,1],[1,-1,1]]);;
gap> Order(v);
6
gap> Order(w);
8
```

```
gap> G := Group(v,w);;
gap> Size(G);
infinity
gap> IsAbelian(G);
false
gap> Modulus(G);
12
```

Then, we construct the normal subgroup `N` as the normal closure of some element `z`:

Example

```
gap> z := (v*w*v)^6;
<bijective integral rcwa mapping with modulus 12>
gap> Display(z);

Bijective integral rcwa mapping with modulus 12

               n mod 12                |                  n^f
---------------------------------------+---------------------------------------
   0  2  3  6  8  9                    | n
   1  4  7 10 11                       | n - 12
   5                                   | n + 12
```

We are looking for generators:

Example

```
gap> line := g -> List([0..11], n -> n^g - n);;
gap> M := [];;
gap> M[1] := line(z);
[ 0, -12, 0, 0, -12, 12, 0, -12, 0, 0, -12, -12 ]
gap> M[2] := line(z^v);
[ -12, 0, 0, -12, 12, 0, -12, 0, 0, -12, -12, 0 ]
gap> M[3] := line(z^w);
[ -12, 0, 0, 0, 12, 0, -12, 0, 0, -12, -12, 12 ]
gap> RankMat(M);
3
gap> M[4] := line(z^(v^-1));;
gap> M[5] := line(z^(w^-1));;
gap> RankMat(M);
5
gap> M[6] := line(z^(v*w));;
gap> RankMat(M);
5
gap> M[6] := line(z^(w*v));;
gap> RankMat(M);
6
gap> M[7] := line(z^(v^2));;
gap> RankMat(M);
7
```

```
gap> M[8] := line(z^(w^2));;
gap> RankMat(M);
8
gap> M[9] := line(z^(v*w*v));;
gap> RankMat(M);
8
gap> M[9] := line(z^(w*v*w));;
gap> RankMat(M);
9
gap> M[10] := line(z^(v^-2));;
gap> RankMat(M);
10
gap> M[11] := line(z^(w^-2));;
gap> RankMat(M);
10
gap> M[11] := line(z^(v^2*w));;
gap> RankMat(M);
11
gap> M[12] := line(z^(w*v^2));;
gap> RankMat(M);
12
gap> Display(M);
[ [    0,  -12,    0,    0,  -12,   12,    0,  -12,    0,    0,  -12,  -12 ],
  [  -12,    0,    0,  -12,   12,    0,  -12,    0,    0,  -12,  -12,    0 ],
  [  -12,    0,    0,    0,   12,    0,  -12,    0,    0,  -12,  -12,   12 ],
  [    0,    0,  -12,   12,    0,  -12,    0,    0,  -12,  -12,    0,  -12 ],
  [    0,    0,  -12,    0,   12,  -12,   12,    0,   12,    0,    0,  -12 ],
  [    0,    0,   12,   12,    0,   12,    0,    0,   12,  -12,   12,    0 ],
  [    0,    0,   12,   12,    0,   12,    0,    0,   12,  -12,    0,   12 ],
  [    0,    0,    0,   12,    0,  -12,    0,    0,  -12,  -12,   12,  -12 ],
  [    0,   12,   12,    0,   12,    0,    0,  -12,  -12,   12,    0,    0 ],
  [   12,    0,    0,   12,   12,    0,   12,    0,    0,   12,  -12,    0 ],
  [    0,   12,   12,    0,   12,    0,    0,  -12,  -12,    0,   12,    0 ],
  [    0,   12,    0,    0,   12,   12,    0,   12,    0,   12,    0,  -12 ] \
]
gap> DeterminantMat(M);
-285315214344192
```

Now, we have our normal subgroup:

Example

```
gap> gens := [z,z^v,z^w,z^(v^-1),z^(w^-1),z^(w*v),z^(v^2),
>             z^(w^2),z^(w*v*w),z^(v^-2),z^(v^2*w),z^(w*v^2)];;
gap> N := Group(gens);
<integral rcwa group with 12 generators>
gap> IsAbelian(N);
true
gap> Size(N);
infinity
```

8.7 Behaviour of the moduli of powers

In this section, we give some examples illustrating how different the series of the moduli of powers of a given integral rcwa mapping can look like.

Example

```
gap> List([0..4],i->Modulus(a^i));
[ 1, 4, 16, 64, 256 ]
gap> List([0..6],i->Modulus(ab^i));
[ 1, 18, 18, 18, 18, 18, 1 ]
gap> List([0..3],i->Modulus(r^i));
[ 1, 9, 9, 1 ]
gap> List([0..9],i->Modulus(s^i));
[ 1, 9, 9, 27, 27, 27, 27, 27, 27, 1 ]
gap> List([0..7],i->Modulus(g^i));
[ 1, 6, 12, 12, 12, 12, 6, 1 ]
gap> List([0..3],i->Modulus(u^i));
[ 1, 5, 25, 125 ]
gap> List([0..2],i->Modulus(y^i));
[ 1, 18, 324 ]
gap> List([0..6],i->Modulus(v^i));
[ 1, 3, 3, 3, 3, 3, 1 ]
gap> List([0..8],i->Modulus(w^i));
[ 1, 4, 4, 4, 4, 4, 4, 4, 1 ]
gap> z := RcwaMapping([[2,  1, 1],[1,  1,1],[2, -1,1],[2, -2,1],
>                      [1,  6, 2],[1,  1,1],[1, -6,2],[2,  5,1],
>                      [1,  6, 2],[1,  1,1],[1,  1,1],[2, -5,1],
>                      [1,  0, 1],[1, -4,1],[1,  0,1],[2,-10,1]]);;
gap> Order(z);
infinity
gap> Display(z);

Bijective integral rcwa mapping with modulus 16, of order infinity

               n mod 16                |                  n^f
---------------------------------------+---------------------------------------
   0                                   | 2n + 1
   1  5  9 10                          | n + 1
   2                                   | 2n - 1
   3                                   | 2n - 2
   4  8                                | (n + 6)/2
   6                                   | (n - 6)/2
   7                                   | 2n + 5
  11                                   | 2n - 5
  12 14                                | n
  13                                   | n - 4
  15                                   | 2n - 10

gap> List([0..35],i->Modulus(z^i));
[ 1, 16, 32, 64, 64, 128, 128, 128, 128, 128, 128, 256, 256, 256, 256, 256,
  256, 512, 512, 512, 512, 512, 512, 1024, 1024, 1024, 1024, 1024, 1024,
  2048, 2048, 2048, 2048, 2048, 2048, 4096 ]
gap> e1 := RcwaMapping([[1,4,1],[2,0,1],[1,0,2],[2,0,1]]);;
```

```
gap> Order(e1);
infinity
gap> List([1..20],i->Modulus(e1^i));
[ 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 ]
gap> e2 := RcwaMapping([[1,4,1],[2,0,1],[1,0,2],[1,0,1],
>                       [1,4,1],[2,0,1],[1,0,1],[1,0,1]]);;
gap> Order(e2);
infinity
gap> List([1..20],i->Modulus(e2^i));
[ 8, 4, 8, 4, 8, 4, 8, 4, 8, 4, 8, 4, 8, 4, 8, 4, 8, 4, 8, 4 ]
gap> Display(e2);

Bijective integral rcwa mapping with modulus 8, of order infinity

                n mod 8                 |                  n^f
----------------------------------------+---------------------------------------
  0 4                                   | n + 4
  1 5                                   | 2n
  2                                     | n/2
  3 6 7                                 | n

gap> Display(e2^2);

Integral rcwa mapping with modulus 4

                n mod 4                 |                  n^f
----------------------------------------+---------------------------------------
  0                                     | n + 8
  1 2 3                                 | n
```

References

[1] Stefan Kohl. Residue class-wise affine representations of groups, 2002. Draft. 6

[2] Jeffrey C. Lagarias. 3x+1 problem annotated bibliography, 1998. http://www.research.att.com/ jcl/doc/3x+1bib.ps. 4

[3] Frank Lübeck and Max Neunhöffer. *GAPDoc (version 0.99)*. RWTH Aachen, 2002. GAP package, available at http://www.math.rwth-aachen.de/ Frank.Luebeck. 2

[4] Hans Läuchli and Peter M. Neumann. On linearly ordered sets and permutation groups of countable degree. *Arch. Math. Logic*, 27:189–192, 1988. 4

[5] Leonard Soicher. *GRAPE – GRaph Algorithms using PErmutation groups (version 4.1)*. Queen Mary, University of London, 2002. GAP package, available at http://www.gap-system.org. 2

Index