ALGORITHMS FOR A CLASS OF INFINITE PERMUTATION GROUPS

STEFAN KOHL

ABSTRACT. Motivated by the famous $3n+1$ Conjecture, we call a mapping from $\mathbb{Z}$ to $\mathbb{Z}$ *residue-class-wise affine* if there is a positive integer m such that it is affine on residue classes (mod m). This article describes a collection of algorithms and methods for computing in permutation groups and monoids formed by residue-class-wise affine mappings.

1. INTRODUCTION

Lothar Collatz' famous $3n+1$ Conjecture asserts that iterated application of the mapping

$$T:\ \mathbb{Z}\ \to\ \mathbb{Z},\quad n\ \mapsto\ \begin{cases} n/2 & \text{if } n \text{ is even,}\\ (3n+1)/2 & \text{if } n \text{ is odd}\end{cases}$$

to a positive integer yields 1 after a finite number of steps. The question whether this holds is still open today. See [11] for a comprehensive annotated bibliography.

Motivated by Collatz' conjecture, we make the following general definition:

Definition 1.1. We call a mapping $f:\mathbb{Z}\to\mathbb{Z}$ *residue-class-wise affine* or in short an *rcwa* mapping, if there is a positive integer m such that the restrictions of f to the residue classes $r(m)\in\mathbb{Z}/m\mathbb{Z}$ are all affine, i.e. given by

$$f|_{r(m)}:\ r(m)\to\mathbb{Z},\ \ n\ \mapsto\ \frac{a_{r(m)}\cdot n+b_{r(m)}}{c_{r(m)}}$$

for certain coefficients $a_{r(m)},b_{r(m)},c_{r(m)}\in\mathbb{Z}$ depending on $r(m)$. We call the smallest possible m the *modulus* of f, written $\mathrm{Mod}(f)$.

In general, a mapping from $\mathbb{Z}$ to $\mathbb{Z}$ cannot be described by a finite amount of data. The rcwa mappings form a class of such mappings which can. It is easy to see that they form a monoid under composition.

The following observations are immediate:

Remark 1.2. Any coefficient $c_{r(m)}$ must divide both $a_{r(m)}\cdot r+b_{r(m)}$ and m. Furthermore we have

$$\frac{1}{m}\ \cdot\ \sum_{r(m)\in\mathbb{Z}/m\mathbb{Z}}\frac{c_{r(m)}}{a_{r(m)}}\quad\begin{cases}\leqslant 1 & \text{if } f \text{ is injective,}\\ \geqslant 1 & \text{if } f \text{ is surjective, and}\\ =1 & \text{if } f \text{ is bijective.}\end{cases}$$

Obviously, the Collatz mapping T is an rcwa mapping, and we have $\mathrm{Mod}(T)=2$. It is easy to check that the mapping T is surjective, but not injective – the preimage of an integer n under T is $\{2n,(2n-1)/3\}$ if $n\equiv 2 \bmod 3$, and $\{2n\}$ otherwise. For the mapping T, the expression in Remark 1.2 takes the value $1/2\cdot(2/1+2/3)=4/3\geqslant 1$.

2000 *Mathematics Subject Classification.* 20B40.

A specific example of a bijective rcwa mapping, an rcwa *permutation*, is

$$\alpha : \mathbb{Z} \longrightarrow \mathbb{Z}, \quad n \longmapsto \begin{cases} 2n/3 & \text{if } n \equiv 0 \pmod 3, \\ (4n-1)/3 & \text{if } n \equiv 1 \pmod 3, \\ (4n+1)/3 & \text{if } n \equiv 2 \pmod 3. \end{cases}$$

This permutation has already been investigated by Lothar Collatz in 1932, but its cycle structure is still unknown (cf. Keller [5], Wirsching [12]).

A simpler example of an rcwa permutation is $\tau : n \mapsto n + (-1)^n$. This mapping interchanges the residue classes $0(2) = 2\mathbb{Z}$ and $1(2) = 1+2\mathbb{Z}$ setwise. A natural generalization of this to arbitrary pairs of disjoint residue classes is the following (remember that by the *Chinese Remainder Theorem*, two residue classes $r_1(m_1)$ and $r_2(m_2)$ are disjoint if and only if r_1 and r_2 are incongruent modulo the gcd of m_1 and m_2):

Definition 1.3. Given disjoint residue classes $r_1(m_1)$ and $r_2(m_2)$ of $\mathbb{Z}$, we define the *class transposition* $\tau_{r_1(m_1),r_2(m_2)}$ as the involution which interchanges $r_1 + km_1$ and $r_2 + km_2$ for each integer k and which fixes all other points. Here we assume that $0 \leqslant r_1 < m_1$ and that $0 \leqslant r_2 < m_2$.

This article mainly describes algorithms for computing in groups whose elements are rcwa permutations:

Definition 1.4. We denote the group which is formed by all residue-class-wise affine permutations of $\mathbb{Z}$ by $\mathrm{RCWA}(\mathbb{Z})$. We call its subgroups *residue-class-wise affine* groups, or in short *rcwa* groups.

The set of all class transpositions of $\mathbb{Z}$ generates a subgroup of $\mathrm{RCWA}(\mathbb{Z})$ which we denote by $\mathrm{CT}(\mathbb{Z})$. In [8] it is shown that this group is simple, and that its class of subgroups includes

- the free groups of finite rank,
- all free products of finitely many finite groups,
- certain divisible torsion groups, and
- uncountably many distinct simple groups.

There are finitely generated subgroups of $\mathrm{CT}(\mathbb{Z})$ which do not have finite presentations, and there are also finitely generated subgroups of $\mathrm{CT}(\mathbb{Z})$ for which the membership problem is algorithmically unsolvable (cf. [8], Corollary 4.5). Nevertheless we will see that there are quite practical methods to compute in rcwa groups.

This article describes algorithms and methods for the following tasks:

- Basic tasks:
  - Performing arithmetical operations with rcwa mappings and rcwa permutations, such as multiplying and computing inverses.
  - Performing set-theoretic operations with residue classes, such as computing unions, intersections and differences.
  - Computing images and preimages of set-theoretic unions of residue classes of $\mathbb{Z}$ under rcwa mappings.
- Group constructions:
  - Constructing finite direct products of rcwa groups.
  - Constructing wreath products of rcwa groups with finite groups, and restricted wreath products of rcwa groups with the infinite cyclic group $(\mathbb{Z}, +)$.
  - Constructing an rcwa group which is isomorphic to the free product of given finite groups.

  - Constructing an rcwa group isomorphic to the free group of a given rank.
- Computing in rcwa groups:
  - Computing the order of an rcwa permutation and the order of an rcwa group.
  - Solving the membership problem in 'many' cases.
  - Factoring an element of an rcwa group into a product of the given generators.
  - Determining whether an rcwa group acts transitively on $\mathbb{Z}$ or on a given set-theoretic union of residue classes, and determining the degree of transitivity.
  - Looking for an element of a given rcwa group which maps a given tuple of pairwise distinct integers to another given tuple.
  - Extracting roots of suitable rcwa permutations of finite order.

All of these algorithms and methods are sufficiently practical to be used in solving nontrivial problems.

Illustrative examples are given. The purpose of most of the examples is more to show how the algorithms work than to demonstrate what is computationally feasible. None of them is time- or memory-critical, i.e. none of them needs more than a few seconds on a standard PC, and they do not exhaust the memory of even a rather old machine.

Sometimes rough estimates of the computational complexity are given. However for most of the more advanced methods, an analysis of the runtime- and memory requirements would be delicate. The reason for this is that there are often both very easy and extremely difficult cases which one sometimes cannot easily distinguish a priori. For example, as stated above, the membership problem for rcwa groups is in general algorithmically unsolvable – but it can be solved quickly in many common situations.

In this article, the term 'algorithm' means that termination is guaranteed, while the term 'method' means that termination is not guaranteed. A method may run into an infinite loop or otherwise be unable to produce a result for given arguments. However, if a result is returned then it is correct, unless the descriptor given in brackets after the method number states explicitly that the method returns only a 'guess'.

The algorithms and methods are grouped together in sections, depending on their purpose. To avoid forward references, the ordering of the sections follows mainly the dependencies between the algorithms and methods.

All of the algorithms and methods described in this article are implemented in the author's package RCWA [9] for the computer algebra system GAP [3]. Further examples of their use can be found in the manual of this package.

2. Basic Arithmetic

The purpose of this section is to describe how to represent rcwa mappings and rcwa groups in memory, how to check whether a given rcwa mapping is injective, surjective or bijective, and how to perform basic arithmetic operations with such mappings such as multiplying and computing inverses.

Definition 2.1 (Internal Representation of rcwa Mappings)**.** Let f be an rcwa mapping, and let m denote its modulus. Assume that the restriction of f to a residue class $r(m)$ is given by $n \mapsto (a_{r(m)} \cdot n + b_{r(m)})/c_{r(m)}$. We store f as a *reduced coefficient list*, i.e. as a list of m triples $(a_{r(m)}, b_{r(m)}, c_{r(m)})$ of coprime integers, where all $c_{r(m)}$ are positive.

For example, the coefficient list $((1,0,2),(3,1,2))$ is used to represent the Collatz mapping T, and the class transposition $\tau = \tau_{0(2),1(2)}$ is represented by the coefficient list $((1,1,1),(1,-1,1))$.

Two rcwa mappings are equal if and only if the corresponding coefficient lists are equal. Therefore testing for equality is cheap. In order to obtain the normal form described in Definition 2.1, we need a reduction algorithm:

Algorithm 2.2 (Reduction of Coefficient Lists)**.** Let m be a positive integer. Further let $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m)\in\mathbb{Z}/m\mathbb{Z}}$ be a coefficient list which describes an rcwa mapping and which is possibly not yet in reduced form. This means that the entries of the coefficient triples may be not yet coprime, that the coefficients $c_{r(m)}$ are possibly negative and that m may be any multiple of the actual modulus. We reduce this coefficient list as follows:

(1) For all $r(m) \in \mathbb{Z}/m\mathbb{Z}$, do the following:
   (a) Put $d := \gcd(a_{r(m)}, b_{r(m)}, c_{r(m)})$, and divide $a_{r(m)}$, $b_{r(m)}$ and $c_{r(m)}$ by d.
   (b) If $c_{r(m)} < 0$, then put $a_{r(m)} := -a_{r(m)}$, $b_{r(m)} := -b_{r(m)}$ and $c_{r(m)} := -c_{r(m)}$.
(2) Let $p_1, \ldots, p_l$ be the prime divisors of m, and put $k := 1$.
(3) Check if $\forall (i,j) \in \{1, \ldots, p_k - 1\} \times \{0, \ldots, m/p_k - 1\}$ $a_{j(m)} = a_{i\cdot m/p_k + j(m)}$ $\wedge\ b_{j(m)} = b_{i\cdot m/p_k+j(m)}$ $\wedge$ $c_{j(m)} = c_{i\cdot m/p_k+j(m)}$. If so, then put $m := m/p_k$. Otherwise put $k := k + 1$.
(4) If $k > l$, then go to Step (6).
(5) If $p_k \nmid m$, then put $k := k + 1$.
(6) If $k \leqslant l$, then go to Step (3). Otherwise return the rcwa mapping with modulus m and coefficients $a_{r(m)}$, $b_{r(m)}$ and $c_{r(m)}$, for r running from 0 to $m - 1$.

In Step (2)-(6), the actual modulus of the mapping is determined by looking for the shortest period of the coefficient list.

The memory requirements of Algorithm 2.2 are linear in the input length, and the runtime requirements are slightly more than linear in the input length.

Example 2.3. Assume that we start with the coefficient list $l = ((-1, 0, -2), (30, 10, 20), (4, 0, 8), (-9, -1, -6))$. Then after Step (1.a) we have $l = ((-1, 0, -2), (3, 1, 2), (1, 0, 2), (-3, -1, -2))$, and after Step (1.b) we have $l = ((1, 0, 2), (3, 1, 2), (1, 0, 2), (3, 1, 2))$.

The list l has length $m = 4$, and the only prime divisor of 4 is $p_1 = 2$. In Step (3) we find that dividing our list l into 2 parts of the same length yields two equal lists $((1, 0, 2), (3, 1, 2))$. Thus we put $l := ((1, 0, 2), (3, 1, 2))$.

The length of l is still divisible by 2, therefore we repeat Step (3). However splitting our list into two parts yields two unequal lists $((1, 0, 2))$ and $((3, 1, 2))$ of length 1, thus no further reduction is possible.

There are no further prime divisors $p_2, p_3, \ldots$, hence we are finished and the result is the rcwa mapping with coefficients $((1, 0, 2), (3, 1, 2))$ (which happens to be the Collatz mapping).

We need a few basic terms describing the coefficients of an rcwa mapping:

Definition 2.4. Let f be an rcwa mapping, and assume that f is described by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m)\in\mathbb{Z}/m\mathbb{Z}}$. We define the *multiplier* $\mathrm{Mult}(f)$ of f by $\mathrm{lcm}_{r(m)\in\mathbb{Z}/m\mathbb{Z}}\ a_{r(m)}$, and the *divisor* $\mathrm{Div}(f)$ of f by $\mathrm{lcm}_{r(m)\in\mathbb{Z}/m\mathbb{Z}}\ c_{r(m)}$. We call f *integral* if $\mathrm{Div}(f) = 1$. We call f *class-wise order-preserving* if all $a_{r(m)}$ are positive.

For example, the Collatz mapping T has multiplier 3 and divisor 2, and it is class-wise order-preserving. The mappings $n \mapsto n + 1$ and $n \mapsto -n$ are both integral, but only the first of them is class-wise order-preserving. The multiplier and the divisor of a class transposition $\tau_{r_1(m_1),r_2(m_2)}$ are both equal to $\mathrm{lcm}(m_1, m_2)/\gcd(m_1, m_2)$. Class transpositions are therefore integral if and only if the moduli of the transposed residue classes are the same. They are always class-wise order-preserving.

The following assertion on multipliers and divisors of rcwa mappings is immediate:

Lemma 2.5. *The multiplier respectively divisor of a product of rcwa mappings divides the product of the multipliers respectively divisors of the factors. The inversion of an rcwa permutation interchanges multiplier and divisor.*

The class-wise order-preserving elements of $\mathrm{RCWA}(\mathbb{Z})$ form a subgroup:

Definition 2.6. Let $\mathrm{RCWA}^+(\mathbb{Z}) < \mathrm{RCWA}(\mathbb{Z})$ denote the subgroup which consists of all class-wise order-preserving elements.

The following lemma is needed for multiplying rcwa mappings:

Lemma 2.7. *Let f and g be rcwa mappings. Then the following hold:*

(1) $\mathrm{Mod}(f \cdot g) | \operatorname{lcm}(\mathrm{Mod}(f), \mathrm{Div}(f) \cdot \mathrm{Mod}(g))$.
(2) $\mathrm{Mod}(f \cdot g) | \mathrm{Mod}(f) \cdot \mathrm{Mod}(g)$.

Proof. Let $n \in \mathbb{Z}$. By definition, $n \bmod \mathrm{Mod}(f)$ determines the affine partial mapping of f which is applied to n. The image of n under this mapping is determined (mod $\mathrm{Mod}(g)$) by $n \bmod (\mathrm{Div}(f) \cdot \mathrm{Mod}(g))$. This yields Assertion (1). Assertion (2) follows from Assertion (1), since $\mathrm{Div}(f) | \mathrm{Mod}(f)$ (cf. Remark 1.2). □

Algorithm 2.8 (Multiplication of rcwa Mappings)**.** Let f and g be rcwa mappings. Further let m_f and m_g denote the moduli of f and g, respectively. Assume that f and g are represented by the reduced coefficient lists $(a_{f,r(m_f)}, b_{f,r(m_f)}, c_{f,r(m_f)})_{r(m_f) \in \mathbb{Z}/m_f\mathbb{Z}}$ and $(a_{g,r(m_g)}, b_{g,r(m_g)}, c_{g,r(m_g)})_{r(m_g) \in \mathbb{Z}/m_g\mathbb{Z}}$, respectively. We compute the product of f and g as follows:

(1) Put $m_{fg} := \gcd(m_f \cdot m_g, \operatorname{lcm}(m_f, \mathrm{Div}(f) \cdot m_g))$.
(2) For $r = 0, \ldots, m_{fg} - 1$, put
- $a_{fg,r(m_{fg})} := a_{f,r(m_f)} \cdot a_{g,f(r)(m_g)}$,
- $b_{fg,r(m_{fg})} := a_{g,f(r)(m_g)} \cdot b_{f,r(m_f)} + b_{g,f(r)(m_g)} \cdot c_{f,r(m_f)}$, and
- $c_{fg,r(m_{fg})} := c_{f,r(m_f)} \cdot c_{g,f(r)(m_g)}$.

(3) Apply Algorithm 2.2 to the coefficient list

$$(a_{fg,r(m_{fg})}, b_{fg,r(m_{fg})}, c_{fg,r(m_{fg})})_{r(m_{fg}) \in \mathbb{Z}/m_{fg}\mathbb{Z}},$$

and return the result.

In Step (1) of Algorithm 2.8, a multiplicative upper bound for the modulus is chosen. The choice is made according to Lemma 2.7. The actual multiplications of the affine partial mappings are performed in Step (2). In the last step the modulus of the result is determined, and the coefficient list is brought into normal form.

The memory requirements of Algorithm 2.8 are linear in the multiple m_{fg} of the modulus of the product computed in Step (1), and the runtime is slightly more than linear in m_{fg}.

Example 2.9. Assume that we want to compute the square T^2 of the Collatz mapping. Then, Step (1) yields $m_{fg} := \gcd(2 \cdot 2, \operatorname{lcm}(2, 2 \cdot 2)) = 4$. The coefficient lists of both factors are equal to $((1, 0, 2), (3, 1, 2))$. In Step (2), we compute from this the coefficient list

$$\begin{aligned} l \;:=\; & ((1 \cdot 1, 1 \cdot 0 + 0 \cdot 2, 2 \cdot 2), (3 \cdot 1, 1 \cdot 1 + 0 \cdot 2, 2 \cdot 2), \\ & (1 \cdot 3, 1 \cdot 0 + 1 \cdot 2, 2 \cdot 2), (3 \cdot 3, 3 \cdot 1 + 1 \cdot 2, 2 \cdot 2)) \\ =\; & ((1, 0, 4), (3, 1, 4), (3, 2, 4), (9, 5, 4)). \end{aligned}$$

Step (3) does not yield any reduction in this example, thus the result is the rcwa mapping

$$T^2 : \mathbb{Z} \to \mathbb{Z}, \quad n \mapsto \begin{cases} n/4 & \text{if } n \in 0(4), \\ (3n+1)/4 & \text{if } n \in 1(4), \\ (3n+2)/4 & \text{if } n \in 2(4), \\ (9n+5)/4 & \text{if } n \in 3(4) \end{cases}$$

with coefficient list l.

We describe an algorithm to check whether a given rcwa mapping is injective or surjective, respectively:

Algorithm 2.10 (Test for Injectivity / Surjectivity)**.** Let f be an rcwa mapping, and let m denote its modulus. Assume that the mapping f is represented by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. Since the two algorithms to check whether f is injective respectively surjective are very similar, we describe them together:

(1) If we check for injectivity, then start with the following checks:
   (a) If $\mathrm{Mult}(f) = 0$, then return *false*.
   (b) Otherwise choose some bound $b \in \mathbb{N}$ and check whether the image of the set $\{-b, -b+1, \ldots, b\}$ under f has cardinality $2b+1$. If not, then return *false*.

(2) Put $\hat{m} := m \cdot \mathrm{Mult}(f) / \gcd_{r(m) \in \mathbb{Z}/m\mathbb{Z}} c_{r(m)}$, and set up a list $l = (l_0, \ldots, l_{\hat{m}-1})$ of $\hat{m}$ zeros.

(3) For all $r = 0, \ldots, m-1$ such that $a_{r(m)} \neq 0$, do the following:
   (a) Put $\tilde{m} := m \cdot a_{r(m)} / c_{r(m)}$ and $\tilde{r} := (a_{r(m)} \cdot r + b_{r(m)}) / c_{r(m)} \bmod \tilde{m}$.
   (b) For $\hat{r} \in \{\tilde{r}, \tilde{r} + \tilde{m}, \ldots, \tilde{r} + (\hat{m}/\tilde{m} - 1) \cdot \tilde{m} + \tilde{r}\}$, do the following: If we check for injectivity and it is $l_{\hat{r}} = 1$, then return *false*. Otherwise put $l_{\hat{r}} := 1$.

(4) If we check for injectivity, then return *true*. If we check for surjectivity, then return *true* if $l_0 = l_1 = \cdots = l_{\hat{m}-1} = 1$, and *false* otherwise.

Algorithm 2.10 basically computes the images of the residue classes (mod m) under f, and checks whether they overlap (test for injectivity) or, respectively, whether they entirely cover $\mathbb{Z}$ (test for surjectivity). The memory requirements are linear in the value $\hat{m}$ computed in Step (2), and the runtime requirements are less than linear in $m \cdot \hat{m}$.

Example 2.11. Assume that we would like to check whether the Collatz mapping T is injective. We have $\mathrm{Mult}(T) = 3 \neq 0$, thus T can still be injective. Now choose, say, $b := 2$ and compute $T(\{-2, -1, 0, 1, 2\}) = \{-1, 0, 1, 2\}$. This disproves injectivity.

Now assume that we would like to check whether T is surjective. In Step (2) we put $\hat{m} := (2 \cdot 3) / \gcd(2, 2) = 3$, and set up a list $l := (0, 0, 0)$. In Step (3), we do the following:

- For $r = 0$, we put $\tilde{m} := (2 \cdot 1)/2 = 1$ and $\tilde{r} := (1 \cdot 0 + 0)/2 \bmod \tilde{m} = 0$. Then for $\hat{r} \in \{0, 1, 2\}$, we put $l_{\hat{r}} := 1$.
- For $r = 1$, we put $\tilde{m} := (2 \cdot 3)/2 = 3$ and $\tilde{r} := (3 \cdot 1 + 1)/2 \bmod \tilde{m} = 2$. Then for $\hat{r} \in \{2\}$, we put $l_{\hat{r}} := 1$.

In Step (4) we check whether $l = (1, 1, 1)$. Since this is the case, T is surjective.

What we have done is basically to compute $T(0(2)) = \mathbb{Z}$ and $T(1(2)) = 2(3)$, and to check whether the two images together cover all residue classes (mod 3).

The following lemma is needed for computing inverses of rcwa permutations:

Lemma 2.12. *Let σ be an rcwa permutation, and let m denote its modulus. Assume that σ is described by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. Then the modulus of the inverse of σ divides $m \cdot \mathrm{Mult}(\sigma) / \gcd_{r(m) \in \mathbb{Z}/m\mathbb{Z}} c_{r(m)}$.*

Proof. Clearly, the modulus of σ^{-1} divides the least common multiple of the moduli of the images of the residue classes $r(m) \in \mathbb{Z}/m\mathbb{Z}$ under σ. The assertion follows from this, since the image of a residue class $r(m)$ under the corresponding affine partial mapping $f|_{r(m)} : n \mapsto (a_{r(m)} \cdot n + b_{r(m)})/c_{r(m)}$ is a residue class (mod $m \cdot a_{r(m)}/c_{r(m)}$). □

Algorithm 2.13 (Inversion of rcwa Permutations)**.** Let σ be an rcwa permutation, and let m_σ denote its modulus. Assume that σ is represented by the reduced coefficient list $(a_{\sigma,r(m_\sigma)}, b_{\sigma,r(m_\sigma)}, c_{\sigma,r(m_\sigma)})_{r(m_\sigma)\in\mathbb{Z}/m_\sigma\mathbb{Z}}$. We compute the inverse of σ as follows:

(1) Put $m_{\sigma^{-1}} := m_\sigma \cdot \mathrm{Mult}(\sigma)/\gcd_{r(m_\sigma)\in\mathbb{Z}/m_\sigma\mathbb{Z}} c_{r(m_\sigma)}$.
(2) For $r = 0, \ldots, m_\sigma - 1$, do the following:
   (a) Put $a := c_{\sigma,r(m_\sigma)}$, $b := -b_{\sigma,r(m_\sigma)}$ and $c := a_{\sigma,r(m_\sigma)}$.
   (b) Put $\tilde{m} := m_\sigma \cdot a_{\sigma,r(m_\sigma)}/c_{\sigma,r(m_\sigma)}$.
   (c) Put $\tilde{r} := (a_{\sigma,r(m_\sigma)} \cdot r + b_{\sigma,r(m_\sigma)})/c_{\sigma,r(m_\sigma)} \bmod \tilde{m}$.
   (d) For $\hat{r} = \tilde{r}, \tilde{r}+\tilde{m}, \tilde{r}+2\tilde{m}, \ldots, \tilde{r}+(m_{\sigma^{-1}}/\tilde{m}-1)\cdot\tilde{m}$, put $a_{\sigma^{-1},\hat{r}(m_{\sigma^{-1}})} := a$, $b_{\sigma^{-1},\hat{r}(m_{\sigma^{-1}})} := b$, and $c_{\sigma^{-1},\hat{r}(m_{\sigma^{-1}})} := c$.
(3) Apply Algorithm 2.2 to the coefficient list

$$(a_{\sigma^{-1},\hat{r}(m_{\sigma^{-1}})}, b_{\sigma^{-1},\hat{r}(m_{\sigma^{-1}})}, c_{\sigma^{-1},\hat{r}(m_{\sigma^{-1}})})_{\hat{r}(m_{\sigma^{-1}})\in\mathbb{Z}/m_{\sigma^{-1}}\mathbb{Z}},$$

and return the result.

The memory requirements of Algorithm 2.13 are linear in the value $m_{\sigma^{-1}}$ computed in Step (1), and the runtime requirements are slightly more than linear in $m_{\sigma^{-1}}$.

Example 2.14. We invert Collatz' permutation

$$\alpha \in \mathrm{RCWA}(\mathbb{Z}) : \quad n \longmapsto \begin{cases} 2n/3 & \text{if } n \in 0(3), \\ (4n-1)/3 & \text{if } n \in 1(3), \\ (4n+1)/3 & \text{if } n \in 2(3), \end{cases}$$

which is represented by the coefficient list $((2,0,3),(4,-1,3),(4,1,3))$. In Step (1), $m_{\sigma^{-1}}$ is initialized with $(3\cdot 4)/\gcd(3,3,3) = 4$. In Step (2), we do the following:

- For $r = 0$, we put $a := 3$, $b := 0$ and $c := 2$. Then we put $\tilde{m} := (3\cdot 2)/3 = 2$ and $\tilde{r} := (2\cdot 0 + 0)/3 = 0$. Finally we put $a_{\sigma^{-1},0(4)} := a$, $a_{\sigma^{-1},2(4)} := a$, $b_{\sigma^{-1},0(4)} := b$, $b_{\sigma^{-1},2(4)} := b$, $c_{\sigma^{-1},0(4)} := c$ and $c_{\sigma^{-1},2(4)} := c$.
- For $r = 1$, we put $a := 3$, $b := 1$ and $c := 4$. Then we put $\tilde{m} := (3\cdot 4)/3 = 4$ and $\tilde{r} := (4\cdot 1 - 1)/3 = 1$. Finally we put $a_{\sigma^{-1},1(4)} := a$, $b_{\sigma^{-1},1(4)} := b$ and $c_{\sigma^{-1},1(4)} := c$.
- For $r = 2$, we put $a := 3$, $b := -1$ and $c := 4$. Then we put $\tilde{m} := (3\cdot 4)/3 = 4$ and $\tilde{r} := (4\cdot 2 + 1)/3 = 3$. Finally we put $a_{\sigma^{-1},3(4)} := a$, $b_{\sigma^{-1},3(4)} := b$ and $c_{\sigma^{-1},3(4)} := c$.

This yields the coefficient list $((3,0,2),(3,1,4),(3,0,2),(3,-1,4))$. In this example, Step (3) does not yield any further reduction, hence we obtain

$$\alpha^{-1} \in \mathrm{RCWA}(\mathbb{Z}) : \quad n \longmapsto \begin{cases} 3n/2 & \text{if } n \in 0(2), \\ (3n+1)/4 & \text{if } n \in 1(4), \\ (3n-1)/4 & \text{if } n \in 3(4). \end{cases}$$

We fix a representation of rcwa groups in memory:

Definition 2.15 (Internal Representation of rcwa Groups)**.** We store a finitely generated rcwa group as a list of generating rcwa permutations.

3. Set-Theoretic Unions of Residue Classes

Since set-theoretic unions of residue classes are objects which are commonly needed in computations with rcwa groups, we describe how to compute with them. The algorithms may look more or less straightforward, but since they are crucial for computing in rcwa groups, the author thinks that they are still worth being described.

Obviously, any union of finitely many residue classes of $\mathbb{Z}$ can be written as a union of residue classes modulo a common modulus.

Definition 3.1. Let $S \subseteq \mathbb{Z}$ be a union of finitely many residue classes. We define the *modulus* $\mathrm{Mod}(S)$ of S by the least positive integer m such that S can be written as a union of residue classes (mod m).

We need to implement algorithms for computing with elements of the closure of the class of unions of finitely many residue classes of $\mathbb{Z}$ with respect to addition and subtraction of finite sets of integers. This class of sets is closed under taking images and preimages under rcwa mappings. We represent these sets in memory in a straightforward way:

Definition 3.2 (Internal Representation of Residue Class Unions)**.** Let $S_{\mathrm{rc}} \subseteq \mathbb{Z}$ be a union of finitely many residue classes, and let S_{exc} and S_{inc} be finite subsets of S_{rc} and the complement of S_{rc} in $\mathbb{Z}$, respectively. Let m denote the modulus of S_{rc}, and assume that $S_{\mathrm{rc}} = r_1(m) \cup \cdots \cup r_l(m)$.

We represent the set $S := (S_{\mathrm{rc}} \cup S_{\mathrm{inc}}) \setminus S_{\mathrm{exc}}$ in memory as a record which contains the *modulus* m, the set $R = \{r_1, \ldots, r_l\}$ of residues and the sets S_{inc} and S_{exc} of included and excluded elements.

In order to simplify notation, we also call m the modulus of S. If S degenerates to a finite set, then for consistency we set the modulus equal to 0.

To distinguish them from unions of residue classes without attached sets of 'included' and 'excluded' integers, we use for sets like S the technical term *residue class union*.

The chosen representation certainly has the drawback that it is sometimes not the most economic one in terms of memory requirements, but it has the important feature that it is unique and that it therefore allows quick comparisons.

The membership test for such residue class unions is cheap as well: A given integer lies in a given residue class union if and only if it does not lie in the stored set of excluded integers, but it does lie either in the stored set of included integers, or its residue modulo the stored modulus lies in the stored list of residues.

Another very basic task is to compute unions, intersections and differences:

Algorithm 3.3 (Union, Intersection and Difference of Residue Class Unions)**.** Let $S_1 = (S_{1,\mathrm{rc}} \cup S_{1,\mathrm{inc}}) \setminus S_{1,\mathrm{exc}}$ and $S_2 = (S_{2,\mathrm{rc}} \cup S_{2,\mathrm{inc}}) \setminus S_{2,\mathrm{exc}}$ be residue class unions in the sense of Definition 3.2. Further let m_1 and m_2 denote the moduli of $S_{1,\mathrm{rc}}$ and $S_{2,\mathrm{rc}}$, respectively, and let R_1 and R_2 denote the corresponding sets of residues. We compute the set-theoretic union, intersection and difference of S_1 and S_2 as follows:

(1) Put $m := \mathrm{lcm}(m_1, m_2)$.
(2) This step depends on whether we want to compute the union, the intersection or the difference of S_1 and S_2. Either put
   - $R := \{0 \leqslant r < m \mid r \bmod m_1 \in R_1 \vee r \bmod m_2 \in R_2\}$ (union),
   - $R := \{0 \leqslant r < m \mid r \bmod m_1 \in R_1 \wedge r \bmod m_2 \in R_2\}$ (intersection), or
   - $R := \{0 \leqslant r < m \mid r \bmod m_1 \in R_1 \wedge r \bmod m_2 \notin R_2\}$ (difference).
(3) Let $p_1, \ldots, p_l$ be the prime divisors of m, and put $k := 1$.
(4) Put $\tilde{m} := m/p_k$ and $\tilde{R} := \{r \bmod \tilde{m} \mid r \in R\}$.

(5) If $|\tilde{R}| = |R|/p_k$, then put $m := \tilde{m}$ and $R := \tilde{R}$. Otherwise put $k := k + 1$.
(6) If $p_k \nmid m$, then put $k := k + 1$.
(7) If $k \leqslant l$, then go to Step (4).
(8) This step depends again on whether we want to compute the union, the intersection or the difference of S_1 and S_2.
   - For the union, proceed as follows:
      (a) Put $S_{\mathrm{inc}} := S_{1,\mathrm{inc}} \cup S_{2,\mathrm{inc}}$, and remove all integers n from S_{inc} for which $n \bmod m \in R$.
      (b) Put $S_{\mathrm{exc}} := \cup_{i=1}^{2} \{n \in S_{i,\mathrm{exc}} \mid n \bmod m_{3-i} \notin R_{3-i}\}$, and remove all integers n from S_{exc} for which $n \bmod m \notin R$.
   - For the intersection, proceed as follows:
      (a) Put $S_{\mathrm{inc}} := (\cup_{i=1}^{2} \{n \in S_{i,\mathrm{inc}} \mid n \bmod m_{3-i} \in R_{3-i}\}) \cup (S_{1,\mathrm{inc}} \cap S_{2,\mathrm{inc}})$, and remove all n from S_{inc} for which $n \bmod m \in R$.
      (b) Put $S_{\mathrm{exc}} := S_{1,\mathrm{exc}} \cup S_{2,\mathrm{exc}}$, and remove all integers n from S_{exc} for which $n \bmod m \notin R$.
   - For the difference, proceed as follows:
      (a) Put $S_{\mathrm{inc}} := \{n \in S_{1,\mathrm{inc}} \mid n \bmod m_2 \notin R_2\} \cup \{n \in S_{2,\mathrm{exc}} \mid n \bmod m_1 \in R_1\}$, and remove all integers n from S_{inc} for which $n \bmod m \in R$.
      (b) Put $S_{\mathrm{exc}} := S_{1,\mathrm{exc}} \cup S_{2,\mathrm{inc}}$, and remove all integers n from S_{exc} for which $n \bmod m \notin R$.
(9) Return the residue class union with the modulus m, the set of residues R and the finite sets S_{inc} and S_{exc} of included and excluded integers.

The actual computation of the union, intersection or difference of the unions of residue classes takes place in Step (1)-(2) of Algorithm 3.3, while the purpose of Step (3)-(7) is the determination of the modulus of the result. The finite sets of included and excluded integers are dealt with in Step (8).

The runtime- and memory requirements of Algorithm 3.3 are both at most linear in the lcm of the moduli of the arguments. If the lists of residues of the arguments are sparse, then the runtime- and memory requirements are lower provided that Step (2) is implemented in a suitable way (e.g. for the intersection, in this case it is worthwhile using the *Chinese Remainder Theorem*, etc.). Here and in the following, for simplicity we disregard the time needed to process the finite sets S_{inc} and S_{exc}.

We give an algorithm to check whether one residue class union is a subset of another:

Algorithm 3.4 (Checking for a Subset Relation between Residue Class Unions)**.** Let $S_1 = (S_{1,\mathrm{rc}} \cup S_{1,\mathrm{inc}}) \setminus S_{1,\mathrm{exc}}$ and $S_2 = (S_{2,\mathrm{rc}} \cup S_{2,\mathrm{inc}}) \setminus S_{2,\mathrm{exc}}$ be residue class unions in the sense of Definition 3.2. Further let m_1 and m_2 denote the moduli of $S_{1,\mathrm{rc}}$ and $S_{2,\mathrm{rc}}$, respectively, and let R_1 and R_2 denote the corresponding sets of residues. To determine whether S_2 is a subset of S_1, we proceed as follows:

(1) Put $m := \mathrm{lcm}(m_1, m_2)$.
(2) If not all integers in $S_{2,\mathrm{inc}}$ lie in S_1 as well, then return *false*.
(3) If some integer in $S_{1,\mathrm{exc}}$ lies in S_2 as well, then return *false*.
(4) For $i = 1, 2$, put $\tilde{R}_i := \{0 \leqslant r < m \mid r \bmod m_i \in R_i\}$.
(5) If $\tilde{R}_2$ is a subset of $\tilde{R}_1$, then return *true*, otherwise return *false*.

Sometimes we will add a constant to the elements of a residue class union, or we will multiply or divide them by some constant. These arithmetic operations are then applied elementwise to the stored set of residues and to the finite sets of included and excluded elements. For example we have $(1(4) \cup \{4, 8\}) \setminus \{-3, 9\} + 2 = (3(4) \cup \{6, 10\}) \setminus \{-1, 11\}$

and $3 \cdot (0(4) \cup 1(4)) = 0(12) \cup 3(12)$. Adding a constant leaves the modulus unchanged, while multiplications and divisions are applied to the modulus as well. Of course, dividing a residue class union by a constant is possible only if all of its elements are divisible by that constant. Multiplying a residue class union by 0 yields the set $\{0\}$.

A common task is to refine a partition of a residue class union into residue classes to some prescribed length:

Algorithm 3.5 (Refine a Partition into Residue Classes to Prescribed Length)**.** Let $S \subseteq \mathbb{Z}$ be a nonempty union of finitely many residue classes, let $\mathcal{P}$ be a partition of S into residue classes, and let $l > |\mathcal{P}|$ be a positive integer. We refine $\mathcal{P}$ to a partition of length l as follows:

(1) Let $r(m)$ be a residue class in $\mathcal{P}$ with smallest modulus.
(2) Replace $r(m)$ by the two residue classes $r(2m)$ and $r + m(2m)$.
(3) If $|\mathcal{P}| < l$ then go to Step (1).
(4) Return $\mathcal{P}$.

Often it is helpful to write a union of residue classes as a disjoint union of a 'small' number of residue classes. For example, the set $1(12) \cup 2(12) \cup 3(12) \cup 4(12) \cup 5(12) \cup 6(12) \cup 7(12) \cup 9(12) \cup 10(12) \cup 11(12)$ has the partition $\{1(2), 2(4), 4(12)\}$. The following algorithm is reasonably fast and usually produces partitions of either minimal or close-to-minimal length:

Algorithm 3.6 (Short Partitions of Unions of Residue Classes)**.** Let $S \subseteq \mathbb{Z}$ be a union of finitely many residue classes, let m denote its modulus and let $R = \{r_1, \ldots, r_l\}$ denote the corresponding set of residues. We determine a partition of S into 'few' residue classes as follows:

(1) If $|R| = 1$, then return $\{r_1(m)\}$.
(2) Let $d_1, \ldots, d_{\tau(m)}$ be the divisors of m, sorted in ascending order, and put $\mathcal{P} := \emptyset$.
(3) For $d = d_1, \ldots, d_{\tau(m)}$, do the following:
  (a) If $d \nmid m$ or $|R| < m/d$, then proceed immediately to the next divisor d.
  (b) For $r \in \{\tilde{r} \bmod d \mid \tilde{r} \in R\}$, do the following:
    (i) Check whether $\{r, r + d, \ldots, r + (m/d - 1) \cdot d\} \subseteq R$. If not, then proceed immediately to the next residue r.
    (ii) Put $\mathcal{P} := \mathcal{P} \cup \{r(d)\}$ and $S := S \setminus r(d)$.
    (iii) Put $m := \mathrm{Mod}(S)$, and let R be the set of residues of S.
    (iv) If $d \nmid m$ or $|R| < m/d$, then break the inner loop and proceed immediately to the next divisor d.
  (c) If $S = \emptyset$, then break the loop and go to Step (4).
(4) Return $\mathcal{P}$.

The memory requirements of Algorithm 3.6 are linear in m, and the runtime requirements are between linear and quadratic in m.

Example 3.7. Let S be the union of the residue classes $r(60)$ for $r \in R$, where

$$\begin{aligned} R \;=\; \{&0, 1, 3, 4, 5, 7, 8, 12, 13, 16, 17, 19, 20, 24, 25, 28, 29, \\ &31, 32, 33, 36, 37, 40, 41, 43, 44, 48, 49, 52, 53, 55, 56\}. \end{aligned}$$

We decompose the set S into few disjoint residue classes as follows:

- We determine the divisors of 60 – these are 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 and 60 – and initialize $\mathcal{P}$ by $\emptyset$.
- $d = 1$: We observe that $|R| = 32 < 60/1$, thus proceed to the next d.

- $d = 2$: We observe that neither $\{0, 2, 4, 6, 8, \ldots, 58\}$ nor $\{1, 3, 5, 7, 9, \ldots, 59\}$ are subsets of R, thus proceed to the next d.
- $d = 3$: We observe that neither $\{0, 3, \ldots, 57\}$ nor $\{1, 4, \ldots, 58\}$ or $\{2, 5, \ldots, 59\}$ are subsets of R, thus proceed to the next d.
- $d = 4$: We observe that $\{0, 4, \ldots, 56\} \subset R$, put $\mathcal{P} := \mathcal{P} \cup \{0(4)\}$ and $S := S \setminus 0(4)$, and let $R := \{1, 3, 5, 7, 13, 17, 19, 25, 29, 31, 33, 37, 41, 43, 49, 53, 55\}$ be the set of residues of S.
- $d = 6$: We observe that $\{1, 7, \ldots, 55\} \subset R$, put $\mathcal{P} := \mathcal{P} \cup \{1(6)\}$ and $S := S \setminus 1(6)$, and let $R := \{3, 5, 17, 29, 33, 41, 53\}$ be the set of residues of S.
- $d = 10$: We observe that none of the sets $\{r, r + 10, \ldots, r + 50\}, r = 0, \ldots, 9$ is a subset of R, thus proceed to the next d.
- $d = 12$: We observe that $\{5, 17, 29, 41, 53\} \subset R$, put $\mathcal{P} := \mathcal{P} \cup \{5(12)\}$ and $S := S \setminus 5(12)$, and let $R := \{3, 33\}$ be the set of residues of S.
- $d = 15$: We observe that $|R| = 2 < 60/15$, thus proceed to the next d.
- $d = 20$: We observe that $|R| = 2 < 60/20$, thus proceed to the next d.
- $d = 30$: We observe that $\{3, 33\} \subset R$, and put $\mathcal{P} := \mathcal{P} \cup \{3(30)\}$ as well as $S := S \setminus 3(30) = \emptyset$.
- We have $\mathcal{P} = \{0(4), 1(6), 5(12), 3(30)\}$, thus $S = 0(4) \cup 1(6) \cup 5(12) \cup 3(30)$.

4. Computing Images and Preimages

In this section, we describe how to compute images and preimages of residue class unions under rcwa mappings, how to compute the support (i.e. the set of moved points) of an rcwa permutation and how to compute the restriction of an rcwa permutation to a union of residue classes. We also describe how to determine an element of $\mathrm{CT}(\mathbb{Z})$ which maps a given union of residue classes to a given other union of residue classes.

Algorithm 4.1 (Images of Residue Class Unions under rcwa Mappings)**.** Let f be an rcwa mapping, and let m denote its modulus. Assume that the mapping f is represented by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. Further let $S \subseteq \mathbb{Z}$ be a residue class union, and let $\tilde{m}$ denote its modulus. We compute the image of S under f as follows:

(1) Put $\tilde{S} := \emptyset$.
(2) For $r = 0, \ldots, m - 1$, do the following:
  (a) By Algorithm 3.3, compute $S_{r(m)} := S \cap r(m)$.
  (b) Compute $\tilde{S}_{r(m)} := f(S_{r(m)}) = (a_{r(m)} \cdot S_{r(m)} + b_{r(m)})/c_{r(m)}$.
  (c) By Algorithm 3.3, compute $\tilde{S} := \tilde{S} \cup \tilde{S}_{r(m)}$.
(3) Return $\tilde{S}$.

The runtime- and memory requirements of Algorithm 4.1 are at most linear in the product of the moduli of f and S, but can be considerably lower if the list of residues of S is reasonably sparse.

Example 4.2. We would like to determine the image of the residue class 0(5) under the Collatz mapping T: In the notation used in Algorithm 4.1, we have $S = 0(5)$, $S_{0(2)} = S \cap 0(2) = 0(10)$ and $S_{1(2)} = S \cap 1(2) = 5(10)$. Hence it is $T(S_{0(2)}) = 0(10)/2 = 0(5)$ and $T(S_{1(2)}) = (3 \cdot 5(10) + 1)/2 = 8(15)$, thus $T(S) = T(S_{0(2)}) \cup T(S_{1(2)}) = 0(5) \cup 8(15)$.

Algorithm 4.3 (Preimages of Integers under rcwa Mappings)**.** Let f be an rcwa mapping, and let m denote its modulus. Assume that the mapping f is represented by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. We compute the preimage of an integer n under f as follows:

(1) Put $R := \emptyset$ and $S_{\mathrm{inc}} := \emptyset$.
(2) For $r = 0, \ldots, m-1$, do the following:
  (a) If $a_{r(m)} = 0$, then proceed as follows:
    (i) If $b_{r(m)} = n$, then add r to the set R of residues.
    (ii) If $b_{r(m)} \neq n$, then proceed to the next r.
  (b) If $a_{r(m)} \neq 0$, then proceed as follows:
    (i) Put $\tilde{n} := (c_{r(m)} \cdot n - b_{r(m)})/a_{r(m)}$.
    (ii) If $\tilde{n}$ is an integer, and if further $\tilde{n} \bmod m = r$, then add $\tilde{n}$ to S_{inc}.
(3) Remove the elements from S_{inc} which are congruent (mod m) to one of the residues in R. Then return the residue class union with the modulus m, the set of residues R and the sets S_{inc} and $\emptyset$ of included and excluded integers.

Algorithm 4.4 (Preimages of Residue Class Unions under rcwa Mappings)**.** Let f be an rcwa mapping, and let m denote its modulus. Assume that f is represented by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. Further let $S = (S_{\mathrm{rc}} \cup S_{\mathrm{inc}}) \setminus S_{\mathrm{exc}} \subseteq \mathbb{Z}$ be a residue class union, and let $\hat{m}$ denote its modulus. We compute the preimage of S under f as follows:

(1) Put $\tilde{m} := m \cdot \hat{m} \cdot \mathrm{Div}(f)$.
(2) Put $R := \{0 \leqslant r < \tilde{m} \mid f(r) \in S_{\mathrm{rc}}\}$.
(3) Put $\tilde{S} := \cup_{r \in R}\, r(\tilde{m})$.
(4) By Algorithm 4.3, compute the preimages of the integers in S_{inc} under f, and add them to $\tilde{S}$ by Algorithm 3.3.
(5) For $n \in S_{\mathrm{exc}}$, do the following:
  (a) By Algorithm 4.3, compute the preimage $\tilde{S}_{n,1}$ of n under f.
  (b) By Algorithm 4.1, compute the image S_n of $\tilde{S}_{n,1}$ under f.
  (c) By Algorithm 4.3 and Algorithm 3.3, compute the preimage $\tilde{S}_{n,2}$ of the finite set $S_n \setminus S_{\mathrm{exc}}$ under f.
  (d) By Algorithm 3.3, compute $D := \tilde{S}_{n,1} \setminus \tilde{S}_{n,2}$.
  (e) If $D \neq \emptyset$, then put $\tilde{S} := \tilde{S} \setminus D$. For this, use Algorithm 3.3.
(6) Return $\tilde{S}$.

The runtime- and memory requirements of Algorithm 4.4 are linear in the product of the modulus of f, the divisor of f and the modulus of S.

Example 4.5. We would like to determine the preimage of the residue class 0(5) under the Collatz mapping T: In Step (1), we compute $\tilde{m} := \mathrm{Mod}(T) \cdot \mathrm{Mod}(0(5)) \cdot \mathrm{Div}(T) = 2 \cdot 5 \cdot 2 = 20$. In Step (2), we compute $R := \{0 \leqslant r < 20 \mid T(r) \in 0(5)\} = \{0, 3, 10, 13\}$. In Step (3), we put $\tilde{S} := \cup_{r \in R}\, r(20) = 0(10) \cup 3(10)$. Since $S_{\mathrm{inc}} = S_{\mathrm{exc}} = \emptyset$, this is already our result.

The support of an rcwa permutation can always be represented as a residue class union:

Algorithm 4.6 (Support of an rcwa Permutation)**.** Let σ be an rcwa permutation, and let m denote its modulus. Assume that the permutation σ is represented by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. We compute the support of σ as follows:

(1) Let S be the union of the residue classes (mod m), for which the coefficient triple $(a_{r(m)}, b_{r(m)}, c_{r(m)})$ is not equal to $(1, 0, 1)$.
(2) For $r = 0, \ldots, m-1$, do the following:
  (a) If $a_{r(m)} = c_{r(m)} = 1$, then proceed to the next r.
  (b) Put $n := b_{r(m)}/(c_{r(m)} - a_{r(m)})$.

(c) If n is an integer, and if further $n \bmod m = r$, then put $S := S \setminus \{n\}$.
(3) Return S.

Step (2) of Algorithm 4.6 takes care of the fixed points of the affine partial mappings. The support of an rcwa group is determined by just computing the union of the supports of its generators. Next we describe how to restrict an rcwa permutation to a union of finitely many residue classes which it maps to itself:

Algorithm 4.7 (Restricting an rcwa Permutation to a Union of Residue Classes)**.** Let σ be an rcwa permutation, and let m_1 denote its modulus. Assume that the permutation σ is represented by the reduced coefficient list $(a_{r(m_1)}, b_{r(m_1)}, c_{r(m_1)})_{r(m_1)\in\mathbb{Z}/m_1\mathbb{Z}}$. Further assume that S is a union of residue classes such that $\sigma(S) = S$, and let m_2 denote the modulus of S. We compute the restriction of the permutation σ to the set S as follows:

(1) Put $m := \mathrm{lcm}(m_1, m_2)$.
(2) For $r = 0, \ldots, m-1$, do the following:
- If $r \in S$, then put $\tilde{a}_{r(m)} := a_{r(m_1)}$, $\tilde{b}_{r(m)} := b_{r(m_1)}$, and $\tilde{c}_{r(m)} := c_{r(m_1)}$.
- If $r \notin S$, then put $\tilde{a}_{r(m)} := 1$, $\tilde{b}_{r(m)} := 0$, and $\tilde{c}_{r(m)} := 1$.
(3) Apply Algorithm 2.2 to the coefficient list $(\tilde{a}_{r(m)}, \tilde{b}_{r(m)}, \tilde{c}_{r(m)})_{r(m)\in\mathbb{Z}/m\mathbb{Z}}$, and return the result.

The runtime- and memory requirements of Algorithm 4.6 are both linear in m, and the ones of Algorithm 4.7 are both approximately linear in $\mathrm{lcm}(m_1, m_2)$.

The groups $\mathrm{RCWA}(\mathbb{Z})$ and $\mathrm{CT}(\mathbb{Z})$ act transitively on the set of nonempty unions of finitely many residue classes which are not equal to $\mathbb{Z}$ itself. The corresponding algorithms are as follows:

Algorithm 4.8 (Transitivity of $\mathrm{RCWA}(\mathbb{Z})$ on Unions of Residue Classes)**.** Let S_1 and S_2 be nonempty unions of residue classes of $\mathbb{Z}$. We compute an rcwa permutation σ which maps S_1 to S_2 as follows:

(1) By Algorithm 3.6, write the sets S_1 and S_2 as well as their complements in $\mathbb{Z}$ as unions of 'few' residue classes. Let $\mathcal{S}_1, \mathcal{S}_2, \mathcal{C}_1$ and $\mathcal{C}_2$ be the resulting partitions.
(2) If the lengths of the partitions $\mathcal{S}_1$ and $\mathcal{S}_2$ differ, then refine the shorter one to the length of the longer one by Algorithm 3.5. In the same way, take care that the partitions $\mathcal{C}_1$ and $\mathcal{C}_2$ have the same length.
(3) Let $\mathcal{P}_i$, $i = 1, 2$ be the concatenation of $\mathcal{S}_i$ and $\mathcal{C}_i$. Obviously, the $\mathcal{P}_i$ are partitions of $\mathbb{Z}$ of the same length.
(4) By putting together affine mappings, construct an rcwa permutation σ which maps $\mathcal{P}_1$ to $\mathcal{P}_2$ in such a way that the ith residue class in $\mathcal{P}_1$ is mapped to the ith residue class in $\mathcal{P}_2$.
(5) Return σ.

Example 4.9. We determine an rcwa permutation which maps $\mathbb{Z} \setminus 0(4)$ to $1(4)$:

- In Step (1), we obtain the partitions $\mathcal{S}_1 = \{1(2), 2(4)\}, \mathcal{S}_2 = \{1(4)\}, \mathcal{C}_1 = \{0(4)\}$ and $\mathcal{C}_2 = \{0(2), 3(4)\}$.
- In Step (2), we refine $\mathcal{S}_2$ to $\{1(8), 5(8)\}$ and $\mathcal{C}_1$ to $\{0(8), 4(8)\}$.
- In Step (3), we concatenate $\mathcal{S}_1$ and $\mathcal{C}_1$ to $\mathcal{P}_1 := \{1(2), 2(4), 0(8), 4(8)\}$, and $\mathcal{S}_2$ and $\mathcal{C}_2$ to $\mathcal{P}_2 := \{1(8), 5(8), 0(2), 3(4)\}$.

- In Step (4), we build our mapping

$$\sigma \in \mathrm{RCWA}(\mathbb{Z}): \quad n \longmapsto \begin{cases} 4n-3 & \text{if } n \in 1(2), \\ 2n+1 & \text{if } n \in 2(4), \\ n/4 & \text{if } n \in 0(8), \\ (n+2)/2 & \text{if } n \in 4(8). \end{cases}$$

Algorithm 4.10 (Transitivity of $\mathrm{CT}(\mathbb{Z})$ on Unions of Residue Classes)**.** Let $S_1, S_2 \subset \mathbb{Z}$ be distinct nonempty unions of residue classes. We determine an element $\sigma \in \mathrm{CT}(\mathbb{Z})$ which maps S_1 to S_2 as follows:

(1) If $S_1 \subset S_2$, then put $D_1 := S_2 \setminus S_1$. Otherwise put $D_1 := \mathbb{Z} \setminus S_1$.
(2) Put $D_2 := \mathbb{Z} \setminus (D_1 \cup S_2)$. Now we have a chain S_1, D_1, D_2, S_2 of 4 nonempty sets starting at S_1 and ending at S_2 in which any two consecutive sets are disjoint.
(3) By Algorithm 3.6, write the sets S_1, D_1, D_2 and S_2 as unions of 'few' residue classes. Let $\mathcal{P}_1, \ldots, \mathcal{P}_4$ be the resulting partitions into residue classes, and let $l_{\max}$ be the maximum of the lengths of these partitions.
(4) If some of the partitions $\mathcal{P}_i$ are shorter than $l_{\max}$, then refine them to length $l_{\max}$ by Algorithm 3.5.
(5) Assume that $\mathcal{P}_i = (r_{i,1}(m_{i,1}), \ldots, r_{i,l_{\max}}(m_{i,l_{\max}}))$, $i = 1, \ldots, 4$, and compute

$$\sigma := \prod_{i=1}^{3} \prod_{j=1}^{l_{\max}} \tau_{r_{i,j}(m_{i,j}), r_{i+1,j}(m_{i+1,j})}.$$

(6) Return σ.

Remark 4.11. Given disjoint residue classes $r_1(m_1)$ and $r_2(m_2)$, the corresponding class transposition is given by

$$\tau_{r_1(m_1), r_2(m_2)} \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} (m_2 n + m_1 r_2 - m_2 r_1)/m_1 & \text{if } n \in r_1(m_1), \\ (m_1 n + m_2 r_1 - m_1 r_2)/m_2 & \text{if } n \in r_2(m_2), \\ n & \text{otherwise.} \end{cases}$$

Example 4.12. We determine an element $\sigma \in \mathrm{CT}(\mathbb{Z})$ such that $\sigma(1(3)) = 2(4) \cup 3(4)$:

- In Step (1), we put $D_1 := \mathbb{Z} \setminus 1(3) = 0(3) \cup 2(3)$.
- In Step (2), we put $D_2 := \mathbb{Z} \setminus ((0(3) \cup 2(3)) \cup (2(4) \cup 3(4))) = 1(12) \cup 4(12)$.
- In Step (3), we find the (in this example obvious) partitions $\{1(3)\}$, $\{0(3), 2(3)\}$, $\{1(12), 4(12)\}$ and $\{2(4), 3(4)\}$.
- In Step (4), in order to obtain partitions of the same length, in our example we only need to refine $\{1(3)\}$ to $\{1(6), 4(6)\}$.
- In Step (5), we compute

$$\sigma := \tau_{0(3),1(6)} \cdot \tau_{2(3),4(6)} \cdot \tau_{0(3),1(12)} \cdot \tau_{2(3),4(12)} \cdot \tau_{2(4),1(12)} \cdot \tau_{3(4),4(12)}.$$

This yields

$$\sigma \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} (2n+4)/3 & \text{if } n \in 1(6), \\ 6n-2 & \text{if } n \in 3(6), \\ (2n+1)/3 & \text{if } n \in 4(6), \\ 6n-5 & \text{if } n \in 5(6), \\ n/2 & \text{if } n \in 0(24) \cup 18(24), \\ (3n-4)/2 & \text{if } n \in 2(24) \cup 20(24), \\ (3n-10)/2 & \text{if } n \in 6(24) \cup 12(24), \\ (n+2)/2 & \text{if } n \in 8(24) \cup 14(24). \end{cases}$$

5. Representing Groups as rcwa Groups

The purpose of this section is to describe how to represent a given group as an rcwa group. This is easiest for the finite groups:

Definition 5.1. Let $m \in \mathbb{N}$, and let S_m be the symmetric group of degree m. We define the monomorphism $\varphi_m : \mathrm{S}_m \hookrightarrow \mathrm{CT}(\mathbb{Z})$ by $\sigma \mapsto (\sigma^{\varphi_m} : n \mapsto n + \sigma(n \bmod m) - n \bmod m)$, where we assume that S_m acts naturally on the set $\{0, 1, \ldots, m-1\}$.

The group $\mathrm{RCWA}(\mathbb{Z})$ is not co-Hopfian. The following monomorphisms will play a crucial role in several algorithms given in this section:

Definition 5.2. Given an injective rcwa mapping f, let $\pi_f : \mathrm{RCWA}(\mathbb{Z}) \hookrightarrow \mathrm{RCWA}(\mathbb{Z})$, $\sigma \mapsto \sigma_f$ be the monomorphism defined by the properties $\forall \sigma \in \mathrm{RCWA}(\mathbb{Z})\ f\sigma_f = \sigma f$ and $\mathrm{supp}(\mathrm{im}\,\pi_f) \subseteq \mathrm{im}\, f$. Then we call π_f the *restriction monomorphism* associated with f.

Sometimes we also need the right inverses of restriction monomorphisms:

Definition 5.3. Let f be an injective rcwa mapping, and let π_f denote the restriction monomorphism associated with f. Then we call the right inverse $\tilde{\pi}_f : \mathrm{im}\,\pi_f \to \mathrm{RCWA}(\mathbb{Z})$ of π_f the *induction epimorphism* associated with f.

In order to compute images of rcwa permutations under restriction monomorphisms and induction epimorphisms, we need an algorithm to compute right inverses of injective rcwa mappings:

Algorithm 5.4 (Right Inverses of Injective rcwa Mappings)**.** Let f be an injective rcwa mapping, and let m denote its modulus. Assume that the mapping f is represented by the reduced coefficient list $(a_{f,r(m)}, b_{f,r(m)}, c_{f,r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$. We compute a right inverse of f, i.e. a mapping $\tilde{f}$ such that $f \cdot \tilde{f} = 1$, as follows:

(1) Put $\tilde{m} := \mathrm{lcm}_{r(m) \in \mathbb{Z}/m\mathbb{Z}}\ m \cdot a_{f,r(m)}/c_{f,r(m)}$.
(2) For all $\tilde{r}(\tilde{m}) \in \mathbb{Z}/\tilde{m}\mathbb{Z}$, put $a_{\tilde{f},\tilde{r}(\tilde{m})} := 1$, $b_{\tilde{f},\tilde{r}(\tilde{m})} := 0$ and $c_{\tilde{f},\tilde{r}(\tilde{m})} := 1$.
(3) For $r = 0, \ldots, m-1$, do the following:
  - For all $\tilde{r} \in \{0, \ldots, \tilde{m}-1\}$ which lie in the image of $r(m)$ under f, put $a_{\tilde{f},\tilde{r}(\tilde{m})} := c_{f,r(m)}$, $b_{\tilde{f},\tilde{r}(\tilde{m})} := -b_{f,r(m)}$ and $c_{\tilde{f},\tilde{r}(\tilde{m})} := a_{f,r(m)}$.
(4) Apply Algorithm 2.2 to the coefficient list $(a_{\tilde{f},\tilde{r}(\tilde{m})}, b_{\tilde{f},\tilde{r}(\tilde{m})}, c_{\tilde{f},\tilde{r}(\tilde{m})})_{\tilde{r}(\tilde{m}) \in \mathbb{Z}/\tilde{m}\mathbb{Z}}$, and return the result.

Algorithm 5.5 (Images of rcwa Permutations under Restriction Monomorphisms)**.** Let f be an injective rcwa mapping, and let σ be an rcwa permutation. We compute the image $\tilde{\sigma}$ of σ under the restriction monomorphism π_f associated with f as follows:

(1) By Algorithm 5.4, compute a right inverse $\tilde{f}$ of f.
(2) Put $\tilde{\sigma} := \tilde{f} \cdot \sigma \cdot f$.
(3) By Algorithm 4.1, compute the image of f.
(4) By Algorithm 4.7, compute the restriction $\tilde{\sigma}|_{\mathrm{im}\, f}$ of the permutation $\tilde{\sigma}$ to the image of f.
(5) Return $\tilde{\sigma}|_{\mathrm{im}\, f}$.

Be careful not to confuse computing the image of an rcwa permutation under a restriction monomorphism with restricting an rcwa permutation to a set.

Algorithm 5.6 (Images of rcwa Permutations under Induction Epimorphisms)**.** Let f be an injective rcwa mapping, and let σ be an rcwa permutation whose support is a subset of $\mathrm{im}\, f$. We compute the image of σ under the induction epimorphism $\tilde{\pi}_f$ as follows:

(1) By Algorithm 5.4, compute a right inverse $\tilde{f}$ of f.
(2) Return $\tilde{\sigma} := f \cdot \sigma \cdot \tilde{f}$.

In terms of runtime- and memory requirements, computing right inverses corresponds to computing inverses, and computing images under restriction monomorphisms and induction epimorphisms corresponds to computing conjugates.

Example 5.7. We compute the image of the class transposition $\tau : n \mapsto n + (-1)^n$ under the restriction monomorphism associated with $f : n \mapsto 2n$: A right inverse of f is

$$\tilde{f} : \; n \mapsto \begin{cases} n/2 & \text{if } n \in 0(2), \\ n & \text{if } n \in 1(2). \end{cases}$$

We compute

$$\tilde{f} \cdot \tau \cdot f : \; n \mapsto \begin{cases} 2n-2 & \text{if } n \in 1(2), \\ n+2 & \text{if } n \in 0(4), \\ n-2 & \text{if } n \in 2(4), \end{cases}$$

and restrict this mapping to the image of f to obtain

$$\pi_f(\tau) : \; n \mapsto \begin{cases} n & \text{if } n \in 1(2), \\ n+2 & \text{if } n \in 0(4), \\ n-2 & \text{if } n \in 2(4). \end{cases}$$

This is the class transposition $\tau_{0(4),2(4)}$.

Remark 5.8. We observe that the image of a class transposition $\tau_{r_1(m_1),r_2(m_2)}$ under a restriction monomorphism $\pi_{n \mapsto mn+r}$ is $\tau_{mr_1+r(mm_1),mr_2+r(mm_2)}$. Therefore restriction monomorphisms of the form $\pi_{n \mapsto mn+r}$ embed $\mathrm{CT}(\mathbb{Z})$ into itself.

Further we observe that any class transposition $\tau_{r_1(m_1),r_2(m_2)}$ can be obtained as an image of τ under a restriction monomorphism – we have $\tau_{r_1(m_1),r_2(m_2)} = \pi_\mu(\tau)$, where

$$\mu : \; n \longmapsto \begin{cases} (m_1 n + 2r_1)/2 & \text{if } n \in 0(2), \\ (m_2 n + (2r_2 - m_2))/2 & \text{if } n \in 1(2) \end{cases}$$

maps the residue classes 0(2) and 1(2) to $r_1(m_1)$ and $r_2(m_2)$, respectively.

It is obvious that the image of an rcwa group under a restriction monomorphism or under an induction epimorphism can be computed by applying Algorithm 5.5 respectively Algorithm 5.6 to all stored generators.

Constructing direct products of rcwa groups and wreath products of rcwa groups with finite groups is now more or less straightforward:

Algorithm 5.9 (Direct Products of rcwa Groups)**.** Let G and H be finitely generated rcwa groups. We compute an rcwa group isomorphic to the direct product of G and H as follows:

(1) By Algorithm 5.5, compute the image of G under the restriction monomorphism $\pi_{n\mapsto 2n}$ and the image of H under the restriction monomorphism $\pi_{n\mapsto 2n+1}$.
If G and H are subgroups of $\mathrm{CT}(\mathbb{Z})$, then by Remark 5.8 the computed images are still subgroups of $\mathrm{CT}(\mathbb{Z})$.
(2) Return the rcwa group whose list of stored generators is the concatenation of the lists of generators of the groups computed in Step (1).

Algorithm 5.10 (Wreath Products of rcwa Groups with Finite Groups)**.** Let G be a finitely generated rcwa group, and let $P < \mathrm{S}_m$ be a finite permutation group. We compute an rcwa group isomorphic to the natural wreath product of G with P as follows:

(1) Let $\tilde{P}$ be the image of P under the embedding φ_m.
(2) Determine a set $\{r_1(m), \ldots, r_k(m)\}$ of representatives for the orbits on $\mathbb{Z}/m\mathbb{Z}$ under the action of $\tilde{P}$.
(3) For $i = 1, \ldots, k$, use Algorithm 5.5 to compute the image G_i of G under the restriction monomorphism $\pi_{n\mapsto mn+r_i}$.
If G is a subgroup of $\mathrm{CT}(\mathbb{Z})$, then by Remark 5.8, the groups G_i are still subgroups of $\mathrm{CT}(\mathbb{Z})$.
(4) Return the rcwa group whose list of stored generators is the concatenation of the lists of generators of the groups $G_1, \ldots, G_k$ and $\tilde{P}$.

Example 5.11. We construct an rcwa group isomorphic to the natural wreath product $(\mathbb{Z}, +) \wr \mathrm{A}_5$ of the infinite cyclic group with the alternating group of degree 5. For this, let $G := \langle n \mapsto n + 1 \rangle$ and $P := \langle (1, 2, 3, 4, 5), (3, 4, 5) \rangle$. In Step (1), we compute the generators

$$n \mapsto \begin{cases} n+1 & \text{if } n \in \mathbb{Z} \setminus 4(5), \\ n-4 & \text{if } n \in 4(5) \end{cases} \quad \text{and} \quad n \mapsto \begin{cases} n & \text{if } n \in 0(5) \cup 1(5), \\ n+1 & \text{if } n \in 2(5) \cup 3(5), \\ n-2 & \text{if } n \in 4(5) \end{cases}$$

of $\tilde{P}$. The group $\tilde{P}$ acts transitively on the set $\{0(5), \ldots, 4(5)\}$, therefore in Step (2) we get $k = 1$ and $r_1(m) = 0(5)$. In Step (3), we compute $G_1 := \pi_{n\mapsto 5n}(G)$. The group G_1 is generated by

$$n \mapsto \begin{cases} n+5 & \text{if } n \in 0(5), \\ n & \text{if } n \in \mathbb{Z} \setminus 0(5). \end{cases}$$

Now the result is the group which is generated by $\tilde{P}$ and G_1.

It is already much less obvious how to construct wreath products of rcwa groups with the infinite cyclic group $(\mathbb{Z}, +)$:

Algorithm 5.12 (Restricted Wreath Products of rcwa Groups with $(\mathbb{Z}, +)$)**.** Let G be a finitely generated rcwa group. We compute an rcwa group isomorphic to the natural restricted wreath product of G with $(\mathbb{Z}, +)$ as follows:

(1) By Algorithm 5.5, compute the image $\tilde{G}$ of G under $\pi_{n\mapsto 4n+3}$.
If G is a subgroup of $\mathrm{CT}(\mathbb{Z})$ then, by Remark 5.8, the group $\tilde{G}$ is still a subgroup of $\mathrm{CT}(\mathbb{Z})$.
(2) Return the rcwa group whose list of stored generators is obtained by adding the permutation $\tau \cdot \tau_{0(2),1(4)}$ to the list of generators of $\tilde{G}$.

Algorithm 5.12 is valid since the images of the residue class 3(4) under the elements of the cyclic group generated by $\tau \cdot \tau_{0(2),1(4)}$ are pairwise disjoint residue classes.

Example 5.13. We construct an rcwa group isomorphic to the natural restricted wreath product $\mathrm{C}_2 \wr (\mathbb{Z}, +)$ of the cyclic group of order 2 with the infinite cyclic group $(\mathbb{Z}, +)$:

Put $G := \langle \tau \rangle$. In Step (1), we compute the image $\tilde{G}$ of G under the restriction monomorphism $\pi_{n\mapsto 4n+3}$ – we obtain $\tilde{G} = \langle \tau_{3(8),7(8)} \rangle$. In Step (2), we add $\tau \cdot \tau_{0(2),1(4)}$ to the set of generators of $\tilde{G}$, and obtain the result $\langle \tau \cdot \tau_{0(2),1(4)}, \tau_{3(8),7(8)} \rangle$.

Next we describe how to construct representations of free groups:

Algorithm 5.14 (Construction of Free Groups)**.** Let r be a positive integer. We construct a subgroup of $\mathrm{CT}(\mathbb{Z})$ isomorphic to the free group of rank r as follows:

(1) If $r = 1$, then return the cyclic group generated by $\tau \cdot \tau_{0(2),1(4)}$.
(2) Put $m := 2r$.
(3) For $i = 1, \ldots, r$, do the following:
- By Algorithm 4.10, construct an rcwa permutation $\sigma_i \in \mathrm{CT}(\mathbb{Z})$ which maps the complement $\mathbb{Z} \setminus 2i - 2(m)$ to the residue class $2i - 1(m)$.
(4) Return the group which is generated by $\sigma_1, \ldots, \sigma_r$.

This algorithm uses an adaptation of the construction given on page 27 in [1] from $\mathrm{PSL}(2, \mathbb{C})$ to $\mathrm{CT}(\mathbb{Z})$. As an analogue of the closed discs used there, the algorithm makes use of the residue classes modulo twice the rank of the free group.

In case it is not important whether the result is a subgroup of $\mathrm{CT}(\mathbb{Z})$ or not, we can also construct the σ_i in Step (3) by Algorithm 4.8. This sometimes yields rcwa permutations with smaller moduli.

Example 5.15. We construct an rcwa group isomorphic to the free group of rank 2. For this, we determine rcwa permutations σ_1 and σ_2 which map $\mathbb{Z} \setminus 0(4)$ to $1(4)$ and $\mathbb{Z} \setminus 2(4)$ to $3(4)$, respectively. We use Algorithm 4.8, and obtain

$$\sigma_1 : n \mapsto \begin{cases} 4n-3 & \text{if } n \in 1(2), \\ 2n+1 & \text{if } n \in 2(4), \\ n/4 & \text{if } n \in 0(8), \\ (n+2)/2 & \text{if } n \in 4(8) \end{cases} \quad \text{and} \quad \sigma_2 : n \mapsto \begin{cases} 4n-1 & \text{if } n \in 1(2), \\ 2n+7 & \text{if } n \in 0(4), \\ (n-2)/4 & \text{if } n \in 2(8), \\ (n-4)/2 & \text{if } n \in 6(8). \end{cases}$$

(cf. Example 4.9). Now we have $\mathrm{F}_2 \cong \langle \sigma_1, \sigma_2 \rangle$. This is in fact even a subgroup of $\mathrm{CT}(\mathbb{Z})$, since $\sigma_1 = \tau_{0(2),1(2)} \cdot \tau_{3(4),5(8)} \cdot \tau_{0(2),1(8)}$ and $\sigma_2 = \tau_{0(2),1(2)} \cdot \tau_{1(4),7(8)} \cdot \tau_{0(2),3(8)}$ are factorizations into generators.

We conclude this section by giving an algorithm to construct representations of free products of finite groups:

Algorithm 5.16 (Free Products of Finite Groups)**.** Let $G_0, \ldots, G_{m-1}$ be finite groups. We construct a subgroup of $\mathrm{CT}(\mathbb{Z})$ isomorphic to their free product as follows:

(1) If the factors are two cyclic groups of order 2, then return $\langle \tau, \tau_{0(2),1(4)} \rangle$.

(2) Determine regular permutation groups $P_0, \ldots, P_{m-1}$ which are isomorphic to the factors $G_0, \ldots, G_{m-1}$.
(3) Determine the images $H_0, \ldots, H_{m-1}$ of the permutation groups $P_0, \ldots, P_{m-1}$ under the embeddings $\varphi_{|P_0|}, \ldots, \varphi_{|P_{m-1}|}$.
(4) By Algorithm 4.10, for any $r \in \{0, \ldots, m-1\}$ find an element $\sigma_r \in \mathrm{CT}(\mathbb{Z})$ which maps the residue class $0(|P_r|)$ to the set $\mathbb{Z} \setminus r(m)$.
(5) Return the group which is generated by the conjugates $H_0^{\sigma_0}, \ldots, H_{m-1}^{\sigma_{m-1}}$.

This algorithm follows the proof of Theorem 4.2 in [8]. Similar to the above, in case it is not important whether the result is a subgroup of $\mathrm{CT}(\mathbb{Z})$ or not, we can also construct the σ_r in Step (4) by Algorithm 4.8.

Example 5.17. We construct an rcwa group which is isomorphic to the free product $\mathrm{C}_2 \star \mathrm{C}_3 \cong \mathrm{PSL}(2, \mathbb{Z})$ of the cyclic group of order 2 and the cyclic group of order 3:

- In Step (2), we put $P_0 := \langle (0,1) \rangle$ and $P_1 := \langle (0,1,2) \rangle$.
- In Step (3), we find $H_0 = \langle \tau \rangle$ and $H_1 = \langle \tau_{0(3),1(3)} \cdot \tau_{0(3),2(3)} \rangle$.
- In Step (4), we determine two conjugating rcwa permutations:
  - For $r = 0$, we need to find an rcwa permutation σ_0 which maps the residue class $0(2)$ to the set $\mathbb{Z} \setminus 0(2) = 1(2)$. For this, Algorithm 4.8 delivers $\sigma_0 := \tau$.
  - For $r = 1$ we need to find an rcwa permutation σ_1 which maps the residue class $0(3)$ to the set $\mathbb{Z} \setminus 1(2) = 0(2)$. For this, Algorithm 4.8 delivers $\sigma_1 := \alpha$, where α denotes Collatz' permutation mentioned in the introduction.
- In Step (5), we obtain the result $\mathrm{C}_2 \star \mathrm{C}_3 \cong \langle \tau, \tau_{1(4),3(4)} \cdot \tau_{0(2),1(4)} \rangle$.

6. Tame and Wild rcwa Groups

Certain rcwa groups have a particularly uncomplicated structure. Accordingly we call them *tame*:

Definition 6.1. We call an rcwa permutation σ *tame* if it permutes a partition of $\mathbb{Z}$ into finitely many residue classes on each of which it is affine, and *wild* otherwise. We call an rcwa group G *tame* if there is a common such partition for all elements of G, and *wild* otherwise. We call the specified partitions *respected partitions* of σ respectively G.

For example, any rcwa permutation of finite order is tame. The same holds for any finite rcwa group. Furthermore, all integral rcwa permutations are tame. An example of a tame rcwa permutation of infinite order is $n \mapsto n + 1$. This permutation respects the trivial partition of $\mathbb{Z}$.

Let G be a tame rcwa group, and assume that $\mathcal{P}$ is a respected partition of G. Then it is an immediate consequence of the definition that G embeds into the wreath product of the infinite dihedral group with the symmetric group of degree $|\mathcal{P}|$.

Tame groups are therefore not interesting in themselves, but they play an important role both in investigating rcwa groups by theoretical means and in computing in them.

For example, tameness is invariant under conjugation – if $\alpha \in \mathrm{RCWA}(\mathbb{Z})$ respects a partition $\mathcal{P}$, then a conjugate α^β respects the partition consisting of the images of the intersections of the residue classes in $\mathcal{P}$ with the sources of the affine partial mappings of β under β.

Furthermore if a tame group does not act faithfully on a respected partition, then the kernel of the action clearly does not act on $\mathbb{N}_0$. Thus as the group $\mathrm{CT}(\mathbb{Z})$ acts on $\mathbb{N}_0$, its tame subgroups are finite.

However, the product of two tame permutations is in general not tame. Tameness of products also does not induce an equivalence relation on the set of tame permutations: Let,

for example, $a := \tau_{1(6),4(6)}$, $b := \tau_{0(5),2(5)}$ and $c := \tau_{3(4),4(6)}$. Then ab and bc are tame, but ac is not.

In [6], a generalization of the above notion of tameness to not necessarily bijective rcwa mappings is considered. Namely, an rcwa mapping is called *tame* if the set of moduli of its powers is bounded, and *wild* otherwise. In [10] it is shown that in this sense, a surjective, but not injective rcwa mapping is always wild. Theorem 2.5.8 in [6] establishes the compatibility of these characterizations of tameness, namely it states that an rcwa group is tame if and only if the set of the moduli of its elements is bounded. This motivates the following definition:

Definition 6.2. Let G be an rcwa group. If the group G is tame, we define its *modulus* $\mathrm{Mod}(G)$ by the lcm of the moduli of its generators. If G is wild, we set $\mathrm{Mod}(G) := 0$.

In the sequel we give some criteria and methods to decide whether an rcwa group is tame or wild. Obviously, an rcwa group which has a wild element is itself wild. Therefore in order to disprove tameness, it is sufficient to find a wild element. In practice, good candidates can usually be found by looking at short products of generators and their inverses. On the other hand, tameness can be established by finding a respected partition.

Criterion 6.3 ('Balancedness' Criterion)**.** Let σ be an rcwa permutation. Assume that there is a prime p which divides the multiplier of the mapping σ, but not its divisor, or vice versa. Then σ is wild.

Proof. Without loss of generality we can assume that p divides the divisor, but not the multiplier of σ. Assume that σ is tame, and let $\mathcal{P}$ be a respected partition. Then there is a cycle $(r_0(m_0), \ldots, r_{l-1}(m_{l-1})) \subseteq \mathcal{P}$ such that the following hold:

(1) There is an index i such that $p|(m_i/m_{(i+1) \bmod l})$.
(2) There is no index j such that $p|(m_{(j+1) \bmod l}/m_j)$.

This yields a contradiction. □

For example, Collatz' permutation α mentioned in the introduction has multiplier 4 and divisor 3. Therefore Criterion 6.3 can be used to show that α is wild.

Two further useful criteria are based on certain directed graphs assigned to rcwa mappings:

Definition 6.4. Let f be an rcwa mapping, and let m be a positive integer. We define the *transition graph* $\Gamma(f, m)$ of f for modulus m as follows:

- The vertices of $\Gamma(f, m)$ are the residue classes (mod m).
- There is an edge from $r_1(m)$ to $r_2(m)$ if and only if $f(r_1(m)) \cap r_2(m) \neq \emptyset$.

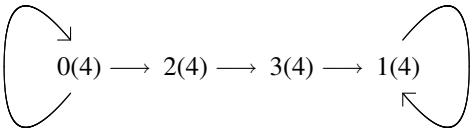
Criterion 6.5 (Sources-and-Sinks Criterion)**.** Let σ be an rcwa permutation. Assume that there is a positive integer m such that the transition graph $\Gamma(\sigma, m)$ has a weakly-connected component which is not strongly-connected. Then σ is wild.

This is part of the assertion of Theorem A.11 in [6].

Example 6.6. We look at the transition graph of $\sigma := \tau \cdot \tau_{0(2),1(4)}$ for modulus 4 in order to see that this rcwa permutation is wild. We have

$$\sigma \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} 2n-1 & \text{if } n \in 1(2), \\ n/2 & \text{if } n \in 0(4), \\ n+1 & \text{if } n \in 2(4), \end{cases}$$

and our graph looks like this:



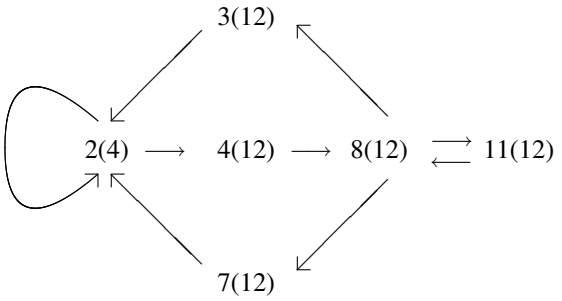
It is obvious that this graph is weakly-connected, but not strongly-connected. Therefore by Criterion 6.5, the rcwa permutation σ is indeed wild. Note, however, that Criterion 6.3 would not have established the wildness of σ.

There are wild rcwa permutations which can neither be shown to be wild by Criterion 6.3 nor by Criterion 6.5:

Example 6.7. Put $\sigma := \tau_{2(4),3(4)} \cdot \tau_{4(6),8(12)} \cdot \tau_{3(4),4(6)}$. Then we have

$$\sigma \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} (3n+2)/2 & \text{if } n \in 2(4), \\ (n+1)/3 & \text{if } n \in 8(12), \\ 2n & \text{if } n \in 4(12), \\ 2n-2 & \text{if } n \in 11(12), \\ n-1 & \text{if } n \in 3(12) \cup 7(12), \\ n & \text{if } n \in 1(4) \cup 0(12). \end{cases}$$

The permutation σ has cycles of any length $l \equiv 1 \pmod 3$, and the transition graph of σ for modulus 12 looks as follows (for simplicity, we have put together the vertices 2(12), 6(12) and 10(12)):



For such cases, there is yet another useful criterion:

Criterion 6.8 (Loops Criterion)**.** Let σ be an rcwa permutation, and let m be its modulus. Assume that the transition graph $\Gamma(\sigma, m)$ has loops. Then σ is wild.

For a proof, see [7]. We give an algorithm for computing transition graphs of rcwa mappings:

Algorithm 6.9 (Computation of Transition Graphs)**.** Let f be an rcwa mapping, and let m be a positive integer. We compute the transition graph $\Gamma(f, m)$ of f for modulus m as follows:

(1) Set up an $m \times m$ zero matrix M, with rows and columns labelled from 0 to $m-1$.
(2) For $n = 0, \ldots, \mathrm{Mod}(f) \cdot \mathrm{Div}(f) \cdot m - 1$, put $M_{n \bmod m, f(n) \bmod m} := 1$.
(3) Return the graph with vertices $0(m), \ldots, m-1(m)$, in which there is a directed edge from $r_1(m)$ to $r_2(m)$ if and only if $M_{r_1,r_2} = 1$.

Algorithm 6.9 basically does nothing other than running over a sufficiently long range of integers to pick up all possible transitions between residue classes (mod m) induced by f.

Further examples of drawings of transition graphs can be found in [6].

In practice, in order to apply Criterion 6.5 to an rcwa permutation σ, it is a sensible choice first to look at the transition graph $\Gamma(\sigma, \mathrm{Mod}(\sigma))$.

Assume that we have an rcwa group in which we have not found a wild element after having applied the above criteria to various of its elements. Then we may wish to verify that it is in fact tame. In order to do this, we need to construct a respected partition. For this, we first determine an educated guess for the modulus of our group:

Method 6.10 (Obtaining a Guess for the Modulus of an rcwa Group)**.** Let G be an rcwa group. In order to obtain a reasonable guess for the modulus of G, we proceed as follows:

- For $l = 1, 2, \ldots$, compute the least common multiple m_l of the moduli of all products of at most l elements of the union of the set of stored generators of G and the set of their inverses.
- Once the sequence $(m_l)_{l \in \mathbb{N}}$ seems to become stable, return its apparent limit.

Example 6.11. We would like to obtain an educated guess for the modulus of the group G which is generated by

$$g: \ n \mapsto \begin{cases} 2n+2 & \text{if } n \in 0(3), \\ n+4 & \text{if } n \in 1(6), \\ n/2 & \text{if } n \in 2(6), \\ n-4 & \text{if } n \in 4(6), \\ n-2 & \text{if } n \in 5(6) \end{cases} \quad \text{and} \quad h: \ n \mapsto \begin{cases} 2n+2 & \text{if } n \in 0(3), \\ n-2 & \text{if } n \in 1(6), \\ n/2 & \text{if } n \in 2(6), \\ n-1 & \text{if } n \in 4(6), \\ n+1 & \text{if } n \in 5(6). \end{cases}$$

We compute balls around 1, and look at the moduli of their elements:

- In the ball of radius 1, we find the identity and 4 elements with modulus 6.
- In the ball of radius 2, we find the identity, 8 elements with modulus 6, and 8 elements with modulus 12.
- In the ball of radius 3, we find the identity, 24 elements with modulus 6, and 28 elements with modulus 12.
- In the ball of radius 4, we find the identity, 44 elements with modulus 6, and 114 elements with modulus 12.

At this point, we guess that all elements of the group G have moduli which divide 12, and that therefore G has modulus 12. We will verify this guess in the example following the next algorithm.

The following algorithm assumes that we already know the modulus:

Algorithm 6.12 (Determination of Respected Partitions)**.** Let G be a tame rcwa group, and let m be its modulus. We determine a respected partition of G as follows:

(1) Let l be a list of the residue classes (mod m).
(2) By Algorithm 4.6, compute the support S of G.
(3) By Algorithm 3.6, write the complement of S as a union of 'few' residue classes, and let $\mathcal{P}$ be a list of these.
(4) Remove all residue classes from l which intersect trivially with S.
(5) Loop over l, until hitting a residue class $r(m)$ such that the moduli of all residue classes in the orbit Ω of $r(m)$ under the action of G divide m.

(6) Put $\mathcal{P} := \mathcal{P} \cup \Omega$ and $S := S \setminus \cup_{r_i(m_i)\in\Omega} r_i(m_i)$.
(7) If $S \neq \emptyset$ then go to Step (4), otherwise return $\mathcal{P}$.

Afterwards we can check easily whether our guessed modulus was correct. The only things we need to do are to verify whether $\mathcal{P}$ is indeed a list of single residue classes, that G acts on $\mathcal{P}$ and that all restrictions of generators of G to residue classes in $\mathcal{P}$ are affine.

Of course, Algorithm 6.12 can be used to compute respected partitions of tame rcwa permutations as well, if one applies it to the cyclic group which is generated by the respective permutation.

Example 6.13. We determine a respected partition of the group G from Example 6.11:

- In Step (1), we initialize l with $\{0(12), 1(12), \ldots, 11(12)\}$.
- In Step (2), we compute the support S of G, and obtain $S = \mathbb{Z}$.
- In Step (3), the complement is empty, thus we initialize $\mathcal{P}$ with the empty list.
- In Step (4), there is nothing to do, as no residue class intersects trivially with $\mathbb{Z}$.
- In Step (5), we first compute the orbit $0(12)^G$ of the residue class 0(12) under the action of G. We obtain

$$0(12)^G = \{0(12), 1(12), 3(12), 4(12), 5(12), 6(12), 7(12), \\ 9(12), 10(12), 11(12), 2(24), 8(24), 14(24), 20(24)\}.$$

  This orbit contains four residue classes whose moduli do not divide our guessed modulus 12. We can skip 1(12), since $1(12) \in 0(12)^G$. Next we compute $2(12)^G = \{0(6), 1(6), 3(6), 4(6), 5(6), 2(12), 8(12)\}$. This orbit satisfies our condition, thus we put $\Omega := 2(12)^G$.
- In Step (6), we add the residue classes in Ω to $\mathcal{P}$. Further we subtract the union of the residue classes in Ω from S.
- In Step (7), we observe that $S = \emptyset$, conclude that we are ready and find that our result is $\mathcal{P} = \{0(6), 1(6), 3(6), 4(6), 5(6), 2(12), 8(12)\}$.

7. Computing Group Orders and Element Orders

In this section, we explain how to compute the order of a given rcwa permutation and the order of a given rcwa group. Using respected partitions, the order of an rcwa permutation can be computed in the following way:

Method 7.1 (Order of an rcwa Permutation, Basic Approach)**.** Let σ be an rcwa permutation. We compute the order of σ as follows:

(1) By applying the criteria discussed in Section 6, check whether σ is tame. If it is not tame, then return ∞.
(2) By Algorithm 6.12, compute a respected partition $\mathcal{P}$ of σ.
(3) Determine the order k of the permutation which is induced by σ on the partition $\mathcal{P}$, and put $\tilde{\sigma} := \sigma^k$.
(4) If $\tilde{\sigma} = 1$, then return k.
(5) If σ is not class-wise order-preserving and $\tilde{\sigma}^2 = 1$, then return $2k$. Otherwise return ∞.

However if the modulus of σ is large, applying the wildness criteria and determining a respected partition both may take an unnecessarily long time.

Using a fancier approach, the order of an rcwa permutation can often be determined much faster. For example, one can sometimes detect infinite order by just evaluating a suitable epimorphism:

Remark 7.2. From [6], Section 2.11 and 2.12 we know that there are epimorphisms

$$\pi^+ : \ \mathrm{RCWA}^+(\mathbb{Z}) \to (\mathbb{Z}, +), \quad \sigma \ \mapsto \ \frac{1}{m} \sum_{r(m) \in \mathbb{Z}/m\mathbb{Z}} \frac{b_{r(m)}}{|a_{r(m)}|}$$

and

$$\pi^- : \ \mathrm{RCWA}(\mathbb{Z}) \to \mathbb{Z}^\times, \quad \sigma \ \mapsto \ (-1)^{\pi^+(\sigma) + \sum\limits_{r(m):\ a_{r(m)} < 0} \frac{m - 2r}{m}},$$

where we use the notation for the coefficients introduced in Definition 1.1.

For order computations, mainly the first of these epimorphisms is useful. However we will see later that both epimorphisms can be used in the membership test for rcwa groups.

Method 7.3 (Order of an rcwa Permutation, Fancier Approach)**.** Let σ be an rcwa permutation. We compute the order of σ as follows:

(1) If $\sigma = 1$, then return 1.
(2) If σ is class-wise order-preserving and does not lie in the kernel of the epimorphism π^+, then return ∞.
(3) Apply Criterion 6.3. If this establishes that σ is wild, then return ∞.
(4) Put $e := 1$.
(5) Repeat the following until e either remains stable during a prescribed number of iterations or exceeds a prescribed bound:
   (a) Pick a 'random' integer n.
   (b) Attempt to compute the cycle of σ containing n. Stop if the length or the elements exceed some prescribed bounds.
   (c) If the cycle has been determined in full, then let l be its length and put $e := \mathrm{lcm}(e, l)$. Otherwise exit the loop and assume that e has exceeded the prescribed bound.
(6) If e has not exceeded the prescribed bound, then check whether $\sigma^e = 1$. If so, then return e. When computing the power σ^e, stop if the modulus exceeds a prescribed bound.
(7) Compute the order of σ by Method 7.1, and return the result.

Example 7.4. We would like to determine the order of $\sigma := \tau_{0(6),4(6)} \cdot \tau_{3(4),0(6)} \cdot \tau_{2(4),1(6)}$. We have

$$\sigma \in \mathrm{CT}(\mathbb{Z}) : \quad n \ \longmapsto \ \begin{cases} (3n-9)/2 & \text{if } n \in 3(8), \\ (9n-35)/4 & \text{if } n \in 7(8), \\ n+4 & \text{if } n \in 0(12), \\ (2n+4)/3 & \text{if } n \in 1(12), \\ (3n-4)/2 & \text{if } n \in 2(12), \\ (3n+8)/2 & \text{if } n \in 6(12), \\ (2n+1)/3 & \text{if } n \in 4(18) \cup 16(18), \\ (4n+14)/9 & \text{if } n \in 10(18), \\ n & \text{if } n \in 5(12) \cup 8(12) \cup 9(12). \end{cases}$$

Using Method 7.3, we proceed as follows:

- The rcwa permutation σ is class-wise order-preserving, thus in Step (2) we check whether it lies in the kernel of the epimorphism π^+: We have

$$\pi^+(\sigma) = \frac{1}{8} \cdot \left(\frac{-9}{3} + \frac{-35}{9} \right) + \frac{1}{12} \cdot \left(\frac{4}{1} + \frac{4}{2} + \frac{-4}{3} + \frac{8}{3} \right) + \frac{2}{18} \cdot \frac{1}{2} + \frac{1}{18} \cdot \frac{14}{4} = 0,$$

  therefore at this point we cannot reach any conclusions concerning the order of σ.
- In Step (3), we find that $\mathrm{Mult}(\sigma) = \mathrm{Div}(\sigma) = 36$. Therefore Criterion 6.3 is not applicable.
- Now we look at cycles of σ. We put $e := 1$, and choose a tentative order bound of 1000.
  - We pick $n = 0$. The cycle of σ containing 0 is (0,4,3). This cycle has length 3, therefore we put $e := \mathrm{lcm}(e, 3) = 3$.
  - We pick $n = 1$, and obtain the cycle (1,2). This cycle has length 2, therefore we put $e := \mathrm{lcm}(e, 2) = 6$.
  - We pick $n = 6$, and obtain the cycle (6,13,10). The length of this cycle already divides e.
  - We pick $n = 14$, and obtain the cycle (14,19,24,28) of length 4. We put $e := \mathrm{lcm}(e, 4) = 12$.
  - We pick $n = 15$, and obtain the cycle (15,25,18,31,61,42,67,96,100,46,22) of length 11. Now we already have $e := \mathrm{lcm}(e, 11) = 132$.
  - We pick $n = 23$, and obtain the cycle (23,43,60,64,30,49,34) of length 7. This yields $e := \mathrm{lcm}(e, 7) = 924$.
  - We pick $n = 38$, and obtain the cycle (38,55,115,168,172,78,121,82) of length 8. This yields $e := \mathrm{lcm}(e, 8) = 1848$, which is beyond our bound of 1000.

Since we have failed to determine the order of σ so far, we switch to Method 7.1.

In Step (1) of this method, it is checked whether one of our wildness criteria is applicable. We had already checked the 'Balancedness' Criterion 6.3. Next we check the 'Sources-and-Sinks' Criterion 6.5. This fails to be applicable as well, thus it remains to check the 'Loops' Criterion 6.8. Indeed we find that the vertices 7(72) and 46(72) of the transition graph of σ for modulus 72 carry loops, which certifies that σ is wild and has therefore infinite order.

We give a method to determine the order of an rcwa group:

Method 7.5 (Order of an rcwa Group)**.** Let G be an rcwa group. We compute the order of G as follows:

(1) If G is class-wise order-preserving, and if any of the generators does not lie in the kernel of the epimorphism π^+, then return ∞.
(2) By applying the criteria discussed in Section 6, check whether G is wild. If it is, then return ∞.
(3) By Algorithm 6.12, construct a respected partition $\mathcal{P}$ of G.
(4) If G is not class-wise order-preserving, then refine the partition $\mathcal{P}$ by splitting each of the residue classes $r_i(m_i) \in \mathcal{P}$ into 3 residue classes $r_i(3m_i)$, $r_i + m_i(3m_i)$ and $r_i + 2m_i(3m_i)$.
(5) Check whether the action of G on $\mathcal{P}$ is faithful. For this, proceed as follows:
    (a) Choose a set of representatives for the residue classes in $\mathcal{P}$.

(b) Check whether the orbits of all these representatives under the action of G have length at most $|\mathcal{P}|$. If yes, then the action is faithful. If not, then it is not faithful.

(6) If the action of G on $\mathcal{P}$ is faithful, then return the order of the permutation group induced by G on $\mathcal{P}$. Otherwise return ∞.

Example 7.6. We compute the order of the group G from Example 6.11:

- In Step (1), we find that G is class-wise order-preserving, but that both generators lie in the kernel of the epimorphism π^+. Therefore we cannot yet reach any conclusions concerning the group order.
- Step (2)-(4): We already know from Example 6.13 that the group G is tame and that it respects the partition $\mathcal{P} = \{0(6), 1(6), 3(6), 4(6), 5(6), 2(12), 8(12)\}$. Since G is class-wise order-preserving, there is no need to refine this partition.
- Step (5)-(6): We choose the set $\{0, 1, 3, 4, 5, 2, 8\}$ of representatives for the residue classes in $\mathcal{P}$. We start with the element 0 of this set, and put $S := \{0\}$. Then we unite the set S with its images under the generators of G: After doing this for the first time, we get $S = \{0, 2\}$, after the second time, we get $S = \{0, 1, 2\}$, after the third time, we get $S = \{-1, 0, 1, 2, 5\}$ and after the fourth time, we get $S = \{-3, -1, 0, 1, 2, 3, 5, 6\}$. Now we have $|S| = 8 > 7 = |\mathcal{P}|$, which shows that the action of G on $\mathcal{P}$ is not faithful. We conclude that $|G| = \infty$.

8. Extracting Roots of Torsion Elements

Given an rcwa permutation $\sigma \in \mathrm{CT}(\mathbb{Z})$ of finite order and given a positive integer k, there is always an rcwa permutation $\tilde{\sigma}$ such that $\tilde{\sigma}^k = \sigma$ (cf. Theorem 2.6 in [8]). We describe an algorithm to extract such roots:

Algorithm 8.1 (Roots of an rcwa Permutation of Finite Order)**.** Let σ be an rcwa permutation of finite order which can be written as a product of class transpositions, and let $k \in \mathbb{N}$. We determine a k-th root $\tilde{\sigma}$ of σ as follows:

(1) Put $o := \mathrm{ord}(\sigma)$. Further let k_{sing} be the product of the prime factors of k which divide o, and put $k_{\mathrm{reg}} := k/k_{\mathrm{sing}}$.
(2) Put $\tilde{\sigma}_{\mathrm{reg}} := \sigma^{\tilde{k}_{\mathrm{reg}}}$, where $\tilde{k}_{\mathrm{reg}}$ denotes the inverse of k_{reg} modulo o.
(3) If $k_{\mathrm{sing}} = 1$, then return $\tilde{\sigma}_{\mathrm{reg}}$.
(4) By Algorithm 6.12, compute a respected partition $\mathcal{P}$ of $\tilde{\sigma}_{\mathrm{reg}}$.
(5) Put $\tilde{\sigma} := 1$.
(6) Repeat the following until $\mathcal{P} = \emptyset$:
  (a) Pick a residue class $r(m)$ from $\mathcal{P}$, and compute the cycle c of $\tilde{\sigma}_{\mathrm{reg}}$ on $\mathcal{P}$ which contains $r(m)$. Let l be the length of c.
  (b) Remove the residue classes in the cycle c from $\mathcal{P}$.
  (c) Replace all residue classes in the cycle c by their partitions into k_{sing} residue classes with the k_{sing}-fold modulus. This means that a residue class $r(m)$ is replaced by the list of the residue classes (mod $k_{\mathrm{sing}} \cdot m$) with the residues $r, r+m, r+2m, \ldots, r+(k_{\mathrm{sing}}-1)m$. Therefore c is now an $l \times k_{\mathrm{sing}}$-matrix of residue classes.
  (d) For $i = 1, \ldots, l$ do for $j = 1, \ldots, k_{\mathrm{sing}}$ do if i and j are not both equal to 1, then put $\tilde{\sigma} := \tilde{\sigma} \cdot \tau_{c_{1,1}, c_{i,j}}$.
(7) Return $\tilde{\sigma}$.

Example 8.2. Given

$$\sigma \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} 2n+3 & \text{if } n \in 0(2), \\ (n-1)/2 & \text{if } n \in 1(4), \\ n-2 & \text{if } n \in 3(4), \end{cases}$$

we would like to find an rcwa permutation $\tilde{\sigma}$ such that $\tilde{\sigma}^{60} = \sigma$. We proceed as follows:

- In Step (1), we determine the order of σ, and obtain $o = 3$. Further we let k_{sing} be the product of the prime factors of 60 which divide 3, thus $k_{\mathrm{sing}} := 3$, and put $k_{\mathrm{reg}} := 60/k_{\mathrm{sing}} = 20$.
- In Step (2), we determine the inverse $\tilde{k}_{\mathrm{reg}}$ of 20 modulo 3, and obtain $\tilde{k}_{\mathrm{reg}} = 2$. Accordingly we put $\tilde{\sigma}_{\mathrm{reg}} := \sigma^2$. We have

$$\tilde{\sigma}_{\mathrm{reg}} \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} 2n+1 & \text{if } n \in 0(2), \\ n+2 & \text{if } n \in 1(4), \\ (n-3)/2 & \text{if } n \in 3(4). \end{cases}$$

- In Step (4), we compute a respected partition $\mathcal{P}$ of $\tilde{\sigma}_{\mathrm{reg}}$ by Algorithm 6.12. We obtain $\mathcal{P} = \{0(2), 1(4), 3(4)\}$.
- Step (5)-(7): We initialize $\tilde{\sigma}$ with 1, and proceed as follows:
  - In Step (6.a), we pick the residue class 0(2) from $\mathcal{P}$, and compute the cycle $c = (0(2), 1(4), 3(4))$ of $\tilde{\sigma}_{\mathrm{reg}}$ on $\mathcal{P}$ which contains 0(2).
  - In Step (6.b), we remove the residue classes in c from $\mathcal{P}$. Afterwards, in our example we already have $\mathcal{P} = \emptyset$.
  - In Step (6.c), we split each of the residue classes in c into $k_{\mathrm{sing}} = 3$ parts, and get $c = ((0(6), 2(6), 4(6)), (1(12), 5(12), 9(12)), (3(12), 7(12), 11(12)))$.
  - In Step (6.d), we multiply $\tilde{\sigma}$ by the class transpositions $\tau_{0(6),2(6)}$, $\tau_{0(6),4(6)}$, $\tau_{0(6),1(12)}$, $\tau_{0(6),5(12)}$, $\tau_{0(6),9(12)}$, $\tau_{0(6),3(12)}$, $\tau_{0(6),7(12)}$, and $\tau_{0(6),11(12)}$ in the given order.

  In this example, we need to run through the loop in Step (6) only once, since $\tilde{\sigma}_{\mathrm{reg}}$ induces only one cycle on $\mathcal{P}$. We finally obtain our 60th root

$$\tilde{\sigma} \in \mathrm{CT}(\mathbb{Z}): \quad n \longmapsto \begin{cases} n+2 & \text{if } n \in 0(6) \cup 2(6), \\ n+4 & \text{if } n \in 1(6) \cup 3(12) \cup 5(12), \\ 2n-7 & \text{if } n \in 4(6), \\ n-6 & \text{if } n \in 9(12), \\ (n-11)/2 & \text{if } n \in 11(12). \end{cases}$$

Looking at the monomorphisms φ_m, it is an immediate observation that any finite group embeds into a divisible residue-class-wise affine torsion group.

9. On the Membership Problem

The purpose of this section is to present a couple of methods and criteria which can be used to solve the following problem by means of computation:

Problem 9.1 (Membership Problem)**.** Given an rcwa group G with generators $\sigma_1, \sigma_2, \ldots$ and given a further rcwa permutation σ, decide whether σ is an element of G or not.

We will use the notation introduced in Problem 9.1 throughout this section. At the beginning, we list a number of criteria for disproving membership.

In Section 6, we have discussed tame rcwa permutations and -groups. We immediately get the following criterion:

Criterion 9.2 (Tame Group, Wild Permutation Criterion)**.** If σ is wild and G is tame, then σ is not an element of G.

This shows e.g. that Collatz' permutation α mentioned in the introduction does not lie in the group G given in Example 6.11.

Later in this section we will describe how to solve the membership problem for tame rcwa groups. However, there are criteria whose application is computationally much cheaper. For example, from Lemma 2.5 we obtain the following criterion:

Criterion 9.3 (Multiplier-and-Divisor Criterion)**.** If the multiplier or the divisor of σ has a prime factor which divides neither the multiplier nor the divisor of any of the generators σ_i, then σ is not an element of G.

This shows again that Collatz' permutation α is not an element of the group G from Example 6.11 – both generators of G have multiplier and divisor 2, but the divisor of Collatz' permutation is 3.

Lemma 2.7, Assertion (2), yields a very similar criterion for the moduli:

Criterion 9.4 (Modulus Criterion)**.** If the modulus of σ has a prime factor which divides the modulus of none of the generators σ_i, then σ is not an element of G.

Example 9.5. We look again at the group G from Example 6.11. Further let

$$\sigma \in \mathrm{CT}(\mathbb{Z}): \quad n \mapsto \begin{cases} n+1 & \text{if } n \in 0(5) \cup 1(5) \cup 2(5) \cup 3(5), \\ n-4 & \text{if } n \in 4(5). \end{cases}$$

Then we have $\mathrm{Mod}(\sigma) = 5$. Since the generators of G have both modulus 6, Criterion 9.4 shows that $\sigma \notin G$.

A further obvious criterion is the following:

Criterion 9.6 (Class-Wise Order-Preservingness Criterion)**.** If all generators σ_i are class-wise order-preserving, but σ is not, then σ is not an element of G.

Criterion 9.6 shows for example that the permutation $n \mapsto -n$ is not an element of the group G from Example 6.11.

There are some very simple criteria based on the action of G on $\mathbb{Z}$:

Criterion 9.7 (Support Criterion)**.** If the support of σ is not a subset of the support of G, then σ is not an element of G.

Criterion 9.8 (Fixes Nonnegative Integers Criterion)**.** If all σ_i map the set of nonnegative integers to itself, but σ does not do so, then σ is not an element of G.

Criterion 9.8 shows for example that the permutation $n \mapsto n+1$ does not lie in the group $\mathrm{CT}(\mathbb{Z})$.

Another such criterion uses finite orbits:

Criterion 9.9 (Finite Orbits Criterion)**.** If there is a finite orbit on $\mathbb{Z}$ under the action of G which σ does not map to itself, then σ is not an element of G. The same holds if there is a finite orbit on which σ induces a permutation which does not lie in the permutation group induced by G on that orbit.

In practice one can choose an interval $I := \{-n, \ldots, n\}$, compute all finite orbits of G on $\mathbb{Z}$ which intersect nontrivially with I and whose length is less than some given bound or whose elements all have an absolute value less than a given bound, and check whether Criterion 9.9 applies to one of them.

Example 9.10. Let $G := \langle \tau_{10(12),0(15)} \cdot \tau_{3(14),8(14)}, \tau_{1(5),10(15)} \rangle$ and $\sigma := \tau_{10(12),0(15)}$. We show that $\sigma \notin G$:

- None of the Criteria 9.2, 9.3, 9.4, 9.6, 9.7 or 9.8 is applicable.
- Although the group G has very many finite orbits of very many different lengths, we don't find one which is not fixed setwise by σ as well.

  ('Typically', if there are many finite orbits, the chance that all of them are fixed by a given non-member as well is very small – thus 'usually' non-membership in a group with many finite orbits can already be certified at this point.)
- We observe that the group G has the finite orbit

  $$\Omega := \{31, 36, 100, 101, 106, 115, 120, 269, 274, 310, 325, 330, 375\}.$$

  We further observe that G induces on Ω the group

  $$H := \langle (31,36)(101,106,115,120)(269,274,325,330)(310,375), \\ (31,100)(36,115)(101,310)(106,325) \rangle \cong \mathrm{PSL}(3,3),$$

  and that σ induces on Ω the permutation

  $$\tilde{\sigma} := (106,120)(274,330)(310,375).$$

  Using standard methods to test membership in a finite permutation group, we find that $\tilde{\sigma} \notin H$. Now by Criterion 9.9, we conclude that $\sigma \notin G$.

A further criterion uses partitions of $\mathbb{Z}$ into unions of residue classes:

Criterion 9.11 (Invariant Partition Criterion)**.** Let m be a positive integer, and assume that there is a partition $\mathcal{P}$ of $\mathbb{Z}$ into unions of residue classes (mod m) which is invariant under the action of G. If σ does not leave $\mathcal{P}$ invariant, then σ is not an element of G.

In practice, sensible choices for m are multiples of the moduli of the generators of G.

Example 9.12. Let $G := \langle \tau_{1(4),2(4)} \cdot \tau_{1(4),3(4)}, \tau_{3(9),6(18)} \cdot \tau_{1(6),3(9)} \rangle$ and $\sigma := \tau_{1(6),9(36)}$. We show that $\sigma \notin G$:

- We find that G leaves the partition

  $$\mathcal{P} := \{1(6) \cup 3(9) \cup 2(12) \cup 5(12) \cup 6(18) \cup 15(36) \cup 18(36) \\ \cup 22(36) \cup 23(36) \cup 27(36),\ 9(36) \cup 10(36) \cup 11(36), \\ 33(36) \cup 34(36) \cup 35(36),\ 4(12) \cup 8(12) \cup 0(36)\}$$

  of $\mathbb{Z}$ into unions of residue classes (mod 36) invariant.
- We observe that the residue classes 1(6) and 9(36) which are interchanged by σ belong to distinct parts in $\mathcal{P}$. Now by Criterion 9.11, we conclude that $\sigma \notin G$.

Two further criteria make use of the epimorphisms described in Remark 7.2:

Criterion 9.13 (First Epimorphism Criterion)**.** If all generators of G lie in the kernel of π^-, but σ does not do so, then σ is not an element of G.

Criterion 9.14 (Second Epimorphism Criterion)**.** Assume that $G < \mathrm{RCWA}^+(\mathbb{Z})$, that σ is class-wise order-preserving and that $\gcd\{\pi^+(\sigma_1), \ldots, \pi^+(\sigma_r)\} \nmid \pi^+(\sigma)$. Then σ is not an element of G.

Example 9.15. Let $G := \langle \sigma_1, \sigma_2 \rangle < \mathrm{RCWA}(\mathbb{Z})$, where σ_1 and σ_2 are given by

$$n \mapsto \begin{cases} -2n+2 & \text{if } n \in 0(12), \\ n+2 & \text{if } n \in 1(6) \cup 5(6), \\ (n-2)/2 & \text{if } n \in 2(24) \cup 10(24), \\ -n+2 & \text{if } n \in 3(6), \\ 2n+2 & \text{if } n \in 4(12) \cup 8(12), \\ n & \text{if } n \in 6(24) \cup 22(24), \\ -n+4 & \text{if } n \in 14(24), \\ (-n+10)/2 & \text{if } n \in 18(24) \end{cases} \quad \text{and} \quad n \mapsto \begin{cases} n+3 & \text{if } n \in 0(10), \\ n+10 & \text{if } n \in 1(10), \\ -n+4 & \text{if } n \in 2(5), \\ n-3 & \text{if } n \in 3(10), \\ -n+8 & \text{if } n \in 4(5), \\ -n+10 & \text{if } n \in 5(10), \\ n+2 & \text{if } n \in 6(10), \\ n-2 & \text{if } n \in 8(10), \end{cases}$$

respectively. Further let

$$\sigma \in \mathrm{RCWA}(\mathbb{Z}) : \quad n \mapsto \begin{cases} 2n+3 & \text{if } n \in 0(10) \cup 6(10) \cup 8(10), \\ n-5 & \text{if } n \in 1(20) \cup 17(20), \\ -2n+6 & \text{if } n \in 2(10) \cup 4(10), \\ (n-3)/2 & \text{if } n \in 3(40) \cup 11(40) \cup 19(40), \\ n & \text{if } n \in 5(20) \cup 9(20) \cup 13(20), \\ (n-13)/2 & \text{if } n \in 7(20) \cup 15(20), \\ (-n+11)/2 & \text{if } n \in 23(40) \cup 31(40) \cup 39(40). \end{cases}$$

A couple of additions, subtractions, multiplications and divisions of coefficients reveal that $\pi^-(\sigma_1) = \pi^-(\sigma_2) = 1$, but $\pi^-(\sigma) = -1$ (it is even feasible to do these computations by hand!). By Criterion 9.13, we conclude that $\sigma \notin G$.

We describe a method to solve the membership problem for tame rcwa groups:

Method 9.16 (Membership Test for Tame Groups)**.** Assume that G and σ are tame, and let m denote the modulus of G. In order to find out whether σ is an element of G or not, we proceed as follows:

(1) If $\mathrm{Mod}(\sigma) \nmid m$, then return *false*.
(2) By Algorithm 6.12, determine a respected partition $\mathcal{P}$ of G.
(3) If σ does not respect the partition $\mathcal{P}$, then return *false*.
(4) Determine the permutation $\sigma_\mathcal{P}$ which is induced by σ on $\mathcal{P}$, and the finite permutation group $G_\mathcal{P}$ which is induced by G on $\mathcal{P}$.
(5) If $\sigma_\mathcal{P} \notin G_\mathcal{P}$, then return *false*.
(6) By element factorization in the finite permutation group $G_\mathcal{P}$, determine an element $\tilde{\sigma}$ of G which induces the same permutation $\sigma_\mathcal{P}$ on $\mathcal{P}$ as σ does.
(7) Put $k := \sigma \cdot \tilde{\sigma}^{-1}$.
(8) Compute the kernel K of the action of G on $\mathcal{P}$.
(9) Check whether k is an element of K. As K is a polycyclic group, this can be done by the membership test for polycyclically presented groups. If $k \in K$, then return *true*, otherwise return *false*.

While the author knows a method to perform Step (8) and has implemented it in RCWA, this method can likely be improved and be turned into an algorithm. Therefore, here we only give a brief outline of how it works:

Method 9.17 (Outline: Kernel of Action on Respected Partition)**.** Perform a random walk on the group G, and take powers of the elements encountered along the way by the orders

of the induced permutations on the respected partition $\mathcal{P}$ to get kernel elements. Collect kernel generators in this way until some condition is fulfilled from which one can conclude that the generators picked up so far generate indeed the entire kernel.

Turning Method 9.17 into an algorithm would turn Method 9.16 into an algorithm as well. Concerning algorithms for finite permutation groups and polycyclic groups, we refer to [4]. For the latter, we refer also to [2].

We observe that the kernel of the action of an rcwa group on a respected partition is generated by products of rcwa permutations of the following types:

Definition 9.18. Let $r(m) \subseteq \mathbb{Z}$ be a residue class.

(1) We define the *class shift* $\nu_{r(m)} \in \mathrm{RCWA}(\mathbb{Z})$ by

$$\nu_{r(m)} : \ n \ \mapsto \begin{cases} n+m & \text{if } \ n \in r(m), \\ n & \text{otherwise.} \end{cases}$$

(2) We define the *class reflection* $\varsigma_{r(m)} \in \mathrm{RCWA}(\mathbb{Z})$ by

$$\varsigma_{r(m)} : \ n \ \mapsto \begin{cases} -n+2r & \text{if } \ n \in r(m), \\ n & \text{otherwise,} \end{cases}$$

where we assume that $0 \leqslant r < m$.

For convenience, we set $\nu := \nu_{\mathbb{Z}} : n \mapsto n+1$ and $\varsigma := \varsigma_{\mathbb{Z}} : n \mapsto -n$.

Example 9.19. Let G be the group generated by

$$\sigma_1 : \ n \ \mapsto \begin{cases} 2n+1 & \text{if } n \in 0(2), \\ (n-1)/2 & \text{if } n \in 1(4), \\ -n+6 & \text{if } n \in 3(4) \end{cases} \quad \text{and} \quad \sigma_2 : \ n \ \mapsto \begin{cases} n+6 & \text{if } n \in 0(3), \\ -n+2 & \text{if } n \in 1(3), \\ n & \text{if } n \in 2(3). \end{cases}$$

We would like to find out whether

$$\sigma \in \mathrm{RCWA}(\mathbb{Z}) : \ n \ \mapsto \begin{cases} 2n-419 & \text{if } n \in 0(6), \\ (-n+73)/2 & \text{if } n \in 1(12), \\ 2n+1 & \text{if } n \in 2(6), \\ (n+161)/2 & \text{if } n \in 3(12), \\ -2n-115 & \text{if } n \in 4(6), \\ (n-1)/2 & \text{if } n \in 5(12), \\ n+4 & \text{if } n \in 7(12), \\ -n-276 & \text{if } n \in 9(12), \\ -n+6 & \text{if } n \in 11(12). \end{cases}$$

is an element of G or not. We proceed as follows:

- First of all, we check that none of our wildness criteria applies to any of the elements of a ball of small radius around 1 in G. After that, we already have some confidence that G is tame.
- In Step (1), we use Method 6.10, to determine an educated guess for the modulus of G. We obtain 12. Since this is a multiple of $\mathrm{Mod}(\sigma) = 12$, we cannot disprove membership at this point.

- In Step (2), we use Algorithm 6.12 to determine a respected partition $\mathcal{P}$ of G. We obtain

$$\mathcal{P} \;=\; \{0(6), 2(6), 4(6), 1(12), 3(12), 5(12), 7(12), 9(12), 11(12)\}.$$

  As a side-effect, now we know that G is tame.
- In Step (3), we check whether σ respects the partition $\mathcal{P}$. It does so, thus we cannot disprove membership at this point.
- In Step (4), we determine the permutation

$$\sigma_{\mathcal{P}} \;=\; (0(6), 1(12))\,(2(6), 5(12))\,(4(6), 9(12), 3(12))\,(7(12), 11(12))$$

  induced by σ on $\mathcal{P}$, and the group

$$G_{\mathcal{P}} \;=\; \langle (0(6), 1(12))\,(2(6), 5(12))\,(4(6), 9(12))\,(7(12), 11(12)),\\ (3(12), 9(12))\rangle$$

  induced by G on $\mathcal{P}$.
- In Step (5), we check whether $\sigma_{\mathcal{P}}$ is an element of $G_{\mathcal{P}}$. This is the case (it is just the 5th power of the product of the two generators), thus we still cannot disprove membership of σ in G.
- In Step (6), we put $\tilde{\sigma} := (\sigma_1 \cdot \sigma_2)^5$. Now, $\tilde{\sigma}$ induces on $\mathcal{P}$ the same permutation as σ does.
- In Step (7), we put $k := \sigma \cdot \tilde{\sigma}^{-1}$ and obtain

$$k \in \mathrm{RCWA}(\mathbb{Z}) : \;\; n \;\longmapsto\; \begin{cases} -n + 210 & \text{if } n \in 0(6), \\ -n + 62 & \text{if } n \in 1(12), \\ n & \text{if } n \in 2(6) \cup 5(12), \\ n + 168 & \text{if } n \in 3(12), \\ n + 60 & \text{if } n \in 4(6), \\ -n + 2 & \text{if } n \in 7(12), \\ n + 288 & \text{if } n \in 9(12), \\ -n + 10 & \text{if } n \in 11(12). \end{cases}$$

- In Step (8), we compute the kernel K of the action of G on $\mathcal{P}$. We obtain

$$K \;=\; \langle \varsigma_{0(6)} \cdot \varsigma_{1(6)} \cdot \varsigma_{11(12)} \cdot \nu_{11(12)}^{-1} \cdot \nu_{1(12)} \cdot \nu_{0(6)},\\ \nu_{0(6)}^2, \nu_{9(12)} \cdot \nu_{4(6)}, \nu_{1(12)}^2, \nu_{3(6)}^2 \rangle.$$

- Now our task is to find out whether k is an element of K or not. We proceed as follows:
  - We look at the group

$$\hat{K} \;:=\; \{\varsigma_{r(m)}, \nu_{r(m)} \mid r(m) \in \mathcal{P}\},$$

    and set up the natural isomorphism φ from $\hat{K}$ to the polycyclic group $\mathrm{D}_\infty^{|\mathcal{P}|}$.
  - We compute the images of K and k under φ. The image of an rcwa permutation g under φ can be computed as follows:
    - Compute the restrictions $g|_{r(m)}$ of g to the residue classes $r(m) \in \mathcal{P}$.
    - Write any $g|_{r(m)}$ in the form $\varsigma_{r(m)}^{\epsilon} \cdot \nu_{r(m)}^{n}$, where $\epsilon \in \{0, 1\}$ and $n \in \mathbb{Z}$.
    - Replace $\varsigma_{r(m)}$ and $\nu_{r(m)}$ by the generators of order 2 respectively ∞ of the corresponding direct factor of the image of φ.

  - Using algorithms for polycyclic groups as implemented in [2], we check whether $k \in K$. We find that k is indeed an element of K.
- We conclude that $\sigma \in G$.

As said, the membership problem for rcwa groups cannot be solved by algorithmic means in general. But this does not mean that basically nothing further can be done if we have some wild rcwa group and some rcwa permutation to which none of the non-membership criteria applies.

If σ is an element of G, then this can in principle certainly be verified by finding a factorization into the stored generators. A naive way of looking for such a factorization is to compute all products of $1, 2, 3, \ldots$ generators and / or inverses of generators, and to check whether one of them equals σ.

However this quickly gets infeasible if σ cannot be written as a product of only 'very few' generators, and it is also obvious that non-membership cannot be decided in this way. Nevertheless, assuming that σ is an element of G, it is possible to speed up the factorization process considerably.

In our factorization method, we need a method to determine an element of G which maps a given tuple of distinct integers to a given other such tuple:

Method 9.20 (Group Element Which Maps a Given Tuple to a Given Other Tuple)**.** Let $G = \langle \sigma_1, \ldots, \sigma_r \rangle$ be an rcwa group. Further let k be a positive integer, and let $(n_1, \ldots, n_k)$ and $(\tilde{n}_1, \ldots, \tilde{n}_k)$ be k-tuples of pairwise distinct integers. In order to determine an element σ of G such that $(n_1^\sigma, \ldots, n_k^\sigma) = (\tilde{n}_1, \ldots, \tilde{n}_k)$, we proceed as follows:

(1) Let $F = \langle f_1, \ldots, f_r \rangle$ be the free group whose rank is the number of stored generators of G. Further let $\mathcal{G}_F$ be the set of generators of F and their inverses, and let $\pi : F \to G$, $f_i \mapsto \sigma_i$ denote the natural epimorphism.
(2) Put $\Omega := \{((n_1, \ldots, n_k), 1)\}$ and $\tilde{\Omega} := \{((\tilde{n}_1, \ldots, \tilde{n}_k), 1)\}$.
    The elements of the sets Ω and $\tilde{\Omega}$ will always be lists of pairs, each consisting of a k-tuple of integers and an element of the free group F. In any pair $(t, f) \in \Omega$, the tuple t will be the image of $(n_1, \ldots, n_k)$ under $\pi(f)$. Similarly, in any pair $(\tilde{t}, \tilde{f}) \in \tilde{\Omega}$ the tuple $\tilde{t}$ will be the image of $(\tilde{n}_1, \ldots, \tilde{n}_k)$ under $\pi(\tilde{f})$.
(3) Repeat the following until there are pairs $(t, f) \in \Omega$ and $(\tilde{t}, \tilde{f}) \in \tilde{\Omega}$ with $t = \tilde{t}$:
    (a) Extend Ω by the set of all pairs which one can get from some pair in Ω by multiplying the second entry by an element f of $\mathcal{G}_F$ and taking the image of the first entry under $\pi(f)$. Extend $\tilde{\Omega}$ in the analogous way.
    (b) If there are several pairs in Ω respectively $\tilde{\Omega}$ whose first entries coincide, then in any such instance remove all of them except of one. As remaining representative choose a pair whose second entry has the smallest word length in terms of generators of F.
(4) Choose any two pairs $(t, f) \in \Omega$ and $(\tilde{t}, \tilde{f}) \in \tilde{\Omega}$ with $t = \tilde{t}$, and return $\pi(f \cdot \tilde{f}^{-1})$. If an expression of the result as a product of generators of G is desired, then return the list of the images of the letters of the word $f \cdot \tilde{f}^{-1}$ under π as well.

If an element with the desired property exists, then it is found after a finite number of steps. Otherwise the method runs into an infinite loop. Using words in a free group avoids lots of potencially time- and memory-consuming multiplications of rcwa permutations. Method 9.20 can be adapted in a trivial way to work for finite sets instead of tuples.

Now we can describe our factorization method:

Method 9.21 (Factorization of Group Elements into Generators)**.** As always in this section, let $G = \langle \sigma_1, \ldots, \sigma_r \rangle$ be an rcwa group. Further assume that $\sigma \in G$. We factor σ into the generators $\sigma_1, \ldots, \sigma_r$ as follows:

(1) Put $k := 1$.
(2) By Method 9.20, compute some $\tilde{\sigma} \in G$ such that $(1^{\tilde{\sigma}}, \ldots, k^{\tilde{\sigma}}) = (1^{\sigma}, \ldots, k^{\sigma})$. This method can also express the returned element $\tilde{\sigma}$ as a product of generators of G. Let l be the list of these factors.
(3) If $\tilde{\sigma} = \sigma$, then return l. Otherwise put $k := k + 1$, and go to Step (2).

The runtime- and memory requirements of Method 9.20 and 9.21 are exponential in the word length. They are practical for word lengths of, say, at most 20 or 30. The following example still needs less than a second in RCWA [9]:

Example 9.22. Let $G := \langle \alpha, \nu \rangle$ be the group generated by the Collatz permutation α and the class shift $\nu : \ n \mapsto n+1$. Further let $\sigma := \nu_{0(2)} \cdot \nu_{1(2)}^{-1}$. We factor σ into the generators of G as described in Method 9.21:

- For $k = 1$, we obtain $\tilde{\sigma} = \nu^{-2} \neq \sigma$.
- For $k = 2$, we obtain $\tilde{\sigma} = \alpha^{-1} \cdot \nu^{-1} \cdot \alpha^{-1} \cdot \nu \cdot \alpha^{-1} \cdot \nu^{-2} \neq \sigma$.
- For $k = 3$, we obtain the same as for $k = 2$.
- For $k = 4$, we obtain $\tilde{\sigma} = \alpha^{-1} \cdot \nu^{-1} \cdot \alpha \cdot \nu^{-1} \cdot \alpha \cdot \nu^{-1} \cdot \alpha \cdot \nu^2 \cdot \alpha^{-1} \neq \sigma$.
- For $k = 5$, we obtain the same as for $k = 4$.
- For $k = 6$, we obtain $\tilde{\sigma} = \alpha^{-1} \cdot \nu^{-1} \cdot \alpha^3 \cdot \nu^{-1} \cdot \alpha^{-1} \cdot \nu^{-1} \cdot \alpha^{-1} \cdot \nu^2 \alpha^{-1} \neq \sigma$.
- For $k = 7$, we obtain $\tilde{\sigma} = \alpha^{-1} \cdot \nu^{-1} \cdot \alpha \cdot \nu^{-2} \cdot \alpha^{-1} \cdot \nu^{-1} \cdot \alpha \cdot \nu^4 = \sigma$.

For $k = 7$ we obtain equality, and thus get the desired (shortest possible!) factorization.

10. Testing for Transitivity

In this section, we describe methods to check whether a given rcwa group acts transitively on $\mathbb{Z}$ or on a given union of finitely many residue classes.

Method 10.1 (Testing an rcwa Group for Transitivity)**.** Let G be an rcwa group, and let $S \subseteq \mathbb{Z}$ be a nonempty union of finitely many residue classes on which G acts. To check whether the action of G on S is transitive, we proceed as follows:

(1) By Algorithm 4.6, compute the support of G. If S is not either a subset of or equal to the support of G, then return *false*.
(2) Look for finite orbits. For this, choose some bounds b_1 and b_2. Then compute all finite orbits which intersect nontrivially with the intersection of $\{-b_1, \ldots, b_1\}$ and S, and whose cardinality is at most b_2. If finite orbits are found, then return *false*.
(3) By Method 7.5, compute the order of G. If G is finite, then return *false*.
(4) Search for an element $\sigma \in G$ and a residue class $r(m) \subseteq S$ such that the restriction of σ to $r(m)$ is given by $n \mapsto n \pm m$. For this, loop over the generators of G, the commutators of generators of G, powers of generators of G, powers of commutators of generators of G and products of $2, 3, 4, \ldots$ generators of G. For any element σ encountered along the way, do the following:
   (a) Put $\tilde{m} := \mathrm{Mod}(\sigma)$, and assume that σ is represented by the reduced coefficient list $(a_{r(\tilde{m})}, b_{r(\tilde{m})}, c_{r(\tilde{m})})_{r(\tilde{m}) \in \mathbb{Z}/\tilde{m}\mathbb{Z}}$.
   (b) Look for residues r such that $a_{r(\tilde{m})} = c_{r(\tilde{m})} = 1$ and $\tilde{m} | b_{r(\tilde{m})}$. For any such r, put $m := b_{r(\tilde{m})}$ and check whether the residue class $r(m)$ is a subset of S. If this is the case for some r, then σ and $r(m)$ satisfy the above conditions.

This search will often be unsuccessful even if the action of G on S is indeed transitive, as there is no guarantee that the group G has a suitable element. If the search was successful, then the cyclic group generated by σ acts transitively on the residue class $r(m)$. If it was unsuccessful, then give up.

(5) Put $\tilde{S} := r(m)$.
(6) For all generators σ_i of G, put $\tilde{S} := \tilde{S} \cup \tilde{S}^{\sigma_i}$. Repeat this until $\tilde{S}$ remains constant. This may be an infinite loop.
(7) If $\tilde{S} = S$, then return *true*. Otherwise return *false*.

Example 10.2. Let G be the group from Example 6.11. We check whether the action of G on $\mathbb{Z}$ is transitive:

- We check whether the support of G is equal to $\mathbb{Z}$ (it is), and whether we find any finite orbits (we don't).
- We compute the order of G, and obtain $|G| = \infty$. Thus we still cannot answer our question.
- We find that $[g, h]|_{2(6)} = \nu_{2(6)}^{-1}$, and put $\tilde{S} := 2(6)$.
- We unite the set $\tilde{S}$ with its images under the generators g and h of G, and obtain $\tilde{S} = 1(3) \cup 2(6)$. We do this for a second time, and obtain $\tilde{S} = \mathbb{Z}$. This establishes the transitivity of the action of G on $\mathbb{Z}$.

Next we describe a method to determine the degree of transitivity of the action of an rcwa group on a union of residue classes, or on the set of positive integers in a union of residue classes. This method usually needs frequent human interaction in order to avoid coefficient explosion, huge memory requirements and infinite loops wherever one finds a way to circumvent these evils at all. For this reason, the description of the method is less explicit than the other method descriptions in this article. First we need to introduce the following terms:

Definition 10.3. Let f be an rcwa mapping, and let m denote its modulus. Assume that the mapping f is represented by the reduced coefficient list $(a_{r(m)}, b_{r(m)}, c_{r(m)})_{r(m) \in \mathbb{Z}/m\mathbb{Z}}$.

We call the union of the residue classes (mod m) for which $c_{r(m)} > a_{r(m)}$ the *set on which f is decreasing*, and use the notation $d(f)$.

Further we call the union of the residue classes (mod m) for which $a_{r(m)} = c_{r(m)} = 1$ and $b_{r(m)} < 0$ the *set on which f shifts down*, and we call the union of those residue classes (mod m) for which $a_{r(m)} = c_{r(m)} = 1$ and $b_{r(m)} > 0$ the *set on which f shifts up*. For these sets, we use the notations $s_d(f)$ and $s_u(f)$, respectively.

We define the *maximal shift* of f by $\max\{|b_{0(m)}|, \ldots, |b_{m-1(m)}|\}$.

Method 10.4 (Outline: Degree of Transitivity of an rcwa Group)**.** Let G be an rcwa group, and let $S \subseteq \mathbb{Z}$ be a union of finitely many residue classes which G acts on. We attempt to determine the degree of transitivity of the action of G on S (or on $S \cap \mathbb{N}_0$) as follows:

(1) If $S \neq \mathbb{Z}$, then proceed as follows:
  (a) By Algorithm 4.7, restrict G to the set S.
  (b) Find an injective rcwa mapping f whose image is S. For this, determine partitions of $\mathbb{Z}$ and S into the same number of residue classes, and construct an rcwa mapping which maps the one to the other class-by-class.
  (c) By Algorithm 5.6, compute the image of G under the induction epimorphism $\tilde{\pi}_f$ associated with f, and put $G := \tilde{\pi}_f(G)$.

  Now we are in the situation that we want to determine the degree of transitivity of the action of G on either $\mathbb{Z}$ or $\mathbb{N}_0$.

(2) Put $F := \emptyset$ and $d := 0$.
(3) Determine a set D of elements of the pointwise stabilizer G_F of F in G such that
   - $\cup_{\sigma \in D}(d(\sigma) \cup s_d(\sigma)) = \mathbb{Z}$, if we want to compute the degree of transitivity of the action of G on $\mathbb{N}_0$, and that in addition
   - $\cup_{\sigma \in D}(d(\sigma) \cup s_u(\sigma)) = \mathbb{Z}$, if we want to compute the degree of transitivity of the action of G on $\mathbb{Z}$.

   This can be done by looping over short products of generators of G, leaving out some 'nasty' ones with large modulus or so. If this fails, then go to Step (6).
(4) Let b be the maximum of the maximal shifts of the rcwa permutations in D, and put $S_0 := \{-b, \ldots, b\} \setminus F$ or $S_0 := \{0, \ldots, b\} \setminus F$, depending on whether we want to determine the degree of transitivity of G on $\mathbb{Z}$ or on $\mathbb{N}_0$.
(5) Check whether all integers in the set S_0 belong to the same orbit under the action of G_F. If this is the case, then put $F := F \cup \{d\}$ and $d := d + 1$, and go to Step (3). If this is not the case, one could return d; however, verifying this tends to be infeasible with presently available methods.
(6) Now the goal is to verify that G is not $d+1$-transitive. For this, look for a positive integer m such that the action of G is not transitive on $d + 1$-tuples (mod m). If such an m is found, then return d. Otherwise return 'at least d'.

Examples for the use of Method 10.4 can be found in the manual of the GAP [3] package RCWA [9]. In one of these examples, it is described how to find out that the group $G := \langle \nu, \tau_{1(2),0(4)} \rangle$ acts 3-transitively, but not 4-transitively on $\mathbb{Z}$.

11. Acknowledgement

I thank Edmund F. Robertson for proofreading.

References

1. Pierre de la Harpe, *Topics in geometric group theory*, Chicago Lectures in Mathematics, 2000. MR 1786869 (2001i:20081)
2. Bettina Eick and Werner Nickel, *Polycyclic – Computation with polycyclic groups; Version 2.2*, 2007, refereed GAP package, published at http://www.gap-system.org/Packages/polycyclic.html.
3. The GAP Group, *GAP – Groups, Algorithms, and Programming; Version 4.4.10*, 2007, http://www.gap-system.org.
4. Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien, *Handbook of computational group theory*, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall / CRC, Boca Raton, FL, 2005. MR 2129747 (2006f:20001)
5. Timothy P. Keller, *Finite cycles of certain periodically linear permutations*, Missouri J. Math. Sci. **11** (1999), no. 3, 152–157.
6. Stefan Kohl, *Restklassenweise affine Gruppen*, Dissertation, Universität Stuttgart, 2005, published at http://deposit.d-nb.de/cgi-bin/dokserv?idn=977164071.
7. ———, *Graph theoretical criteria for the wildness of residue-class-wise affine permutations*, 2006, preprint (short note), available at http://www.cip.mathematik.uni-stuttgart.de/~kohlsn/preprints/graphcrit.pdf.
8. ———, *A simple group generated by involutions interchanging residue classes of the integers*, 2006, preprint, available at http://www.cip.mathematik.uni-stuttgart.de/~kohlsn/preprints/simplegp.pdf.
9. ———, *RCWA - Residue-Class-Wise Affine Groups; Version 2.5.4*, 2007, refereed GAP package, published at http://www.gap-system.org/Packages/rcwa.html.
10. ———, *Wildness of iteration of certain residue-class-wise affine mappings*, Adv. in Appl. Math. **39** (2007), no. 3, 322–328. MR 2352043
11. Jeffrey C. Lagarias, *The 3x+1 problem: An annotated bibliography*, 2007, arxiv.org/abs/math.NT/0309224.
12. Günther J. Wirsching, *The dynamical system generated by the 3n+1 function*, Lecture Notes in Mathematics, no. 1681, Springer-Verlag, 1998.

INSTITUT FÜR GEOMETRIE UND TOPOLOGIE, PFAFFENWALDRING 57, UNIVERSITÄT STUTTGART
70550 STUTTGART, GERMANY
E-mail address: kohl@mathematik.uni-stuttgart.de