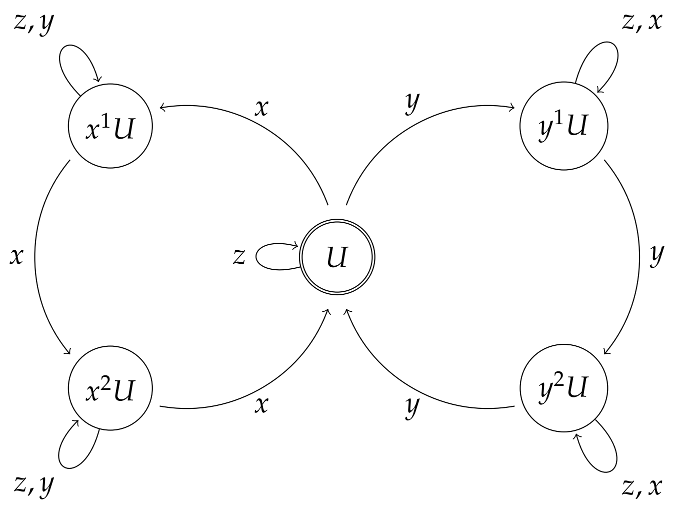
Karlsruher Institut für Technologie
Fakultät für Mathematik

Joachim Breitner

Diploma Thesis

Loop subgroups of F_r and the images of their stabilizer subgroups in $\mathrm{GL}_r(\mathbb{Z})$



Supervisors:
Prof. Dr. Frank Herrlich
Dr. Gabriela Schmithüsen

February 11, 2010

Preface

THE abelianization of the free group over r generators is a group homomorphism $F_r \to \mathbb{Z}^r$. Its kernel is a characteristic subgroup of F_r and thus, this map gives rise to a map between the respective automorphism groups

$$B\colon \operatorname{Aut}(F_r) \to \mathrm{GL}_r(\mathbb{Z}).$$

This homomorphism can also be given explicitly. It maps each automorphism $\gamma \in \operatorname{Aut}(F_r)$ to an integral matrix by counting the number of occurrences of each generator in the image of each generator: If $g_1, \dots, g_r$ are the generators of the free group, this means

$$\gamma \mapsto \left(\#_{g_i}\, \gamma(g_j)\right)_{i,j=1\dots r}.$$

In the study of imprimitive translation surfaces, especially origamis [Sch05], this map is used to obtain Veech groups. Each origami can be associated with a subgroup $U \leq F_2$ of finite index. The image of the stabilizer group of U in $\operatorname{Aut}(F_2)$ under B, intersected with $\mathrm{SL}_2(\mathbb{Z})$, gives the Veech group of the origami. In the case of origamis, an interesting question is which subgroups of $\mathrm{SL}_2(\mathbb{Z})$ occur as Veech groups. There is a positive answer for many congruence subgroups (see [Sch05]), and for all subgroups of the principal congruence group of level 2 (see [EM09, Theorem 1.2]).

The construction of a Veech group can be generalized to subgroups of F_r for a general $r \in \mathbb{N}$ instead of origamis. For rank of 3 or higher, there is a reason to hope that these groups will be easier to understand, as every normal subgroup of $\mathrm{SL}_r(\mathbb{Z})$ besides $\{I\}$ and $\{I, -I\}$ is a congruence group and thus can be thought of as a subgroup of $\mathrm{SL}_r(\mathbb{Z}/l\mathbb{Z})$, where l is the congruence level of the subgroup [Sur03, Theorem 4-4.2].

To investigate this situation, we implemented an algorithm called *CosetProject* that projects a subgroup of $\operatorname{Aut}(F_r)$ to $\mathrm{GL}_r(\mathbb{Z})$ and detects its congruence level. Together with the algorithm implemented in [Fre08], which calculates the stabilizer group of a subgroup of F_r in $\mathrm{Aut}(F_r)$, it can be used to detect the congruence level of a Veech group. The algorithm is described in detail in Chapter 2.

Following these investigations, a certain class of subgroups of F_r arose my interest. I dubbed them *loop subgroups*, due to the appearance of their coset graphs. These are generalizations of the L-origamis, but their analogs to the Veech groups show very different behavior. They are studied in Chapter 3, where the image of their stabilizer group under the map B is calculated. Because of the similar results, the index two subgroups are treated in the same manner in Section 3.10.

I owe sincere thanks to Dr. Gabriela Schmithüsen, under whose guidance this thesis was created. She was always generous with time, advice, regular proofreading, ideas and inspiration.

Moreover, I wish to thank Mareike Schmidtobreick and Pascal Maillard for proofreading the final work.

Contents

List of Algorithms

CHAPTER 1

Fundamentals

THE document at hand does not require very deep mathematical knowledge. Readers familiar with concepts taught in foundations courses – groups, homomorphisms, matrices – should be able to read this thesis with little effort. Some definitions that slightly extend beyond this basic knowledge, such as the group of automorphisms, finitely presented groups and the coset graph, are explained briefly in this chapter. Also, the pseudocode notation used for the algorithms is explained.

1.1 The automorphism group

One of the main objects of interest is $\mathrm{Aut}(F_r)$, the automorphism group of the free group of rank r.

Definition 1 (Automorphism group) Let G be a group. Denote with $\mathrm{Aut}(G)$ the set of automorphisms of G, i.e. the bijective homomorphisms from G to G. This set becomes a group, called the *automorphism group of* G, when equipped with the function concatenation $\circ$ as the group operator.

Remark 1 Because abelian groups are $\mathbb{Z}$-modules, one can identify the automorphisms of the group $\mathbb{Z}^r$ with the group of integral, invertible matrices of rank r:

$$\mathrm{Aut}(\mathbb{Z}^r) = \mathrm{GL}(\mathbb{Z}^r) = \mathrm{GL}_r(\mathbb{Z})$$

Definition 2 (Stabilizer group) For a subgroup $U \leq G$, the *stabilizer subgroup* is defined as the set of automorphisms that map U to U:

$$\mathrm{Stab}_{\mathrm{Aut}(G)}(U) := \{\gamma \in \mathrm{Aut}(G) : \gamma(U) = U\}.$$

Definition 3 A subgroup $U \leq G$ is called *characteristic* if it is left invariant under all automorphisms of U, i.e. $\forall \gamma \in \mathrm{Aut}(G) : \gamma(U) = U$. In other words, $U \leq G$ is called characteristic if $\mathrm{Stab}_{\mathrm{Aut}(G)}(U) = \mathrm{Aut}(G)$.

Remark 2 Let $\varphi \colon G \to G'$ be a surjective group homomorphism such that $\mathrm{Kern}\,\varphi$ is a characteristic subgroup of G. This homomorphism naturally gives rise to a homomorphism $\hat{\varphi} \colon \mathrm{Aut}(G) \to \mathrm{Aut}(G')$, defined by

$$\hat{\varphi}(\gamma)(g') = \varphi(\gamma(g)) \text{ with } g \in \varphi^{-1}(g')$$

for $\gamma \in \mathrm{Aut}(G), g' \in G'$.

This definition is sound: If $g_1, g_2 \in \varphi^{-1}(g')$, then $g_1 g_2^{-1} \in \mathrm{Kern}\,\varphi$. Therefore,

$$\varphi(\gamma(g_1)) \cdot \varphi(\gamma(g_2^{-1})) = \varphi(\gamma(g_1 g_2^{-1})) \in \varphi(\gamma(\mathrm{Kern}\,\varphi)) = \varphi(\mathrm{Kern}\,\varphi) = \{\mathrm{Id}\}$$

and hence $\varphi(\gamma(g_1)) = \varphi(\gamma(g_2))$.

The kernel of the map $F_r \to \mathbb{Z}^r$ mentioned in the preface is, by definition of abelianization, the commutator subgroup $[F_r, F_r]$ of the free group. Every commutator subgroup is characteristic. This justifies the introduction of the map $B \colon \mathrm{Aut}(F_r) \to \mathrm{GL}_r(\mathbb{Z})$.

1.2 The free group

Definition 4 (Free group) The *free group F_r* of rank r with generators $g_1, \ldots, g_r$ is the set of words over the alphabet $\{g_1, \ldots, g_r, g_1^{-1}, \ldots, g_r^{-1}\}$, considering two words equal if they can be reduced to the same word by means of canceling $g_i g_i^{-1}$ and $g_i^{-1} g_i$. The group operation is the concatenation of words.

The empty word is the identity of the group F_r, and in the following denoted by Id.

Example 1 In F_3 with generators g_1, g_2, g_3, the following identities hold:

$$\begin{gathered}(g_1 g_2^{-1} g_3^{-1})^{-1} = g_3 g_2 g_1^{-1} \\ g_1 = g_1 g_2^{-1} g_2 = (g_2^{-1} g_1)^{-1} g_2 \\ (g_1 g_2 g_3^{-1}) \cdot (g_3 g_2^{-1} g_1^{-1}) = \mathrm{Id}.\end{gathered}$$

An equivalent definition of the free group F_r with generators $g_1, \dots, g_r$ is the universal property of the free group:

Remark 3 (Universal property of the free group)
For any group G and map $f\colon \{g_1, \dots, g_r\} \to G$, there is exactly one group homomorphism $\varphi\colon F_r \to G$ such that $\varphi(g_i) = f(g_i)$ for $i = 1, \dots, r$.

A useful consequence of this property is that group homomorphisms $F_r \to G$ can be sufficiently specified by stating the images of the generators of F_r. Naturally, this includes automorphisms of the free group.

Example 2 For $i \in \{1, \dots, r\}$, the map

$$\#_{g_i}\colon F_r \to \mathbb{Z}$$

is defined by its images of the generators of F_r:

$$\#_{g_i}(g_k) := \begin{cases} 1, & k = i \\ 0, & k \neq i. \end{cases}$$

It is the generator-counting map which calculates the number of occurrences of the generator g_i in a word $w \in F_r$, counting negative powers of g_i negatively.

Remark 4 Every finitely generated group $G = \langle h_1, \dots, h_r \rangle$ is isomorphic to a quotient group of the free group of rank r.

PROOF By the universal property of the free group, the mapping $g_i \mapsto h_i$ defines a group homomorphism $\varphi\colon F_r \to G$. This homomorphism is surjective, so by the fundamental homomorphism theorem, we have

$$F_r/\ker\varphi \cong G.$$

■

Definition 5 (Finitely presented group) Let F_r be the free group of rank r and generators $g_1, \dots, g_r$, and $R = \{r_1, \dots, r_s\} \subset F_r$ be a finite subset of F_r, called the relations. Then $\langle g_1, \dots, g_r \mid r_1, \dots, r_s \rangle$ is called a *finite presentation* of the group

$$F_r/\langle\langle R \rangle\rangle,$$

where

$$\langle\langle R \rangle\rangle := \bigcap_{\substack{N \lhd F_r \\ R \subset N}} N$$

is the normal closure of the set R.

Any group that has a finite presentation is called a *finitely presented group*.

Usually, the symbols used as generators in the presentation are interchangeably used for the group element they represent.

The free groups of finite rank are trivially finitely presented. Many other important groups – $\mathrm{Aut}(F_r)$, the matrix groups $\mathrm{GL}_r(\mathbb{Z})$ and $\mathrm{SL}_r(\mathbb{Z})$ – are also finitely presented. This is fortunate, as these groups can then be investigated computationally.

Example 3 The modular group is a finitely presented group:

$$\mathrm{PSL}_2(\mathbb{Z}) = {}^{\mathrm{SL}_2(\mathbb{Z})}\!\big/_{\{I,-I\}} = \langle S, T \mid S^2, (ST)^3 \rangle.$$

The symbols S and T represent the matrices $\left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$ respectively $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$.

Example 4 [Sur03, Theorem 4-3.2] The integral special linear group $\mathrm{SL}_r(\mathbb{Z})$, $r \geq 3$, is generated by the $r \cdot (r-1)$ symbols X_{ij}, $i, j \in \{1, \dots, r\}$, $i \neq j$, subject to the relations

$$\begin{gathered}
[X_{ij}, X_{jk}] = X_{ik} \text{ for } i \neq k \\
[X_{ij}, X_{kl}] = \mathrm{Id} \text{ for } j \neq k,\ i \neq l \\
(X_{12} X_{21}^{-1} X_{12})^4 = \mathrm{Id},
\end{gathered}$$

where $[\alpha, \beta] := \alpha\beta\alpha^{-1}\beta^{-1}$ denotes the commutator bracket. The symbol X_{ij} represents the elementary matrix X_{ij}, i.e. the $r \times r$-matrix with ones on the diagonal, one additional one in the i-th row and j-th column and zeroes everywhere else.

Any element of a finitely generated group G with r generators can be represented by a word in the generators and their inverses, which the computer can treat as a list of numbers from the set $\{-r, \dots, -1, 1, \dots, r\}$, where the positive numbers correspond to the generators of the group and the negative numbers to their inverses. In the following, I will at times use a word in the generators of G as an element of G, an element of F_r and as a list of generators.

1.3 The coset graph

The most natural way to specify a subgroup U of a group G, i.e. a subset that is itself again a group, is by a defining property. For example, a point-stabilizer subgroup of the symmetric group can be written as

$$\mathrm{Stab}_{S_n}(1) = \{\pi \in S_n \mid \pi(1) = 1\}.$$

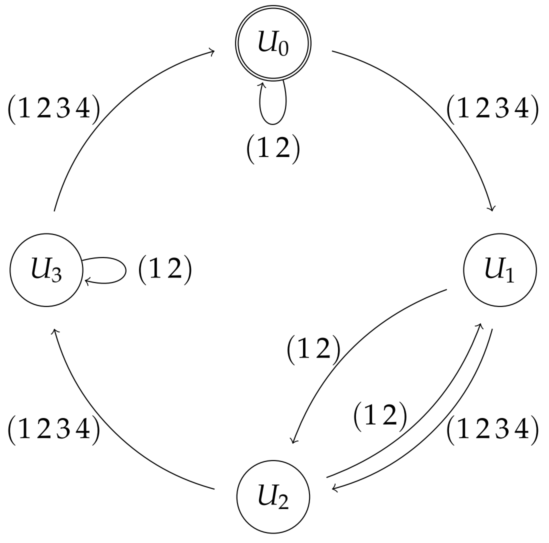


Figure 1: The coset graph of $S_3 \leq S_4$

Another way of specifying subgroups is by a set of subgroup generators: If $H \subset G$ is a subset of the group, then

$$\langle H \rangle := \bigcap_{\substack{U \leq G \\ H \subset U}} U$$

is the smallest subgroup containing H and is called the subgroup generated by H.

While these methods are often convenient to define subgroups, they are not necessarily useful when working with subgroups, especially when applying computational methods to them. Therefore, where possible, it can be useful to describe a subgroup by its coset graph. In this document, we will work with left cosets and will not always mention this fact. Of course, it is possible to work with right cosets in an analogous manner.

Definition 6 (Coset graph) The *coset graph* of a subgroup U of a finitely generated group G with generators $g_1, \ldots, g_r$ is the directed multigraph with the left cosets $\{wU \mid w \in G\}$ of U as nodes and, for each coset wU and generator g_i, one edge from wU to $g_i wU$, labeled g_i.

When convenient, one can also add edges labeled by the inverses of the generators.

Example 5 The coset graph of S_3 as a subgroup of S_4 with respect to the generators $(1\,2)$ and $(1\,2\,3\,4)$ is depicted in Figure 1 on the preceding page. The nodes are numbered for readability. The corresponding cosets are:

$$\begin{aligned} U_0 &:= S_3 \\ U_1 &:= (1\,2\,3\,4) \cdot S_3 \\ U_2 &:= (1\,3)(2\,4) \cdot S_3 \\ U_3 &:= (1\,4\,3\,2) \cdot S_3 \end{aligned}$$

Remark 5 Let $U \leq G$.

1. The coset graph of U is a connected, regular graph of degree $2 \cdot r$, as every node has r outgoing and r ingoing edges.

2. The number of nodes in the coset graph of U is $[G : U]$. Therefore, the graph is finite if and only if U has finite index.

   We can see in Figure 1 on the previous page that $[S_4 : S_3] = 4$

3. The node labels are not essential and can be omitted as long as one remembers which node, called the *origin*, represents the subgroup itself. Using another node to represent the subgroup gives the coset graph of a conjugate of the subgroup. The coset graph of every subgroup conjugate can be obtained this way.

   We can see in Figure 1 on the preceding page that S_3 is not normal in S_4: The coset graph of S_3 looks different when using U_1 as the origin, therefore $S_3 \neq (1\,2\,3\,4)S_3(1\,2\,3\,4)^{-1}$.

4. The Cayley graph of G is the coset graph of the trivial subgroup $\{\mathrm{Id}\} \leq G$.

In the computational part of this thesis, we will often trace elements of the group, i.e. a word in the generators, in the graph. This is best explained by example. Tracing the permutation $(1\,2)^{-1}(1\,2\,3\,4)(1\,2)$ in Figure 1 on the previous page, starting at the node U_0, ends up at the node U_2: Looking at the generators in the word from the right, we first have $(1\,2)$. The edge with that label from U_0 is reflexive, so we stay in node U_0. The next generator is $(1\,2\,3\,4)$, so we follow the edge to U_1. The final generator is an inverse, so we have to traverse the edge from U_2 to U_1 labeled $(1\,2)$ in the opposite direction.

This way, once we have a coset graph for a subgroup $U \leq G$, we can for a word $w \in G$ and a coset vU determine the coset wvU. This gives us a procedure to determine the subgroup membership problem $w \in U$ for a word $w \in G$: We have $w \in G$ if and only if tracing that word from the coset U ends up in the coset U.

In the following chapter, the coset graph is represented by a coset table, which has the adjacency lists of the nodes as rows. It has one row for each coset vU, one column for each generator g_i and entries referring to the number of the row of $g_i vU$. As an optimization measure, columns for the inverses of generators are also added.

1.4 Pseudocode notation

The next chapter contains a series of algorithms, written in pseudocode. The syntax is mostly self-explanatory.

The main body of each algorithm extends from the declaration of input and output to the **return** statement, possible functions are declared afterward. The statement **forall** $x \in S$ executes the body of the statement once for each element of S, with the variable x bound to that value.

Variables are assigned using the $\leftarrow$ operator, which corresponds to := in Pascal-like programming languages. The statement $a \leftarrow b \leftarrow c$ is short for $b \leftarrow c$; $a \leftarrow b$. The equals-to symbol $=$ is exclusively used for comparisons.

Array access is zero-based, i.e. $A[0]$ is the first entry of the array and $A[\texttt{len}(A) - 1]$ is the last entry. Tables are written as arrays of arrays, so $C[0][0]$ is the entry in the first row and first column, while $C[3][2]$ is the entry in the forth row and third column. Therefore, $\texttt{len}(C)$ denotes the number of rows in the table. For convenience, at times we say that a table has columns "labeled by the elements of the set X". In that case, X is finite and there is an implicit numbering of these elements. The expression $C[0][x]$ stands for $C[0][i]$ where i is the implicit number of x.

CHAPTER 2

The CosetProject algorithm

As outlined in the preface, we are interested in applying the map B, as defined in the preface, to the stabilizer subgroup of a finite index subgroup $U \leq F_r$.

For the first step, we can use Algorithm 6 in [Fre08]. Its input is a finite index subgroup $U \leq F_r$, given by a list of Nielsen-reduced generators. The algorithm calculates the stabilizer group $\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$ of the subgroup and returns

- a list of generators of the stabilizer group,
- a list of representatives of the cosets of the stabilizer group or
- the coset graph of the automorphism group.

[Fre08] also contains variants of this algorithm optimized for efficiency.

This algorithm needs a set of generators of $\mathrm{Aut}(F_r)$. Freidinger suggests that the generating set τ_1, σ_{12}, $\sigma_{23}, \ldots, \sigma_{r-1\,r}$, η (cf. [AFV08]) is beneficial for the run time efficiency of her algorithms. See Table 1 on the following page for the definitions of these generators. Note that each of these are self-inverse. An automorphism as returned by her algorithm is represented by their image of the generators of F_r as well as a decomposition in these generators of $\mathrm{Aut}(F_r)$.

$$\tau_i(g_k) = \begin{cases} g_i^{-1}, & k = i \\ g_k, & k \neq i \end{cases} \quad \sigma_{ij}(g_k) = \begin{cases} g_j, & k = i \\ g_i, & k = j \\ g_k, & k \neq i, j \end{cases} \quad \eta(g_k) = \begin{cases} g_2^{-1} g_1, & k = 1 \\ g_2^{-1}, & k = 2 \\ g_k, & k > 2 \end{cases}$$

Table 1: A system of generators of $\mathrm{Aut}(F_r)$

2.1 First approach: Todd-Coxeter

Given $\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$, the task is to calculate $\Gamma := B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)) \leq \mathrm{GL}_r(\mathbb{Z})$ and $\Gamma' := \Gamma \cap \mathrm{SL}_r(\mathbb{Z}) \leq \mathrm{SL}_r(\mathbb{Z})$. For the latter group, generators and relations are given in [Sur03, Theorem 4-3.2] (see Example 4 on page 10). This allows the use of the very general Todd-Coxeter algorithm, which calculates a coset graph of a subgroup of finite index in any finitely generated group. [TC36].

The variant of the algorithms described here is very close to the original algorithm from 1936, which was obviously designed to be carried out by hand. A serious implementation as a computer program would involve a number of optimizations, such as constructing the relation tables on demand or special-casing involutory constructors. [Lee63]

2.1.1 Description of the algorithm

The pseudocode listing Algorithm 1 outlines the Todd-Coxeter algorithm. The input to the algorithm is provided as a list of relations R of the group G, written as words in the generators of the group, and a list of generators H of the subgroup U, also written as words in the generators of the whole group.

It works with three types of tables:

1. A left coset table C, with the columns labeled by the generators and their inverses and the rows labeled by the coset numbers.

2. For each relation w in the presentation of the group G, a relation table R_w with columns labeled by the letters of the word w and rows labeled by the coset numbers.

Algorithm 1: Todd-Coxeter algorithm

```
Input: Group $G$ given by generators $(g_i)_{i=1,\dots,r}$ and relations $R$
Input: Subgroup $U \leq G$ given by the set of generators $H$
Output: A coset table of $U \leq G$

Let $X$ be the set of generators and their inverses.
$C \leftarrow$ empty table with $2r$ columns labeled by the elements of $X$ and one row
forall $w \in R$ do
    $R_w \leftarrow$ empty table with len($w$)+1 columns and one row
    $R_w[0][0] \leftarrow R_w[0][\texttt{len}(w)] \leftarrow 0$
forall $h \in H$ do
    $U_h \leftarrow$ empty table with len($h$)+1 columns and one row
    $U_h[0][0] \leftarrow U_h[0][\texttt{len}(h)] \leftarrow 0$

while there is a table $T_w \in \{U_h, h \in H\} \cup \{R_w, w \in R\}$ with an empty spot do
    Let $i$ be a row that is not complete and $j$ the column of the first empty spot.
    $k \leftarrow T_w[i][j-1]$
    $g \leftarrow w[\texttt{len}(w) - (j+1)]$
    if $C[k][g]$ is not set then
        $l \leftarrow$len(C)
        Extend each of the tables $C$ and $R_w$, $w \in R$ by one row, the $l$-th.
        $C[k][g] \leftarrow l$
        $C[l][g^{-1}] \leftarrow k$
        forall $w \in R$ do $R_w[l][0] \leftarrow R_w[l][\texttt{len}(w)] \leftarrow l$
    $T[i][j] \leftarrow l \leftarrow C[k][g]$
    if $T[i][j+1]$ is set then
        $m \leftarrow T[i][j+1]$
        $\bar{g} \leftarrow w[\texttt{len}(w) - j]$
        if $C[l][\bar{g}]$ is not set then $C[l][\bar{g}] \leftarrow m$
        if $C[m][\bar{g}^{-1}]$ is not set then $C[m][\bar{g}^{-1}] \leftarrow l$
        if $C[l][\bar{g}] \neq m$ then Coincidence($C[l][\bar{g}], m$)
        if $C[m][\bar{g}^{-1}] \neq l$ then Coincidence($C[m][\bar{g}^{-1}], l$)
return $C$

function Coincidence($i, j$) begin
    if $j < i$ then Coincidence($j, i$)
    else
        forall table entries $T[k][l]$ do
            if $T[k][l] = j$ then $T[k][l] \leftarrow i$
        forall $g \in X$ do
            if $C[j][g]$ is set then
                if $C[i][g]$ is not set then $C[i][g] \leftarrow C[j][g]$
                if $C[i][g] \neq C[j][g]$ then Coincidence($C[i][g], C[j][g]$)
end
```

3. For each generator h of the subgroup, a generator table U_h with columns labeled by letters of the word h and exactly one row, labeled by 0.

The entries of these tables are cosets of the subgroup U, represented by natural numbers. The coset $\mathrm{Id} \cdot U$ has the number 0, other numbers are added as required. Let U_i denote the coset represented by i. For convenience, an unlabeled zeroth column is added to the latter two tables, whose entries contain the coset number of the respective row.

The semantics of the coset table is as follows: If the entry $C[i][g]$ in the i-th row and the column labeled by the generator or inverse of a generator g is set, then multiplying the generator g with the coset represented by i gives the coset with number $C[i][g]$:

$$U_{C[i][g]} = g \cdot U_i.$$

Storing this information about both generators and inverses of generators is redundant, but convenient.

For the other tables the semantics is as follows: Let T_w be the table corresponding to the relation or generator w. If both $T[i][j]$ and $T[i][j+1]$ are set, then multiplying the generator $w[\texttt{len}(w) - (j+1)]$, which is the j-th last letter in the word w, with the coset represented by $T[i][j]$ gives the coset with number $T[i][j+1]$:

$$U_{T[i][j+1]} = w[\texttt{len}(w) - (j+1)] \cdot U_{T[i][j]}$$

We have to reverse the words because we are working with left cosets. When working with right cosets, this is not necessary.

Missing entries in these tables will now be filled sequentially, adding new coset numbers as required and merging coset numbers when known to represent the same coset, until the table C is filled completely and fully determines the subgroup U. The other tables can then be discarded and C is returned.

Upon initialization, the tables are generated and an initial row 0, representing the coset $\mathrm{Id} \cdot U$, is added. For the relation and generator tables, the zeroth column of the row is set to 0. Also, the last column is set to zero, as $w \cdot U = U$ for $w \in R$ and $h \cdot U = U$ for $h \in H$.

As long as there is an empty entry in the relation or generator tables, it is filled. If the necessary information is already contained in the coset table, it is used. Otherwise, a new coset number is created. For this, the coset table and the relation tables are extended by an additional row. Both the first and last entry in the new row in the relation tables is set to the number of the new coset, as $w \cdot U_i = U_i$ for a relation

$w \in R$ and a coset U_i. The coset table is amended with this information. Note that we do not extend the generator tables, as in general $h \cdot U_i \neq U_i$ for a generator $h \in H$ of the subgroup. If we did this, we would treat them just like group relations and effectively calculating the coset graph of $\langle\langle H \rangle\rangle$, the normal closure of U, instead of $\langle H \rangle$.

If we complete a row of a relation or generator table this way, since the last column of the row is already known, we find a new bit of information: Let w be the relation or generator of the table, $\bar{g}$ the first letter of the word w and i the number of the row. If we have just put k in the penultimate entry in the row, then we know that $\bar{g}$ multiplied with the coset represented by k gives the coset represented by i:

$$U_i = \bar{g} \cdot U_k$$

This is called a *deduction*. If the corresponding entry $C[k][\bar{g}]$ (resp. the entry for the inverse edge in the graph) in the coset table is still empty, we set it to i. If the entry is already set to l and differs from i, we know that the numbers i and l represent the same coset. This is called a *coincidence* and is handled by the function `Coincidence`.

This function replaces every occurrence of the larger coset number in the tables with the lower coset numbers. The information in the respective rows in the coset table is compared. If it disagrees, a new coincidence was found, and the function is called recursively. The row of the larger coset number is deleted from the tables (without changing the indexes of the following rows).

Eventually, all entries in the tables (ignoring deleted rows) have been filled. Therefore, C contains the full coset table of U and is returned. In practice, one might want to compress the table by moving the rows "up" to fill the empty spots left over by deleted lines before returning it.

2.1.2 An example application of the Todd-Coxeter algorithm

We want to derive the coset graph of $S_3 \leq S_4$, as seen in Figure 1 on page 11. The representation

$$\langle B, C \mid B^2,\ C^4,\ (BC)^3,\ (BC^{-1}BC)^3 \rangle$$

is given in [Cox36], along with presentations for other symmetric groups and alternating groups. The symbols B and C represent the permutations $(1\,2)$ and $(1\,2\,3\,4)$ respectively. In our case of S_4, this can be reduced further to

$$\langle B, C \mid B^2,\ C^4,\ (BC)^3 \rangle$$

C	B	B^{-1}	C	C^{-1}
0				

R_{B^2}	B	B
0		0

R_{C^4}	C	C	C	C
0				0

$R_{(BC)^3}$	C	B	C	B	C	B
0						0

U_B	B
0	0

$U_{C^{-1}BC^2}$	C	C	B	C^{-1}
0				0

Table 2: Todd-Coxeter tables for $S_3 \leq S_4$ after initialization

as $(BC^{-1}BC)^3 \in \langle\langle B^2, C^4, (BC)^3 \rangle\rangle$ (this can be verified by some lengthy manual calculations[1] or by using a computer algebra system like [GAP08]). The subgroup generators $(1\,2)$ and $(1\,2\,3)$ of S_3 are represented by the words B and $C^{-1}BC^2$.

For the Todd-Coxeter algorithm, we create six tables: The coset table C, the three relations tables R_{B^2}, R_{C^4} and $R_{(BC)^3}$ and the subgroup generator tables U_B and $U_{C^{-1}BC^2}$, initialized as shown in Table 2.

We have a subgroup generator of length one, so the deduction $B \cdot S_3 = S_3$ follows immediately. This special-casing of length-one relations and subgroup generators was omitted in the pseudocode listing in Algorithm 1. This deduction fills the entry $C[0][B] = 0$ and its inverse direction, $C[0][B^{-1}] = 0$. We then proceed by filling the table $U_{C^{-1}BC^2}$. Because we have no information about $C[0][C]$, we introduce a new coset number 1, and likewise coset 2 and 3. We now are in the situation of Table 3.

The deduction from table $U_{C^{-1}BC^2}$ tells us that we have to set $C[3][C^{-1}] = 0$, which is no problem, and $C[0][C] = 3$. But this table entry is already set to 1, so these coset numbers actually represent the same coset. A call to `Coincidence`$(1, 3)$ copies the value of $C[3][B^{-1}]$, which is 2, to $C[1][B^{-1}]$, replaces 3 by 1 everywhere and marks coset 3 as deleted. We are now in the situation of Table 4.

We now continue to fill the relation tables. The value $R_{B^2}[0][0]$ is already known from $C[0][B] = 0$. For $R_{B^2}[1][0]$ we introduce a new coset number 4, which is already merged with coset number 2 when we fill the next row in the table R_{B^2}. We continue filling R_{C^4}, which leads to the introduction of coset number 5. To fill $R_{B^2}[5][0]$, we introduce coset number 6, which allows us to fill R_{C^4} completely with known information. Turning to the remaining table, $T_{(CB)^3}$, we fill the first row. For the last entry, we require a new coset 7. Now, the coset tables look as in Table 5.

The deduction would yield $C[7][B] = 0$ and thus $C[0][B^{-1}] = 7$, which is in conflict with the existing data. Therefore, `Coincidence`$(0, 7)$ is called and rows 0 and 7 are merged. As they disagree in the last column, `Coincidence`$(5, 6)$ is called, leaving

[1] $(BC^{-1}BC)^3 = B^2(B^2(((B^2((B^2(C^4)^B(((BC)^3)^{-1})^B)^{CB^{-1}}B^2C^4)^{C^{-1}B}(((BC)^3)^{-1})^B)^{CB^{-1}}B^2C^4)^{C^{-1}})^B$ $(((BC)^3)^{-1})^B)^{CB}(B^2(B^2(B^2(((BC)^3)^{-1})^B)^{CB})^{CB})^{C^{-1}}$ where $\beta^\alpha := \alpha^{-1}\beta\alpha$.

C	B	B^{-1}	C	C^{-1}
0	0	0	1	
1			2	0
2	3			1
3		2		

R_{B^2}	B	B
0		0
1		1
2		2
3		3

R_{C^4}	C	C	C	C
0				0
1				1
2				2
3				3

$R_{(BC)^3}$	C	B	C	B	C	B
0						0
1						1
2						2
3						3

U_B	B
0	0

$U_{C^{-1}BC^2}$	C	C	B	C^{-1}
0	1	2	3	0

Table 3: Todd-Coxeter tables for $S_3 \leq S_4$ before the first coincidence

C	B	B^{-1}	C	C^{-1}
0	0	0	1	
1		2	2	0
2	1			1
~~3~~		~~2~~		~~0~~

R_{B^2}	B	B
0		0
1		1
2		2
~~3~~		~~3~~

R_{C^4}	C	C	C	C
0				0
1				1
2				2
~~3~~				~~3~~

$R_{(BC)^3}$	C	B	C	B	C	B
0						0
1						1
2						2
~~3~~						~~3~~

U_B	B
0	0

$U_{C^{-1}BC^2}$	C	C	B	C^{-1}
0	1	2	1	0

Table 4: Todd-Coxeter tables for $S_3 \leq S_4$ after the first coincidence

C	B	B^{-1}	C	C^{-1}
0	0	0	1	5
1	2	2	2	0
2	1	1	5	1
~~3~~		~~2~~		~~0~~
~~4~~	~~1~~	~~1~~		
5	6	6	0	2
6	5	5	7	
7				6

R_{B^2}	B	B
0	0	0
1	2	1
2	1	2
~~3~~		~~3~~
~~4~~		~~4~~
5	6	5
6	5	6
7		7

R_{C^4}	C	C	C	C
0	1	2	5	0
1	2	5	0	1
2	5	0	1	2
~~3~~				~~3~~
~~4~~				~~4~~
5				5
6				6
7				7

$R_{(BC)^3}$	C	B	C	B	C	B
0	1	2	5	6	7	0
1						1
2						2
~~3~~						~~3~~
~~4~~						~~4~~
5						5
6						6
7						7

U_B	B
0	0

$U_{C^{-1}BC^2}$	C	C	B	C^{-1}
0	1	2	1	0

Table 5: Todd-Coxeter tables for $S_3 \leq S_4$ before the final deduction

only coset numbers 0, 1, 2 and 5 in place. Now, the coset table is complete. Filling the other tables with this information, as shown in Table 6, does not lead to any new deductions, so the algorithm terminates. It is easily verified that the coset table describes the coset graph as depicted in Figure 1 on page 11.

2.1.3 Analysis of the algorithm

Todd and Coxeter did not explicitly specify the order in which the tables have to be filled. If the algorithm ensures that for any coset added, its row in the coset table is eventually filled, and if the index of U in G is finite then the algorithm is guaranteed to terminate [Ser97].

A formal proof can be found in [Men64]. Mendelsohn adds additional, redundant relations to R, such that they form what Mendelsohn calls an "algorithmic set": A set of relations where each generator is both the first letter of a relation and the last letter of a relation. This way, the coset table rows are guaranteed to fill eventually.

Unfortunately, a run time complexity analysis as usual of this algorithm will fail: Assume that the run time were bounded by a computable function of the size of the input. Note that the algorithm does not terminate when applied to a subgroup of infinite index. Given an arbitrary subgroup, we could use the algorithm to determine whether its index is infinite: If the algorithm runs longer than the given bound, this is the case. But this is a problem known to be unsolvable [BBN59, Theorem 7], therefore no such upper bound can exist [Sim94].

2.2 Second approach: CosetProject

Our implementation of the Todd-Coxeter algorithm, which was relatively naïve, turned out to be too slow for our purposes. Using it here is like using a sledgehammer to crack a nut: We already have the coset graph of $\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$ available, but did not use this information. This observation led to a faster algorithm for the calculation of a coset graph of $B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$.

This algorithm, which is given in pseudocode in Algorithm 2, works in the general setting of projecting a subgroup $U \leq G$ with a given coset table C onto a quotient group G' given by additional relations R such that $G' = G/\langle\langle R\rangle\rangle$. Let $\pi : G \to G'$ denote the projection map. Note that π maps cosets of U to cosets of $U' := \pi(U)$.

C	B	B^{-1}	C	C^{-1}
0	0	0	1	5
1	2	2	2	0
2	1	1	5	1
~~3~~		~~2~~		~~0~~
~~4~~	~~1~~	~~1~~		
5	5	5	0	2
~~6~~	~~5~~	~~5~~	~~0~~	
~~7~~	~~0~~			~~6~~

R_{B^2}	B	B
0	0	0
1	2	1
2	1	2
~~3~~		~~3~~
~~4~~		~~4~~
5	5	5
~~6~~	~~5~~	~~6~~
~~7~~		~~7~~

R_{C^4}	C	C	C	C
0	1	2	5	0
1	2	5	0	1
2	5	0	1	2
~~3~~				~~3~~
~~4~~				~~4~~
5	0	1	2	5
~~6~~				~~6~~
~~7~~				~~7~~

$R_{(BC)^3}$	C	B	C	B	C	B
0	1	2	5	5	0	0
1	2	1	2	1	2	1
2	5	5	0	0	1	2
~~3~~						~~3~~
~~4~~						~~4~~
5	0	0	1	2	5	5
~~6~~						~~6~~
~~7~~						~~7~~

U_B	B
0	0

$U_{C^{-1}BC^2}$	C	C	B	C^{-1}
0	1	2	1	0

Table 6: Todd-Coxeter tables for $S_3 \leq S_4$ at the end of the algorithm

Algorithm 2: CosetProject algorithm

```
Input: Generators $(g_i)_{i=1,\dots,r}$ of a group $G$
Input: The coset table $C$ of a subgroup $U \leq G$
Input: Relations $R$ describing the quotient group $G' := G/\langle\langle R \rangle\rangle$
Output: The coset table of the image $U'$ of $U$ in $G'$

Let $X$ be the set of generators and their inverses.
forall cosets numbers $i \in \{0, \dots, \mathtt{len}(C) - 1\}$ do
    if row $i$ in $C$ is not marked as merged then
        forall relations $w \in R$ do
            Trace $w$ in $C$ starting with coset $i$, let $j$ be the resulting coset.
            if $i \neq j$ then Coincidence$(i, j)$
Return $C$.

function Coincidence$(i, j)$ begin
    if $j < i$ then Coincidence$(j, i)$
    else
        forall table entries $C[k][g]$ do
            if $C[k][g] = j$ then $C[k][g] \leftarrow i$
        forall $g \in X$ do
            if $C[i][g] \neq C[j][g]$ then Coincidence$(C[i][g], C[j][g])$
        Mark row $j$ in $C$ as merged.
end
```

During the process of the algorithm, the coset table of U is modified to become the coset table of U'. Speaking in terms of the coset graph, edges are bent to point to another coset of of U that maps to the same coset of U' as the previous target of the edge.

To do so, the algorithm iterates through all cosets U_i of $U \leq G$ and relations w in the presentation of G' and traces the word w in the coset graph of U. Because $\pi(w) = \mathrm{Id}$ in G' we have $\pi(w) \cdot \bar{v} \cdot \pi(U) = \bar{v} \cdot \pi(U)$. So if we end up at a different coset number, these numbers represent the same coset of U'. We replace the higher number everywhere in the tables with the lower number and mark the row of the higher number as merged, so that we can skip it in the main loop. The entries of the two rows are compared, and if they disagree, we have found the next pair of cosets of U that map to the same coset of U'.

2.2.1 Proof of termination

The main loop iterates through the cosets and relations, both of which are finite lists. Also the loops in the function `Coincidence` loop over finite lists. For each coset i the function `Coincidence` is called at most once with i as the second argument, as after the first loop in the body of the function, the table does not contain an i anywhere. Therefore, only finitely many calls to `Coincidence` occur, and the algorithm is guaranteed to terminate.

2.2.2 Proof of correctness

Define the relation $i \sim j$ on the set of cosets of U to indicate that `Coincidence`(i, j) has been called during the execution of the algorithm. Define the relation $\approx$ as the equivalence relation generated by $\sim$, i.e. its transitive, reflexive, symmetrical hull. Let cosets of U that are projected to the same coset of U' in G' be related by $\equiv$, i.e. $i \equiv j \iff \pi(U_i) = \pi(U_j)$.

Note that at the beginning of the algorithm, C is the coset table for U in G, so $j = C[i][g] \iff U_j = g \cdot U_i$. As the entries of C are changed, this property can be violated. Instead, we will show that at any point in the algorithm, the following invariant holds:

$$j = C[i][g] \implies \pi(U_j) = \pi(U_{C[i][g]}) = \pi(g \cdot U_i).$$

The invariant obviously holds at the beginning of the algorithm.

A change to the table C only happens after a call to `Coincidence`(i, j) with $i < j$, which replaces the value j by i. To preserve the invariant, we want $\pi(U_i) = \pi(U_j)$ to hold, i.e. $i \equiv j$. The function is called on two occasions:

Case 1: The main loop calls `Coincidence`(i, j) if there is a relation $w \in R$ such that tracing the word w in the graph from node i ends up in node j. This implies, as the invariant holds, $\pi(U_j) = \pi(w \cdot U_i)$. Since $\pi(w) = \mathrm{Id}$, we have $\pi(U_i) = \pi(U_j)$ and hence $i \equiv j$.

Case 2: $i = C[i'][g]$, $j = C[j'][g]$ and `Coincidence`(i, j) is called from the body of a call to `Coincidence`(i', j'). By induction on the nesting level of the calls to `Coincidence`, with the base case being Case 1 from above, we know $i' \equiv j'$. $i = C[i'][g]$ indicates $\pi(g \cdot U_{i'}) = (U_i)$, and $j = C[j'][g]$ indicates $\pi(g \cdot U_{j'}) = (U_j)$. Hence we have $\pi(U_i) = \pi(g \cdot U_{i'}) = \pi(g) \cdot \pi(U_{i'}) = \pi(g) \cdot \pi(U_{j'}) = \pi(g \cdot U_{j'}) = \pi(U_j)$ and therefore $i \equiv j$.

This proves that the invariant is never violated and also shows that the implication "$i \sim j \implies i \equiv j$" holds. Because $\approx$ is contained in every equivalence relation that contains $\sim$, "$i \approx j \implies i \equiv j$" follows.

On the other hand, "$i \equiv j \implies i \approx j$" holds as well: We have $\pi(U_i) = \pi(U_j)$ if and only if there is a word $w \in \langle\langle R \rangle\rangle$ with $U_j = w \cdot U_i$. This means that w is a product of conjugates of elements of R. Because $\approx$ is transitive, it is sufficient to assume that $w = vrv^{-1}$ is a conjugate of $r \in R$. Let k be the coset reached by tracing v^{-1} from i in the coset graph, i.e. $U_k = v^{-1} \cdot U_i$, and likewise let l denote the coset with $U_l = r \cdot U_k = r \cdot v^{-1} \cdot U_i$. At some point in the main loop of the algorithm, the relation r is traced from the coset k, `Coincidence`(l, k) is called and $k \approx l$ holds. We prove $k \approx l$ by induction over the length of v. If v is actually the empty word, $k = i$ and $l = j$ and we are done.

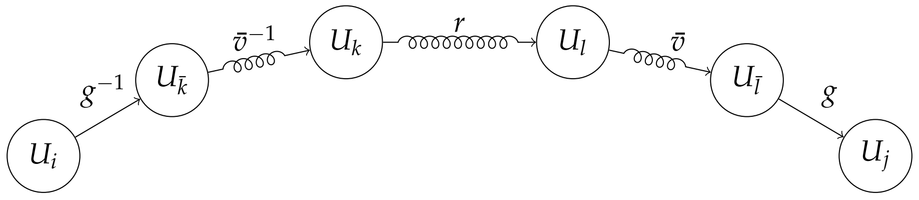


Figure 2: Denotations of cosets in the proof for $i \equiv j \implies i \approx j$

Otherwise, let $g \in X$ be the first letter in the word v, i.e. $v = g \cdot \bar{v}$, let $U_{\bar{k}} = g^{-1} \cdot U_i$ and $U_{\bar{l}} = \bar{v} \cdot r \cdot v^{-1} \cdot U_i$. The situation is illustrated in Figure 2. By the induction hypothesis, we have $\bar{k} \approx \bar{l}$. In the second loop of the function `Coincidence`$(\bar{k}, \bar{l})$ the entries of the coset table for the cosets $\bar{k}$ and $\bar{l}$ are compared. The original entries of $C[\bar{k}][g]$ resp. $C[\bar{l}][g]$ as they were before the algorithm runs are i resp. j, as $g \cdot U_{\bar{k}} = g \cdot g^{-1} \cdot U_i = U_i$ and $g \cdot U_{\bar{l}} = g \cdot \bar{v} \cdot r \cdot v^{-1} \cdot U_i = w \cdot U_i = U_j$. Table entries are only altered if they were arguments to `Coincidence` once, hence $C[\bar{k}][g] \approx i$ and $C[\bar{l}][g] \approx j$. Either we already have $C[\bar{k}][g] = C[\bar{l}][g]$ and $i \approx C[\bar{k}][g] = C[\bar{l}][g] \approx j$ gives $i \approx j$. Or $C[\bar{k}][g] \neq C[\bar{l}][g]$, in which case the algorithm calls `Coincidence`$(C[\bar{k}][g], C[\bar{l}][g])$, and hence $i \approx C[\bar{k}][g] \sim C[\bar{l}][g] \approx j$ gives $i \approx j$.

Every call to `Coincidence` removes the larger of the arguments completely from the table entries. Therefore, when the algorithm ends, there is only one representative, the smallest one, left for each equivalence class of $\equiv$. The rows corresponding to the other cosets are marked as merged. Especially, row 0 is not marked as merged. Let $U_i' := \pi(U_i)$ for an i whose row is not marked as merged. The invariant $\pi(U_{C[i][g]}) = \pi(g \cdot U_i)$ implies $U'_{C[i][g]} = \pi(g) \cdot U'$, so C actually is the coset graph of $U' = \pi(U)$, as required.

2.3 Run time analysis

In the following analysis I assume that the index of U in G is smaller than the largest integer representable by a machine word and that the table entries are one such machine word. Therefore, operations involving these numbers run in $O(1)$ and the table entries occupy $O(1)$ pieces of storage.

The space complexity of the algorithm is very good, as it mostly just changes the table it obtains as the input. This table has one row per coset of $U \leq G$ and two columns per generator of G, thus a total number of $[G : U] \cdot 2 \cdot r$ entries. The marking of the rows can be done by writing invalid entries to the table rows, otherwise an additional $[G : U]$ bits of storage is required. The stack for the recursive function `Coincidence` requires constant storage for the arguments and is at most $[G : U] - 2$ levels deep.

As to the time complexity of the algorithm: The outer main loop is executed at most $[G : U]$ times, and less if some rows are marked as merged. Inside the loop, each relation $w \in R$ is traced in the table, which takes $|w|$ steps of constant time. Each call to `Coincidence` iterates through the whole table, requiring $[G : U] \cdot 2 \cdot r$ steps of constant time each. Additionally, it compares the $2 \cdot 2 \cdot r$ entries of the two rows, possibly calling `Coincidence`. The total number of calls to `Coincidence`, no matter from where the function is called, is $[G:U]-[G':U']$, as each call to `Coincidence` marks exactly one row as merged. Therefore, the total run time complexity is

$$O\Big([G:U]\cdot\sum_{w\in R}|w|+([G:U]-[G':U'])\cdot([G:U]\cdot 2\cdot r+2\cdot 2\cdot r)\Big)$$

which can be simplified in the O-calculus to

$$O\Big([G:U]\cdot\sum_{w\in R}|w|+[G:U]^2\cdot r\Big).$$

2.3.1 Improved data structures

As a possible optimization, one can skip updating the whole table inside the function `Coincidence` if one maintains an array A of $[G:U]$ entries, initially mapping each coset number to itself. This variant is provided in pseudocode in Algorithm 3 on the following page. The array A or, more precisely, the function `AccA` used to access values from A maps each coset to the equivalent coset with the lowest coset number found so far. At the end of the algorithm, it will map each coset to the lowest representative of its equivalence class with respect to $\equiv$.

Therefore, the main loop needs to call `Coincidence` only if two related cosets are not already found to be related, i.e. they do not map to the same coset via A.

Each call to `Coincidence`(i,j) with $i<j$ sets the array entry $A[j]$ to i. The arguments to `Coincidence` are always lowest representatives, i.e. from the image of `AccA`. Instead of directly comparing the values $C[i][g]$ with $C[j][g]$ the algorithm compares their images under the function `AccA`. This does not change the semantics of the algorithm, but the run time cost for each call to `Coincidence` is reduced to $O(r)$.

Immediately after a call to `AccA`(i), $A[i]$ is set to the result of `AccA`$[i]$, so direct array access is allowed then. When accessing the array A at position j, the function `AccA` first checks if either $A[j]=j$ or `AccA`$(A[j])=A[j]$. If that is not the case, it sets $A[j]$ to the result of the recursive lookup $A[A[j]]$. Note that the condition of the if-clause calls `AccA`$(A[i])$ recursively and thus, $A[A[i]]$ can be accessed directly in the body of the if-clause. This way, `AccA`(j) always returns the number that the unoptimized algorithm would have put in place of j in the table, and each entry is kept up-to-date to avoid unnecessary recursive lookups.

We want to count the calls to `AccA` from other parts of the program as constant-time operations. This is correct unless `AccA` calls itself recursively. Therefore, we have to

Algorithm 3: CosetProject algorithm (variant 1)

```
Input: Generators $(g_i)_{i=1,\dots,r}$ of a group $G$
Input: The coset table $C$ of a subgroup $U \leq G$
Input: Relations $R$ describing the quotient group $G' := G/\langle\langle R \rangle\rangle$
Output: The coset table of the image $U'$ of $U$ in $G'$

Let $X$ be the set of generators of $G$ and their inverses.
Let $A$ be an array of $\texttt{len}(C)$ elements.
forall $i \in \{0, \dots, \texttt{len}(A) - 1\}$ do $A[i] \leftarrow i$

forall cosets numbers $i \in \{0, \dots, \texttt{len}(C) - 1\}$ do
    if row $i$ in $C$ is not marked as merged then
        forall relations $w \in R$ do
            Trace $w$ in $C$ starting with coset $i$, let $j$ be the resulting coset.
            if $\texttt{AccA}(i) \neq \texttt{AccA}(j)$ then $\texttt{Coincidence}(A[i], A[j])$

return $\texttt{map}(\texttt{AccA}, C)$.

function Coincidence $(i,j)$ begin
    if $j < i$ then Coincidence $(j,i)$
    else
        $A[j] \leftarrow i$
        forall $g \in X$ do
            if $\texttt{AccA}(C[i][g]) \neq \texttt{AccA}(C[j][g])$ then
                $\texttt{Coincidence}(A[C[i][g]], A[C[j][g]])$
        Mark row $j$ in $C$ as merged.
end

function $\texttt{AccA}(i)$ begin
    if $i \neq A[i] \wedge A[i] \neq \texttt{AccA}(A[i])$ then
        $A[i] \leftarrow A[A[i]]$
    Return $A[i]$.
end
```

report the time spent inside recursive calls to `AccA` separately. Each of the $[G:U]$ entries of A is changed at most $[G:U]/[G':U']$ times, as that many cosets of U are mapped to the same coset of U'. Hence, up to $[G:U] \cdot [G:U]/[G':U']$ recursive calls to `AccA` may occur.

This gives a total run time of

$$O\Big([G:U] \cdot \sum_{w \in R} |w| + [G:U] \cdot r + [G:U] \cdot \frac{[G:U]}{[G':U']}\Big).$$

2.4 Preimage calculation

The CosetProject algorithm in the present form calculates a coset table of the image subgroup $U' \leq G'$. This also gives the index $[G':U']$ and can be used to efficiently decide for an element of $w \in G'$, given as a word in the generators, whether it is in U'. In some applications, this is not enough and we would also want to know a preimage of w under π that lies in U. In general, using the same word is not sufficient: In the coset graph of $U \leq G$, this might trace to a coset other than U, which also happens to be mapped to U' by π.

The algorithm can be extended in a straight forward manner to remember, for each coset U_i of U, a word $w_i \in G$ such that $U_{A[i]} = w_i \cdot U_i$. Then, given a word $w \in U'$ that traces from U to U_i in the coset graph of U, $w_i \cdot w$ is a preimage of w that lies in U.

The modified algorithm, based on the optimized variant Algorithm 3, is outlined in Algorithm 4 on the next page. Besides the array W, a new parameter to the function `Coincidence` is introduced.

The correctness of the extended algorithm follows from the following invariant involving the array W:

$$U_{A[i]} = W[i] \cdot U_i$$

The invariant holds after the initialization of the arrays. A and W are always changed together: Either in `Coincidence`, or in `AccA`. In the body of the if-clause in `AccA`, let A' and W' denote the arrays after the modification. We have, by using the invariant twice,

$$U_{A'[i]} = U_{A[A[i]]} = W[A[i]] \cdot U_{A[i]} = W[A[i]] \cdot W[i] \cdot U_i = W'[i] \cdot U_i$$

Algorithm 4: CosetProject algorithm (variant 2, preimage calculation)

```
Input: Generators $(g_i)_{i=1,\dots,r}$ of a group $G$
Input: The coset table $C$ of a subgroup $U \leq G$
Input: Relations $R$ describing the quotient group $G' := G/\langle\langle R\rangle\rangle$
Output: The coset table of the image $U'$ of $U$ in $G'$
Output: An array $A$ of len($C$) indexes such that $\pi(U_i) = \pi(U_{A[i]})$
Output: An array $W$ of len($C$) words such that $U_{A[i]} = W[i] \cdot U_i$

Let $X$ be the set of generators and their inverses.
Let $A$ be an array of len($C$) elements.
forall $i \in \{0, \dots, \text{len}(A) - 1\}$ do $A[i] \leftarrow i$
Let $W$ be an array of len($C$) elements, each initialized to the empty word.

forall cosets numbers $i \in \{0, \dots, \text{len}(C) - 1\}$ do
    if row i in C is not marked as merged then
        forall relations $w \in R$ do
            Trace $w$ in C starting with coset $i$, let $j$ be the resulting coset.
            if AccA($i$) $\neq$ AccA($j$) then
                Coincidence($A[i], A[j], W[i] \cdot w^{-1} \cdot W[j]^{-1}$)

Return map(AccA, $C$), $A$ and $W$.

function Coincidence($i, j, w$) begin
    if $j < i$ then Coincidence($j, i, w^{-1}$)
    else
        $A[j] \leftarrow i$
        $W[j] \leftarrow w$
        forall $g \in X$ do
            if AccA($C[i][g]$) $\neq$ AccA($C[j][g]$) then
                $w' \leftarrow W[C[i][g]] \cdot g \cdot w \cdot g^{-1} \cdot W[C[j][g]]^{-1}$
                Coincidence($A[C[i][g]], A[C[j][g]], w'$)
        Mark row $j$ in C as merged.
end

function AccA($i$) begin
    if $i \neq A[i] \wedge A[i] \neq$ AccA($A[i]$) then
        $W[i] \leftarrow W[A[i]] \cdot W[i]$
        $A[i] \leftarrow A[A[i]]$
    Return $A[i]$.
end
```

and thus the invariant is preserved.

When the function `Coincidence` is called from the main loop, j was chosen by $U_j = w \cdot U_i$, hence $U_i = w^{-1} \cdot U_j$. Let $i' = A[i]$, $j' = A[j]$ and $w' = W[i] \cdot w^{-1} \cdot W[j]^{-1}$ be the arguments passed to `Coincidence`. By invoking the invariant, we find that they fulfill the condition $U_{i'} = w' \cdot U_{j'}$:

$$\begin{aligned} w' \cdot U_{j'} &= (W[i] \cdot w^{-1} \cdot W[j]^{-1}) \cdot U_{A[j]} \\ &= W[i] \cdot w^{-1} \cdot W[j]^{-1} \cdot W[j] \cdot U_j \\ &= W[i] \cdot w^{-1} \cdot U_j \\ &= W[i] \cdot U_i \\ &= U_{A[i]} \\ &= U_{i'} \end{aligned}$$

In a call to `Coincidence`(i, j, w), assume $U_i = w \cdot U_j$. This means that the invariant is preserved after updating the array A and W. When `Coincidence` is now called recursively, a similar calculation shows that the arguments again fulfill the assumed condition:

$$\begin{aligned} w' \cdot U_{A[C[j][g]]} &= W[C[i][g]] \cdot g \cdot w \cdot g^{-1} \cdot W[C[j][g]]^{-1} \cdot U_{A[C[j][g]} \\ &= W[C[i][g]] \cdot g \cdot w \cdot g^{-1} \cdot W[C[j][g]]^{-1} \cdot W[C[j][g]] \cdot U_{C[j][g]} \\ &= W[C[i][g]] \cdot g \cdot w \cdot g^{-1} \cdot U_{C[j][g]} \\ &= W[C[i][g]] \cdot g \cdot w \cdot g^{-1} \cdot g \cdot U_j \\ &= W[C[i][g]] \cdot g \cdot w \cdot U_j \\ &= W[C[i][g]] \cdot g \cdot U_i \\ &= W[C[i][g]] \cdot U_{C[i][g]} \\ &= U_{A[C[i][g]]} \end{aligned}$$

By induction, this shows that the condition $U_i = w \cdot U_j$ holds for all calls to the function `Coincidence`(i, j, w). Hence, the invariant is preserved inside calls to `Coincidence` as well and the array contains the desired information.

2.5 Application to the automorphism group

In the preceding sections, we assumed that the preimage group G and the image group G' were given with the same generators and G' could be considered a quotient group of G by the normal closure of the additional relations. In the application described at the beginning of this chapter, this is unfortunately not the case: Subgroups of $\mathrm{Aut}(F_r)$ are given by their coset graph with respect to the generators $\tau_1, \sigma_{12}, \dots, \sigma_{r-1\,r}, \eta$ (see Table 1 on page 16), while we would like to use the generators X_{ij}, $i \neq j$, when referring to subgroups of $\mathrm{SL}_r(\mathbb{Z})$.

Additionally we want, for a subgroup $U \leq \mathrm{Aut}(F_r)$, to calculate $B(U) \cap \mathrm{SL}_r(\mathbb{Z})$, so we have to implement the intersection of a subgroup, given by its coset graph, with an index two subgroup.

Depending on whether we utilize the Todd-Coxeter algorithm or the CosetProject algorithm, different steps have to be taken. These are outlined here and explained below.

When using the Todd-Coxeter algorithm, the steps are:

1. Calculate generators of $U \cap \mathrm{Aut}^+(F_r)$ via Lemma 1.
2. Map generators of the subset $U \cap \mathrm{Aut}^+(F_r)$ through B.
3. Calculate the coset graph using the Todd-Coxeter algorithm.

When using the CosetProject algorithms, the steps are

1. Using Algorithm 5, create a coset graph of $U \cap \mathrm{Aut}^+(F_r) \leq \mathrm{Aut}^+(F_r)$.
2. Apply the CosetProject algorithm to the coset graph of $U \cap \mathrm{Aut}^+(F_r)$ to obtain the coset graph of $B(U) \cap \mathrm{SL}_r(\mathbb{Z}) \leq \mathrm{SL}_r(\mathbb{Z})$.
3. Rewrite the coset graph in terms of the desired generators using Algorithm 5.

2.5.1 Index two subgroup intersection

Lemma 1 *Let $S \leq G$ be a subgroup with index 2. Let $U \leq G$ be a subgroup generated by $h_1, \dots, h_n \in G$ with $U \not\leq S$. Without loss of generality, assume that there is an $m \in \{1, \dots, n\}$ such that $h_1, \dots, h_m \in G \setminus S$ and $h_{m+1}, \dots, h_n \in S$.*

The subgroup $U \cap S \leq G$ is generated by the set

$$\{h_i^2,\, h_i \cdot h_1^{-1} \mid i = 1, \dots, m\} \cup \{h_i,\, h_1 \cdot h_i \cdot h_1^{-1} \mid i = m+1, \dots, n\}.$$

PROOF Let w be a word in the generators $h_1, \dots, h_n$ such that $w \in U \cap S$ and without loss of generality assume that $w \neq \mathrm{Id}$ and that any proper prefix of w is in $U \setminus S$ (otherwise split w after the prefix and apply the proof to both parts).

If w has length 1, it is already one of the generators $h_{m+1}, \dots, h_n$ or their inverses.

If w has length 2, it is a product $h_i^{\varepsilon_i} \cdot h_j^{\varepsilon_j}$ with $\varepsilon_i, \varepsilon_j \in \{-1, 1\}$ and $i, j \in \{1, \dots, m\}$. If $\varepsilon_i = \varepsilon_j = 1$, w is generated by the given generators by

$$w = h_i \cdot h_j = h_i \cdot h_1^{-1} \cdot (h_j \cdot h_1^{-1})^{-1} \cdot h_j^2.$$

The other cases are obtained by multiplying with $(h_i^2)^{-1}$ from the left respectively with $(h_j^2)^{-1}$ from the right.

If the length of w exceeds 2, assume that the proposition holds for shorter words. The first letter of w is one of $h_1, \dots, h_m$ or their inverses, and the second letter is one of $h_{m+1}, \dots, h_n$ or their inverses, because no proper prefix of w is in $U \setminus S$. So $w = h_i^{\varepsilon_i} \cdot h_k^{\varepsilon_k} \cdot w'$ for some $\varepsilon_i, \varepsilon_j \in \{-1, 1\}$ and $w' \in G$. By writing

$$h_i \cdot h_k^{\varepsilon_k} \cdot w' = (h_i \cdot h_1^{-1}) \cdot (h_1 \cdot h_k \cdot h_1^{-1})^{\varepsilon_k} \cdot h_1 \cdot w'$$

and prepending $(h_i^2)^{-1}$ if $\varepsilon_i = -1$ the problem is reduced to the shorter word $h_1 \cdot w'$ and due to the induction hypothesis, w can be written as a product of the alleged generators. ■

If one works with the Todd-Coxeter algorithm, subgroups are initially represented by a generating set. So the generators $h_1, \dots, h_n$ of $U \leq \mathrm{Aut}(F_r)$ can be mapped through B to obtain a generating set $B(h_1), \dots, B(h_n)$ of $B(U) \leq \mathrm{GL}_r(\mathbb{Z})$. Using Lemma 1, we obtain a generating set of $B(U) \cap \mathrm{SL}_r(\mathbb{Z})$, which can be fed to the Todd-Coxeter algorithm to obtain the coset graph of $B(U) \cap \mathrm{SL}_r(\mathbb{Z}) \leq \mathrm{SL}_r(\mathbb{Z})$.

For the CosetProject algorithm it is convenient to first intersect U with $\mathrm{Aut}^+(F_r)$, which is the preimage of $\mathrm{SL}_r(\mathbb{Z})$ under B, and then rewrite the coset graph of $U \cap \mathrm{Aut}^+(F_r)$ using preimages of the generators $X_{ij}, i \neq j$ (see Example 4 on page 10).

A set of generators of $\mathrm{Aut}^+(F_r)$ is, by Lemma 1,

$$\{\tau_1^2, \tau_1 \cdot \tau_1^{-1}\} \cup \{\sigma_{i\,(i+1)}^2, \sigma_{i\,(i+1)} \cdot \tau_1^{-1} \mid i \in \{1, \dots, r-1\}\} \cup \{\eta^2, \eta \cdot \tau_1^{-1}\},$$

which can be simplified to the r generators

$$\{\sigma_{i\,(i+1)} \cdot \tau_1 \mid i \in \{1, \dots, r-1\}\} \cup \{\eta \cdot \tau_1\},$$

as each generator is involutory.

$$
\begin{aligned}
B(\sigma_{12} \cdot \tau_1) &= \begin{pmatrix} 0 & 1 & \\ -1 & 0 & \\ & & 1 \end{pmatrix} \\
&= X_{21}^{-1} \cdot X_{12} \cdot X_{21}^{-1}
\end{aligned}
$$

$$
\begin{aligned}
B(\sigma_{i\,i+1} \cdot \tau_1) &= \begin{pmatrix} -1 & & \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & & \\ & 0 & 1 \\ & -1 & 0 \end{pmatrix} \quad (\text{for } 1 < i < r) \\
&= (X_{i+1\,1}^{-1} \cdot X_{1\,i+1} \cdot X_{i+1\,1}^{-2} \cdot X_{1\,i+1} \cdot X_{i+1\,1}^{-1}) \cdot (X_{i+1\,i}^{-1} \cdot X_{i\,i+1} \cdot X_{i+1\,i}^{-1})
\end{aligned}
$$

$$
\begin{aligned}
B(\eta \cdot \tau_1) &= \begin{pmatrix} -1 & 0 & \\ 1 & -1 & \\ & & 1 \end{pmatrix} \\
&= X_{21}^{-2} \cdot X_{12} \cdot X_{21}^{-2} \cdot X_{12} \cdot X_{21}^{-1}
\end{aligned}
$$

Table 7: The map B in terms of generators

2.5.2 The map B in terms of generators

For the second step when using Todd-Coxeter, we apply the map B to automorphisms given as words in $(\sigma_{12} \cdot \tau_1), \dots, (\sigma_{r-1\,r} \cdot \tau_1), (\eta \cdot \tau_1)$ and expect results in the generators X_{ij}, $i \neq j$. The map-defining generator images can be found by experimenting with the Sury generators, and are given in Table 7. The matrices are those in the case of $r = 3$ and serve exemplification purposes.

No such explicit mapping is required when going the CosetProject path, as the mapping step does not change the used generators. The transformation happens in the invocation of Algorithm 5, using the equations from Section 2.5.4.

2.5.3 Subgroup intersection and coset graphs

To intersect the subgroup $U \leq \mathrm{Aut}(F_r)$, represented by its coset graph with regard to the generators $\tau_1, \sigma_{12}, \dots, \sigma_{(r-1)\,r}, \eta$, with $\mathrm{Aut}^+(F_r)$, one can use Algorithm 5. Given the coset graph of U and the generators of $\mathrm{Aut}^+(F_r)$, it does a breadth-first search to re-assemble the coset graph of $U \cap \mathrm{Aut}^+(F_r) \leq \mathrm{Aut}^+(F_r)$ with respect to the correct generators. The resulting coset table might be sparse and could be compressed by moving the rows "up" to fill the empty spots left over by unused lines before returning it.

Algorithm 5: Subgroup intersection algorithm

```
Input: Generators $(g_i)_{i=1,\dots,r}$ of a group $G$
Input: The coset table $C$ of a subgroup $U \leq G$
Input: Generators $(h_i)_{i=1,\dots,s}$ of a subgroup $S \leq G$, as words in the $g_i$
Output: The coset table of the intersection $U \cap S \leq S$

Let $Y$ be the set of generators of $S$ and their inverses.
Let $C'$ be an empty coset table with $|C|$ rows and columns labeled by the
elements of $Y$.
Let $q$ be a first-in-first-out queue, initialized with $(0)$.
Let $d$ be a set, initialized with $\{0\}$.
while $q$ is not empty do
    $i \leftarrow$ pop($q$)
    forall $h \in Y$ do
        Trace $h$ in $C$ starting with coset $i$, let $j$ be the resulting coset.
        $C'[i][h] \leftarrow j$
        if $j \notin d$ then
            push($q, j$)
            $d \leftarrow d \cup \{j\}$
return $C'$.
```

2.5.4 Generator rewriting

We have a presentation of $\mathrm{SL}_r(\mathbb{Z})$ in terms of the generators X_{ij}, $i \neq j$, and relations between these generators [Sur03, Theorem 4-3.2]. To turn this into a presentation with regard to the generators

$$\{B(\sigma_{i\,(i+1)} \cdot \tau_1) \mid i \in \{1, \dots, r-1\}\} \cup \{B(\eta \cdot \tau_1)\},$$

we have to rewrite the given relations as products of these generators. To do so, it is sufficient to replace each X_{ij} by a product of these generators.

To find these products, we first write them as products of images of τ_1, $\sigma_{12}, \dots,$ $\sigma_{r-1\,r}, \eta$ under B as follows: As the base case, we have

$$X_{21} = B(\tau_2 \cdot \eta) = B(\sigma_{12} \cdot \tau_1 \cdot \sigma_{12} \cdot \eta)$$

and

$$X_{12} = B(\sigma_{12}) \cdot X_{21} \cdot B(\sigma_{12}).$$

An elementary matrix X_{ij} with $i > j$, can then be written as

$$X_{ij} = B\big(\sigma_{j-1\,j} \cdots \sigma_{12} \cdot \sigma_{i-1\,i} \cdots \sigma_{23}\big) \cdot X_{21} \cdot B\big(\sigma_{23} \cdots \sigma_{i-1\,i} \cdot \sigma_{12} \cdots \sigma_{j-1\,j}\big)$$

and an elementary X_{ij} with $i < j$ can be written as

$$X_{ij} = B\big(\sigma_{i-1\,i} \cdots \sigma_{12} \cdot \sigma_{j-1\,j} \cdots \sigma_{23}\big) \cdot X_{12} \cdot B\big(\sigma_{23} \cdots \sigma_{j-1\,j} \cdot \sigma_{12} \cdots \sigma_{i-1\,i}\big).$$

To turn these products of images of $\tau_1, \sigma_{12}, \dots, \sigma_{r-1\,r}, \eta$ under B into products of images of $(\sigma_{12} \cdot \tau_1), \dots, (\sigma_{r-1\,r} \cdot \tau_1), (\eta \cdot \tau_1)$ under B as required, one can proceed as in the proof of Lemma 1. Since these generators of $\mathrm{Aut}(F_r)$ are involutory, this boils down to replace pairs of generators according to the following identities, where g_1 and g_2 stand for one of $\sigma_{12}, \dots, \sigma_{r-1\,r}, \eta$:

$$\begin{aligned}
\tau_1 \cdot \tau_1 &= \mathrm{Id} \\
g_1 \cdot \tau_1 &= (g_1 \cdot \tau_1) \\
\tau_1 \cdot g_1 &= (g_1 \cdot \tau_1)^{-1} \\
g_1 \cdot g_2 &= (g_1 \cdot \tau_1) \cdot (g_2 \cdot \tau_1)^{-1}.
\end{aligned}$$

Applying the CosetProject algorithm to the coset graph of $U \cap \mathrm{Aut}^+(F_r) \leq \mathrm{Aut}^+(F_r)$ with the relations rewritten in terms of these generators gives a coset graph of $B(U) \cap \mathrm{SL}_r(\mathbb{Z}) \leq \mathrm{SL}_r(\mathbb{Z})$. As a last step, Algorithm 5 can be used to transform this coset graph into a coset graph with regard to the generators X_{ij}. In this invocation of the algorithm, the group G is $\mathrm{SL}_r(\mathbb{Z})$ with generators $B(\sigma_{12} \cdot \tau_1), \dots, B(\sigma_{r-1\,r} \cdot \tau_1), B(\eta \cdot \tau_1)$, the subgroup S is the full group $\mathrm{SL}_r(\mathbb{Z})$ and the subgroup generators $(h_i)_{i=1,\dots,s}$ are the elementary matrices X_{ij}, $i \neq j$, written in terms of the group generators as explained above.

2.6 Congruence level

To detect the congruence level of a subgroup $U \leq \mathrm{SL}_r(\mathbb{Z})$, given its coset graph, we use the fact that the principal congruence subgroup of level l is the normal subgroup generated by X_{12}^l: [Sur03, Proof of Theorem 4-4.2]

$$\Gamma_l = \langle\langle X_{12}^l \rangle\rangle.$$

Therefore, to check $\Gamma_l \leq U$, it is sufficient to check whether $X_{12}^l \in wUw^{-1}$ for all $w \in U$. As the coset graphs of conjugates of U are obtained from the coset graph of

U by taking another node as the origin (Remark 5), we trace the word X_{12}^l from each coset U_i of U and check if we end up at U_i. Trying this procedure for $l = 1, 2, \dots$ until we succeed, we calculate the congruence level of U. This procedure is given in pseudocode in Algorithm 6.

Algorithm 6: Congruence level detection

```
Input: The coset table $C$ of a subgroup $U \leq \mathrm{SL}_r(\mathbb{Z})$
Output: The congruence level $l$ of $U$

$l \leftarrow 1$
while true do
    $ok \leftarrow 1$
    forall $i \in \{0, \dots, \mathtt{len}(C)\}$ do
        Trace $X_{12}^l$ in $C$ starting with coset $i$, let $j$ be the resulting coset.
        if $j \neq i$ then $ok \leftarrow 0$
    if $ok = 1$ then
        return $l$
    else
        $l \leftarrow l + 1$
```

2.7 Results

The algorithm from the previous section was used in combination with the algorithm found in [Fre08] to calculate the congruence levels of $B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)) \cap \mathrm{SL}_r(\mathbb{Z})$ for all subgroups $U \leq F_r$ with a given index i, for approachable values of r and i. The results are printed in Table 8 on the next page. Unfortunately, both the number of subgroups and the index of $\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$ grow very fast, so not many samples were obtained. In the case of the index 4 subgroups of F_3 and the index 3 subgroups of F_5, the program was stopped before handling all subgroups.

Although only low congruence levels are found in this sample, any congruence level is reached (see Section 3.9).

Eventually, working on this problem and inspecting the results gave the ideas and hints that led to the definition of loop subgroups and the results presented in the next chapter.

r	$[F_r : U]$	$[\mathrm{Aut}(F_r) : \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)]$	$[\mathrm{SL}_r(\mathbb{Z}) : B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))]$	congruence level of $B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$
3	1	1	1	1
	2	7	7	2
	3	84	7	2
		13	13	3
	4	7	7	2
		28	28	4
		260	13	3
		168	42	2
		1680	7	2
4	1	1	1	1
	2	15	15	2
	3	585	15	2
		40	40	3
5	1	1	1	1
	2	31	31	2
	3	3720	31	2
		121	121	3

Table 8: Some occurring congruence levels

CHAPTER 3

The loop subgroups

In this chapter, we will study *loop subgroups* of the free group F_r for $r \geq 3$. This class of subgroups stands out because the image of its stabilizer group in $\mathrm{GL}_r(\mathbb{Z})$ is fully calculable. Loop subgroups are quite particular among the general subgroups, but there are legitimate reasons to assume that the results for loop subgroups can be generalized, as discussed in the last section.

3.1 Definition of loop subgroups

Let F_r be the free group with generators $g_1 = x, g_2 = y, g_3 = z, g_4, \ldots, g_r$. The $s_1/\ldots/s_r$ loop subgroup U, $s_i \in \mathbb{N}$, is the subgroup with the distinct left cosets

$$N := \{U,\ g_1^1U, \ldots, g_1^{s_1-1}U,\ g_2^1U, \ldots, g_2^{s_2-1}U,\ \ldots,\ g_r^1U, \ldots, g_r^{s_r-1}U\}$$

and the following edges in its left coset graph

$$\begin{aligned}
U &\xrightarrow{g_i} g_i^1U && \forall i = 1, \ldots, r, \\
g_i^kU &\xrightarrow{g_i} g_i^{k+1}U && \forall i = 1, \ldots, r,\ k = 1, \ldots, s_i - 2, \\
g_i^{s_i-1}U &\xrightarrow{g_i} U && \forall i = 1, \ldots, r, \\
g_j^kU &\xrightarrow{g_i} g_j^kU && \forall i, j = 1, \ldots, r,\ j \neq i,\ k = 1, \ldots, s_j - 1.
\end{aligned}$$

The sequence of nodes $U, g_i^1 U, \dots, g_i^{s_i-1} U, U$ is called a *loop*. s_i is the *length* of the loop. A loop is called *odd* (resp. *even*) if its length is odd (resp. even). A loop of length 1 is called a *looplet*[1].

The reason for this nomenclature becomes evident if we draw the coset graph of a loop subgroup:

Example 6 The coset graph of the 3/3/1 loop subgroup of F_3, a subgroup with only odd loops and one looplet, is shown in Figure 3.

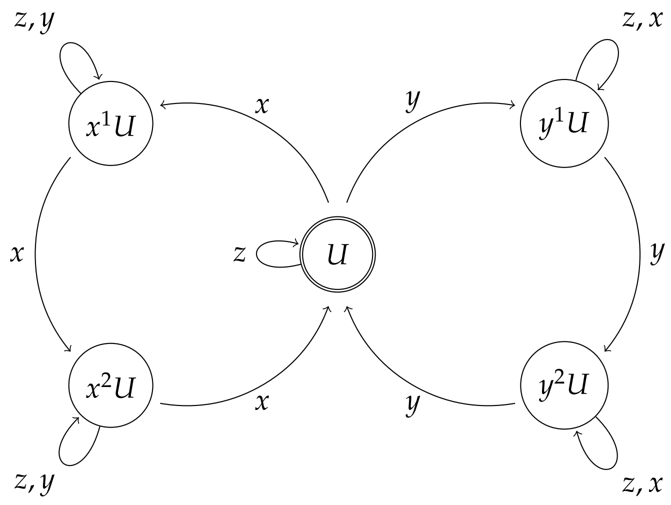


Figure 3: The left coset graph of the 3/3/1 loop subgroup of F_3.

The $s_1/\dots/s_r$ loop subgroup of F_r is generated by the the following set of words in F_r:

$$\{g_i^{s_i},\ i = 1,\dots,r\} \cup \{g_i^{-k} g_j g_i^k,\ i,j = 1,\dots,r,\ i \neq j,\ k = 1,\dots,s_i - 1.\}.$$

This set actually is a basis of U as a free group, as the number of generators is

$$\begin{aligned} r + \sum_{i=1}^{r}(r-1)\cdot(s_i - 1) &= r - 1 + 1 + (r-1)\sum_{i=1}^{r}(s_i - 1) \\ &= (r-1)(1 + \sum_{i=1}^{r}(s_i - 1)) + 1 \\ &= (r-1)[F_r : U] + 1, \end{aligned}$$

which is the number of basis elements of a subgroup of F_r with this index.

[1]The German language allows to build diminutive forms of almost all nouns by appending the suffix *-chen*. I take the liberty to do the same in English, as it makes the text easier and more pleasant to read than if I had named them *small loops*.

3.2 Preparations

Before we now start investigating loop subgroups, some preparational definitions and calculations are due.

3.2.1 Permutations of cosets

In what follows it will often be handy to consider how elements of F_r act on the coset graph of a subgroup U. To make this explicit, let $\mathrm{Sym}(N)$ be the symmetric group on the set N of left cosets of U, and define the group homomorphism

$$\pi\colon F_r \to \mathrm{Sym}(N),$$

which assigns to the word $w \in F_r$ the permutation $\pi(w)$ which maps vU to wvU:

$$\pi(w)(vU) := wvU.$$

We can find out a few things about a word $w \in F_r$ by looking at its permutation in $\mathrm{Sym}(N)$. We have

$$\pi(w)(U) = U \iff w \in U.$$

Furthermore, we have

$$\pi(w) = \mathrm{Id} \iff w \in \mathrm{NT}(U) := \bigcap_{v \in F_r} v^{-1}Uv.$$

To bring automorphisms into this picture, we first note that any $\gamma \in \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$ maps left cosets of U to left cosets of U:

$$\gamma(w)U = \gamma(v)U \iff \gamma(v^{-1}w) \in U \iff v^{-1}w \in U \iff wU = vU$$

Therefore, we can also assign the whole automorphism γ a permutation in $\mathrm{Sym}(N)$ which we denote, by abuse of notation, with $\pi(\gamma)$:

$$\pi(\gamma)(vU) := \gamma(v)U.$$

Note that $\pi(\gamma)^{-1}(vU) = \gamma^{-1}(v)U$. This allows us to formulate the next statement:

Lemma 2 *For $U \leq F_r$, $\gamma \in \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$ and $w \in F_r$, $\pi(\gamma(w))$ is conjugate to $\pi(w)$.*

Proof Using the notation introduced above, for any $v \in F_r$ we have

$$\begin{aligned}\pi(\gamma(w))(vU) &= \gamma(w)vU \\ &= \gamma(w \cdot \gamma^{-1}(v))U \\ &= \pi(\gamma)\big(w \cdot \gamma^{-1}(v)U\big) \\ &= \pi(\gamma)\big(w \cdot \pi(\gamma)^{-1}(vU)\big) \\ &= \pi(\gamma)\Big(\pi(w)\big(\pi(\gamma)^{-1}(vU)\big)\Big) \\ &= \big(\pi(\gamma) \circ \pi(w) \circ \pi(\gamma)^{-1}\big)(vU)\end{aligned}$$

thus $\pi(\gamma(w)) = \pi(\gamma) \circ \pi(w) \circ \pi(\gamma)^{-1}$. ■

Remark 6 A map $\gamma \colon F_r \to F_r$ defined by

$$\gamma(g_j) = \begin{cases} wg_i, & \text{if } j = i \\ g_j, & \text{if } j \neq i \end{cases}$$

is an automorphism if the generator g_i does not occur as a letter in the word w, i.e. $w \in \langle g_1, \dots, g_{i-1}, g_{i+1}, \dots, g_r \rangle$. It stabilizes the subgroup U if $w \in \mathrm{NT}(U)$, as then $\pi(w) = \mathrm{Id}$ holds, implying $\pi(\gamma(v)) = \pi(v)$ and thus $\gamma(v) \in U \iff v \in U$.

Lemma 3 *The index $[\mathrm{Aut}(F_r) : \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)]$ of the stabilizer subgroup of U goes to infinity as the index $[F_r : U]$ of the loop subgroup goes to infinity.*

Proof For $i, j \in \{1, \dots, r\}$, $i \neq j$ and $s_i > 1$ consider the automorphism $\gamma \in \mathrm{Aut}(F_r)$ defined by

$$\gamma(g_k) := \begin{cases} g_j g_i, & \text{if } k = j \\ g_k, & \text{if } k \neq j. \end{cases}$$

Let $n \in \{1, \dots, s_i - 1\}$. The n-th power of γ is given by

$$\gamma^n(g_k) = \begin{cases} g_j g_i^n, & \text{if } k = j \\ g_k, & \text{if } k \neq j. \end{cases}$$

The permutation $\pi(g_i)$ is a cycle of length s_i, hence $\pi(g_i)^n$ is also a cycle of length s_i and one can be obtained from the other by renaming the entries of the cycle. Written formally, there exists a permutation $\sigma \in \mathrm{Sym}(N)$ with $\pi(s_i)^n = \sigma\pi(s_i)\sigma^{-1}$ and $\sigma(U) = U$ [Bos06, Abschnitt 5.3, Aufgabe 6]. Note that σ and $\pi(s_j)$ permute.

We have $g_j^{s_j} \in U$. To check whether $\gamma^n(g_j^{s_j})$ is in U, we calculate

$$\begin{aligned}\pi(\gamma^n(g_j^{s_j})) &= \pi(g_j g_i^n)^{s_j} = (\pi(g_j)\pi(g_i)^n)^{s_j} = (\pi(g_j)\sigma\pi(s_i)\sigma^{-1})^{s_j} \\ &= (\sigma\pi(g_j)\pi(s_i)\sigma^{-1})^{s_j} = \sigma(\pi(g_j g_i))^{s_j}\sigma^{-1}\end{aligned}$$

thus

$$\begin{aligned}\gamma(g_j^{s_j}) \in U &\iff \pi(\gamma(g_j^{s_j}))(U) = U \iff \sigma(\pi(g_j g_i))^{s_j}\sigma^{-1}(U) = U \\ &\iff \pi((g_j g_i)^{s_j})(U) = U \iff (g_j g_i)^{s_j} \in U\end{aligned}$$

By consulting the coset graph of U, we know $(g_j g_i)^{s_j - 1} g_j \in U$ and $g_i \notin U$, as $s_i > 1$, so $\gamma^n(g_j^{s_j}) \notin U$ and $\gamma^n \notin \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$.

Since $\gamma^n \notin \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$ for $1 \leq n < s_i$, the index of the stabilizer subgroup must be larger than $s_i - 1$. Hence,

$$[\mathrm{Aut}(F_r) : \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)] \geq \max_{i \in \{1,\dots,r\}} s_i \to \infty$$

as $[F_r : U] \to \infty$. ∎

The following fact about permutations will be very useful for our later calculations:

Lemma 4 *Let $\sigma, \omega \in S_n$ be permutations of the form*

$$\sigma = (1, 2, \dots, m) \quad \text{and} \quad \omega = (1, m+1, \dots, n)$$

with $1 < m < n$. Then all even permutations are in the commutator subgroup of the group generated by σ and ω, i.e. they can be written as a product of commutations of σ and ω:

$$A_n \leq [\langle \sigma, \omega \rangle, \langle \sigma, \omega \rangle]$$

where A_n is the alternating group of degree n and $[G, G]$ denotes the commutator subgroup of G.

Proof The alternating group A_n, $n \geq 3$, is generated by all three-cycles in S_n [Bos06, section 5.3, Satz 3]. We first generate all three-cycles which do not fix the 1. These are cycles of the form $(1, i, j)$. There are four cases:

- $1 < j \leq m$ and $m < i \leq n$. In this case, we take the inverse and enter the next case.
- $1 < i \leq m$ and $m < j \leq n$. We write the cycle using commutators of ω and σ:

$$(1, i, j) = \sigma^{i-1} \circ \omega^{j-m} \circ \sigma^{-(i-1)} \circ \omega^{-(j-m)}$$

- $1 < i \leq m$ and $1 < j \leq m$. We can reduce this case to the previous by writing:

$$(1, i, j) = (1, j, m+1)^{-1} \circ (1, i, m+1)$$

- $m < i \leq n$ and $m < j \leq n$. This case works analogously to the previous case:

$$(1,i,j) = (1,2,j) \circ (1,2,i)^{-1}$$

Observe that $\omega \circ \sigma = (1,2,\ldots,n)$. If we now have a general three-cycle (k,i,j), it is conjugate to a three-cycle that moves the 1:

$$(k,i,j) = (\omega \circ \sigma)^{k-1} \circ (1, i-(k-1), j-(k-1)) \circ (\omega \circ \sigma)^{-(k-1)}$$

So we can write any even permutation in S_n as a product of ω's and σ's, such that for either of the two generators, the sum of its occurrences, counting negative powers negatively, is zero. ■

3.2.2 The linear group and the principal congruence subgroup

The elementary matrix $X_{ij} \in \mathrm{GL}_r(\mathbb{Z})$, $i \neq j$, is the matrix with ones on the diagonal, one additional one in the i-th row and j-th column and zeroes everywhere else. The elementary matrices generate $\mathrm{SL}_r(\mathbb{Z})$ [Sur03, Theorem 4-3.2]. Another generating set is

$$\{X_{ik}, k \neq i\} \cup \{X_{ji}, j \neq i\}$$

for a fixed i. This follows from the relations $[X_{ji}, X_{ik}] = X_{jk}$ for $j \neq k$ [Sur03, Theorem 4-3.2].

The matrix $T_i := \mathrm{Diag}(1,\ldots,1,-1,1,\ldots,1)$ is defined as the identity matrix with the exception of one -1 on the diagonal in the i-th row. The index of $\mathrm{SL}_r(\mathbb{Z})$ in $\mathrm{GL}_r(\mathbb{Z})$ is 2, thus together with T_1 either of the generating sets above generate the whole group $\mathrm{GL}_r(\mathbb{Z})$.

A similar result for the principal congruence subgroup requires some calculations and therefore, it is formulated as a lemma:

Lemma 5 *The principal congruence subgroup of level 2 in* $\mathrm{GL}_r(\mathbb{Z})$

$$\Gamma_2 = \{A \in \mathrm{GL}_r(\mathbb{Z}) \mid A \equiv \mathrm{Id} \pmod 2\},$$

is generated by the set

$$\{X_{ij}^2, i \neq j\} \cup \{T_i, i = 1,\ldots,r\}$$

of squares of elementary matrices and diagonal matrices which are the identity matrix with the exception of one -1 on the diagonal.

PROOF To show this, we transform a matrix $M = (M_{ij}) \in \Gamma_2$ to the identity matrix, by multiplying it with the given generators. The idea of the proof is to apply the idea behind the Euclidean algorithm. In this proof, M_{ij} will always refer to the current state of the matrix, including any transformation applied so far.

Multiplying the matrix X_{ij}^2 from the left has the effect of adding the j-th row twice on the i-th row. Multiplying it from the right has the effect of adding the i-th column twice on the j-th column. Multiplication from the left with the matrix T_i inverts the signs of the entries in the i-th row.

Starting with the first column, we repeat the following step until the column is that of the identity matrix:

Let i be a row with a smallest non-zero absolute value in the first column, and let j be another row with a larger absolute value in the first column. Add the i-th row on the j-th row $2k$ times, by multiplication of X_{ji}^{2k}, with a $k \in \mathbb{Z} \setminus \{0\}$ such that $|M_{j1} + 2kM_{i1}|$ becomes as small as possible.

Note that if all entries in the first column but one are zero, this one value must be ± 1, as it divides the determinant of the matrix. If there are more than one non-zero entries, they can not be all of equal absolute value, as M_{11} is odd and the other entries are even. Hence, we will always find such rows i and j.

Also, each step reduces the sum of the absolute values in the column $\sum_{i=1}^{r} |M_{i1}|$, because $|M_{j1} + 2kM_{i1}| < |M_{j1}|$. As $\sum_{i=1}^{r} |M_{i1}|$ is always positive, this algorithm terminates when $\sum_{i=1}^{r} |M_{i1}| = 1$, which means, due to the invariant parity of the entries, that the first column is $(1, 0, \dots, 0)^\top$ or $(-1, 0, \dots, 0)^\top$.

If the upper right entry of the matrix is now -1, we multiply the matrix by T_1. M_{11} is now 1. As all entries M_{1i}, $i > 1$ in the first row of the matrix, besides the first one, are even, we can now easily cancel them by multiplication from the right with

$$(X_{12}^2)^{-\frac{M_{12}}{2}} \cdots (X_{1r}^2)^{-\frac{M_{1r}}{2}}.$$

to achieve that the first row and column are those of the identity matrix. Repeating this procedure for the other columns as well, we obtain the identity matrix as a product of the matrix M and the given generators. ∎

Corollary 1 *The set*

$$\{X_{ik}, k \neq i\} \cup \{X_{ji}^2, j \neq i\} \cup \{T_j, j = 1, \dots, r\}$$

for some fixed $i \in \{1, \dots, r\}$ generates a superset of Γ_2.

PROOF This can be shown using the following calculation, which uses the commutator relations given in [Sur03], Theorem 4-3.2:

$$\begin{aligned}
X_{jk}^2 &= [X_{ji}, X_{ik}] \cdot X_{jk} \\
&= X_{ji}X_{ik}X_{ji}^{-1}X_{ik}^{-1} \cdot X_{jk} \\
&= X_{ji} \cdot X_{jk} \cdot X_{ik}X_{ji}^{-1}X_{ik}^{-1} \\
&= X_{ji} \cdot [X_{ji}, X_{ik}] \cdot X_{ik}X_{ji}^{-1}X_{ik}^{-1} \\
&= X_{ji}X_{ji}X_{ik}X_{ji}^{-1}X_{ik}^{-1}X_{ik}X_{ji}^{-1}X_{ik}^{-1} \\
&= X_{ji}^2X_{ik}X_{ij}^{-2}X_{ik}^{-1} \\
&= [X_{ji}^2, X_{ik}]
\end{aligned}$$

■

3.3 The odd case

Before handling the most general case, we will first look at loop subgroups with only odd loops, followed by those with only even loops. The case of loop subgroups with mixed parity will then be easily accessible.

Theorem 1 *For a loop subgroup $U \leq F_r$, $r \geq 3$, with all loops odd and at most $r-2$ looplets, we have*

$$B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)) = \mathrm{GL}_r(\mathbb{Z}),$$

where $B\colon \mathrm{Aut}(F_r) \to \mathrm{GL}_r(\mathbb{Z})$ is the map defined in the preface.

I extract the part of the proof that will be re-used in the general case in a statement of its own:

Lemma 6 *Let $U \leq F_r$, $r \geq 3$, be an $s_1/\dots/s_r$-loop subgroup, $i,j \in \{1,\dots,r\}$ with $i \neq j$ and s_i odd. If $s_i = 1$ or there exists $k \in \{1,\dots,r\}$ with $s_k > 1$, then*

$$X_{ij} \in B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)).$$

PROOF (OF LEMMA 6) We construct a preimage of the elementary matrix X_{ij} as a map of the form

$$\gamma_{ij}(g_k) = \begin{cases} w \cdot g_i \cdot g_j, & k = j \\ g_k, & k \neq j \end{cases}$$

with a suitable $w \in F_r$. By Remark 6, this is an automorphism that stabilizes U if the generator g_j does not occur in w and $\pi(w \cdot g_i) = \mathrm{Id}$.

To have $B(\gamma_{ij}) = X_{ij}$, the number of occurrences of all generators in w must be zero each, that is $w \in [F_r, F_r]$. If the i-th loop actually is a looplet, we can choose $w = \mathrm{Id}$, as $\pi(g_i) = \mathrm{Id}$.

If $s_i > 1$, this is where our preliminary calculations about permutations kick in. g_k is another generator whose loop is not a looplet, i.e. $s_k > 1$. If we only consider the set of cosets of U that are on the loops i or k

$$N_{ik} := \{U, g_i^1 U, \dots, g_i^{s_i-1}U,\, g_k^1 U, \dots, g_k^{s_k-1}U\}$$

we can interpret the permutations $\pi(g_i)$ and $\pi(g_k)$ as elements of $\mathrm{Sym}(N_{ik})$. We are now in the situation of Lemma 4 with $\sigma = \pi(g_i)$ and $\omega = \pi(g_k)$, hence $A_{N_{ik}} \leq \pi([\langle g_i, g_k\rangle, \langle g_i, g_k\rangle])$. Since $\pi(g_i)$ is a cycle of odd length s_i, it is itself an even permutation, thus $\pi(g_i) \in A_{N_{ik}}$. This proves the existence of a word $w \in [\langle g_i, g_k\rangle, \langle g_i, g_k\rangle]$ with $\pi(w) = \pi(g_i)^{-1}$, that is $\pi(w \cdot g_i) = \mathrm{Id}$. With this word, the automorphism γ_{ij} is indeed in $\mathrm{Stab}_{\mathrm{Aut}(F_z)}(U)$ and a preimage of X_{ij}. ■

Proof (of Theorem 1) In order to show that the image of the stabilizer subgroup of U is the full linear group, we will find preimages of a generating set of $\mathrm{GL}_r(\mathbb{Z})$. The automorphism $\tau_i \in \mathrm{Aut}(F_r)$ that sends $g_i \mapsto g_i^{-1}$ and $g_j \mapsto g_j$ for $j \neq i$ stabilizes any loop subgroup, in particular U. Thus we have τ_1 as a suitable preimage of T_1 under B.

If there are at most $r-3$ looplets in the loop subgroup, we can choose suitable $k \in \{1, \dots, r\}$ with $s_k > 1$ for all $i, j \in \{1, \dots, r\}$, $i \neq j$. Thus we find preimages in $\mathrm{Stab}_{\mathrm{Aut}(F_z)}(U)$ for all elementary matrices, by Lemma 6, and the proof is complete.

If there are $r-2$ looplets, since $r \geq 3$, there is at least one looplet, for example the i-th. Now we can find preimages for the elements of the generating set

$$\{X_{ik}, k \neq i\} \cup \{X_{ji}, j \neq i\}$$

of $\mathrm{SL}_n(\mathbb{Z})$, concluding the proof: For the first set of matrices, Lemma 6 can be applied since $s_i = 1$. For the second set either $s_j = 1$ or there is a suitable k, as there are two proper loops and the i-th is none of them, hence Lemma 6 can be applied. ■

Example 7 For the 3/3/1 loop subgroup seen in Example 6 on page 40, the following automorphisms are constructed:

$$\begin{aligned}
\gamma_{31}(x,y,z) &= (z \cdot x, y, z)\\
\gamma_{32}(x,y,z) &= (x, z \cdot y, z)\\
\gamma_{13}(x,y,z) &= (x, y, yxy^{-1}xyx^{-2}y^{-1} \cdot x \cdot z)\\
\gamma_{23}(x,y,z) &= (x, y, \underbrace{xyx^{-1}yxy^{-2}x^{-1}}_{\in[\langle x,y\rangle,\langle x,y\rangle]} \cdot y \cdot z).
\end{aligned}$$

3.4 The level

To get closer to the general case, we will now find a useful lower bound to the subgroup $B(\mathrm{Stab}_{\mathrm{Aut}(F_z)}(U))$. Once we know that the principal congruence subgroup of level 2

$$\Gamma_2 = \{M \in \mathrm{GL}_r(\mathbb{Z}) \mid M \equiv \mathrm{Id} \pmod 2\}$$

is contained in $B(\mathrm{Stab}_{\mathrm{Aut}(F_z)}(U))$, we can consider the projection

$$\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_z)}(U))} \leq \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}),$$

which still contains all information about the group.

The following set generates Γ_2 according to Lemma 5:

$$\{X_{ij}^2,\ i \neq j\} \cup \{T_j, j = 1, \dots, r\}.$$

The set

$$\{X_{ik}, k \neq i\} \cup \{X_{ji}^2, j \neq i\} \cup \{T_j, j = 1, \dots, r\}$$

for some fixed i generates a superset of Γ_2 (Corollary 1)

Lemma 7 *For a loop subgroup $U \leq F_r$, $r \geq 3$, with at most $r-2$ looplets, we have*

$$\Gamma_2 \leq B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)).$$

PROOF As in the proof for the odd case, Theorem 1, we will find preimages in the stabilizer subgroup $\mathrm{Stab}_{\mathrm{Aut}(F_z)}(U)$ to the elements of a generating set of Γ_2. As preimages for the matrices $\{T_j, j = 1, \dots, r\}$ we choose the same automorphisms τ_j, which invert the j-th generator g_j and do not modify the other generators.

We will find preimages of squares of elementary matrices X_{ij}^2 as maps $\gamma_{ij}^{(2)}$ of the form

$$\gamma_{ij}^{(2)}(g_k) = \begin{cases} w \cdot g_i^2 \cdot g_j, & k = j \\ g_k & k \neq j. \end{cases}$$

The important fact to consider here is that while $\pi(g_i)$ may not be an even permutation, $\pi(g_i^2) = \pi(g_i)^2 \in A_N$ certainly is. Therefore Lemma 4 provides us with a suitable $w \in F_r$ so that $\gamma_{ij}^{(2)} \in \mathrm{Stab}_{\mathrm{Aut}(F_r)}$ by Remark 6.

If we have at most $r-3$ looplets, we find preimages to all squares of elementary matrices this way. In the case of $r-2$ looplets, we find, as above, preimages for each of

$$\{X_{ik}, k \neq i\} \cup \{X_{ji}^2, j \neq i\}$$

for a fixed i with $s_i = 1$. ■

3.5 Stabilizer subgroups in $\mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$

When we inspect the loop subgroup with even loops, we will come across the group of integral matrices with odd column sums. It contains the principal congruence subgroup of level 2, hence it is completely determined by its image in $\mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$:

$$S(\mathbb{1}) := \{M \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid \mathbb{1} \cdot M = \mathbb{1}\}$$

where $\mathbb{1} = (1, \ldots, 1) \in (\mathbb{Z}/2\mathbb{Z})^r$ is the row vector, all of whose entries are one. More general, we will be interested in the stabilizer subgroup of a row vector $v = (v_1, \ldots, v_r) \in (\mathbb{Z}/2\mathbb{Z})^r$ under the action of right multiplication:

$$S(v) := \{M \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid v \cdot M = v\}$$

These are indeed groups, as $\mathrm{Id} \in S(v)$ and for $M_1, M_2 \in S(v)$ we have $v \cdot (M_1 M_2) = (v \cdot M_1) \cdot M_2 = v \cdot M_2 = v$ and $v = v \cdot M_1 M_1^{-1} = v \cdot M_1^{-1}$. Of course we have $S(0) = \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$.

3.5.1 Generators

Lemma 8 *The group $S(\mathbb{1})$ is generated by "double elementary matrices" of the form $X_{ik}X_{jk}$, $i, j, k \in \{1, \ldots, r\}$ pairwise distinct, having ones on the diagonal and in two additional positions in the same column, and zeros everywhere else.*

More general, $S(v)$ is generated by the elementary matrices X_{ij} for $i \neq j$ and $v_i = 0$ and the double elementary matrices $X_{ik}X_{jk}$ for $i, j, k \in \{1, \ldots, r\}$ pairwise distinct and $v_i = v_j = 1$

Note that the case $v = 0$ is covered by the lemma, as $S(0) = \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$ is generated by all the elementary matrices X_{ij}, $i \neq j$. If the vector v is a unit vector, i.e. precisely one entry is 1, the generating set does indeed not contain any double elementary matrices.

PROOF What does the defining equation $v \cdot M = v$ tell us? On the one hand, any row i of the matrix where the corresponding entry v_i in the row vector is zero does not affect the equation at all and its entries are irrelevant. On the other hand, the sum of all relevant entries in a column j is odd if and only if $v_j = 1$. Therefore, the alleged generators are indeed contained in $S(v)$.

We first summarize the effect that multiplying one of these generators to an matrix in $S(v)$ has: The multiplication from the left of an elementary matrix X_{ij} has the effect of adding the j-th row on the i-th row. The multiplication from the left of a double elementary matrix $X_{ik}X_{jk}$ has the effect of adding the k-th row simultaneously on the i-th and the j-th row.

We will now transform a matrix in $S(v)$ into the identity matrix by multiplying the alleged generators from the left. To do so, we suppose that the first $i-1$ columns are already that of the elementary matrix and treat each column $i = 1, \dots, r$ consecutively as follows:

1. Ensure that there is a 1 on the diagonal. If there is none, there still must be at least one 1 in the column, otherwise the matrix would not be regular. There even must be a 1 in a row j with $j > i$, as otherwise the current column would be a linear combination of the columns to the left of it, which are, due to our transformation, the unit vectors $e_1, \dots, e_{i-1}$.

   Now add this row j on the i-th row (multiplication with X_{ij}). If $v_i = 1$, then, to be allowed to do that, also add the j-th row on any other row k with $v_k = 1$ (multiplication with $X_{ij}X_{kj}$). This step does not alter the previous columns, as the row j has zeros there.

   If there is no such other row k, that is, if there are exactly two ones in the vector v, namely $v_i = v_j = 1$, and if the current column has only one 1 in the j-th row, some additional shuffling is necessary. Let $k \neq i, j$ be any other row. This implies $v_k = 0$. The matrix

   $$(X_{ik}X_{jk} \cdot X_{ki} \cdot X_{kj})^2$$

   is actually the permutation matrix that swaps the i-th and j-th row, and is here written in terms of the given generators. Multiplying this matrix from the left, we move the 1 to the right spot, while again not altering the previous columns.

2. Eliminate all ones that are not on the diagonal. Ones that are in rows j with $v_j = 0$ can be eliminated directly with X_{ji}. Ones in rows j and k with $v_j = v_k = 1$ can only be eliminated pairwise with $X_{ji}X_{ki}$. But in any case there is an even number of them: Either $v_i = 0$. Then we know that there is an even number of ones on relevant rows to be eliminated. Or $v_i = 1$, then there is an odd number of relevant ones in this column. But the i-th row itself is relevant, and we do not want to eliminate the one on the diagonal. This leaves an even number of relevant ones to be eliminated.

So any matrix in $S(v)$ can be transformed into the identity matrix using the given generators, thus they indeed generate all of $S(v)$. ∎

3.5.2 Maximality

Although not strictly required for what follows, for a better understanding of the subgroups $S(v)$ we will now see that they are maximal if $v \neq 0$.

Lemma 9 *Any subgroup* $S(v) \leq \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$ *with* $v \in (\mathbb{Z}/2\mathbb{Z})^r \setminus \{0\}$ *is conjugated to* $S(\mathbb{1})$.

Proof Let $N \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$ be a matrix with $\mathbb{1} \cdot N = v$. Then

$$\begin{aligned} S(v) &= \{M \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid v \cdot M = v\} \\ &= \{M \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid \mathbb{1} \cdot N \cdot M = \mathbb{1} \cdot N\} \\ &= \{M \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid \mathbb{1} \cdot N \cdot M \cdot N^{-1} = \mathbb{1}\} \\ &= \{N^{-1} \cdot M' \cdot N \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid \mathbb{1} \cdot M' \cdot = \mathbb{1}\} \\ &= N^{-1} S(\mathbb{1}) N. \end{aligned}$$ ∎

Lemma 10 *Any subgroup* $S(v) \leq \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$, $r \geq 3$, *with* $v \in (\mathbb{Z}/2\mathbb{Z})^r \setminus \{0\}$ *is maximal, that is, there is no proper subgroup of* $\mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$ *that is a proper supergroup of* $S(v)$.

Proof Due to Lemma 9 it is sufficient to show that $S(\mathbb{1})$ is maximal. To show this we take a matrix $M \notin S(\mathbb{1})$ and transform it into an elementary matrix by multiplying it with matrices in $S(\mathbb{1})$. Since $S(\mathbb{1})$ also contains the permutation matrices, we then get all elementary matrices and thus the full group $\mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$.

As established in the previous section, multiplication from the left with a double elementary matrix $X_{ik}X_{jk}$ has the effect of adding the k-th row on the i-th and on the j-th row. Multiplication from the left with a permutation matrix swaps rows. Multiplication from the right with $X_{ik}X_{jk}$ has the effect of replacing the k-th column with the sum of the k-th, i-th and j-th columns. The permutation matrices swap the columns when multiplied from the right.

The matrix M has at least one column with an even column sum, otherwise we had $M \in S(\mathbb{1})$. It also has at least one column with an odd column sum. Otherwise M would map the standard basis vectors to vectors with even sums. But these would not span all of $(\mathbb{Z}/2\mathbb{Z})^r$, as the sum of two such vectors has again even sum. By swapping the columns we can ensure that the first column has an odd column sum and the last column has an even column sum. Then we make all column sums in between odd: If column j, $1 < j < r$ has an even column sum we multiply from the right with $X_{1j}X_{rj}$. This replaces this column by the sum of the first, j-th and last column, which then has, as required, odd column sum.

Now we proceed by running the Gaussian Elimination Algorithm for the first $r-1$ columns, as done in the previous section. Since all these columns have an odd column sum, this is possible using transformations which are represented by matrices in $S(\mathbb{1})$. Hence, we obtain a matrix of the form

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & * \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ \vdots & & \ddots & 1 & * \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}.$$

■

The transformations did not alter the parity of the column sums, thus the last column still has an even sum of $2k+2$ for some $k \in \mathbb{N}$. By adding the last row to $2k$ of the other rows having a one in the last column, which is possible with matrices from $S(\mathbb{1})$, we eliminate all but two ones in the last column. This is then an elementary matrix, finishing this proof.

3.6 The even case

With the knowledge about $S(\mathbb{1})$ from the previous section we can find out more about the image of stabilizer subgroups of loop subgroups with all loops even.

Theorem 2 *For a loop subgroup $U \leq F_r$, $r \geq 3$, with all loops even, we have*

$$S(\mathbb{1}) \leq \overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$$

PROOF The set of double elementary matrices $X_{ik}X_{jk}$ generate $S(\mathbb{1})$. By the following lemma, these are in $\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$. ■

Lemma 11 *Let $U \leq F_r$, $r \geq 3$ be an $s_1/\ldots/s_r$-loop subgroup and $i,j,k \in \{1,\ldots,r\}$ distinct with s_i and s_j even. Then*

$$X_{ik}X_{jk} \in B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)).$$

PROOF We again vary the construction that led us to preimages of generators of $\mathrm{GL}_r(\mathbb{Z})$ in the odd case (Lemma 6) and of Γ_2 when calculating the level (Lemma 7). This means that we find automorphisms γ_{ijk} of the form

$$\gamma_{ijk}(g_l) = \begin{cases} w \cdot g_i g_j \cdot g_k, & l = k \\ g_l, & l \neq k. \end{cases}$$

such that γ_{ijk} is a preimage of $X_{ik}X_{jk}$. The conditions under which these are actually automorphisms, and under which they stabilize U are given by Remark 6. We know that the cycles $\pi(g_i)$ and $\pi(g_j)$ are both of even length, thus odd permutations. Therefore, their product $\pi(g_i g_j) \in A_N$ and with Lemma 4 we can find a suitable $w \in [\langle g_i, g_j \rangle, \langle g_i, g_j \rangle]$. ■

3.7 The mixed case

If we combine the proofs for the odd and the even case, we can find an analogous result for general loop subgroups.

Theorem 3 *For a loop subgroup $U \leq F_r$, $r \geq 3$, with at most $r-2$ looplets, we have*

$$S(v) \leq \overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$$

where

$$v_i = \begin{cases} 0, & \text{if } s_i \text{ odd} \\ 1, & \text{if } s_i \text{ even.} \end{cases}$$

PROOF By Lemma 8, $S(v)$ is generated by the elementary matrices X_{ij} for $i \neq j$ and $v_i = 0$ and the double elementary matrices $X_{ik}X_{jk}$ for $v_i = v_j = 1$.

The double elementary matrices $X_{ik}X_{jk}$ are in $\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$ for $v_i = v_j = 1$, as shown in Lemma 11.

Let $i, j \in \{1, \ldots, r\}$, $i \neq j$ and $v_i = 0$. We need to show $X_{ij} \in \overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$. If $s_i = 1$ or there is $k \in \{1, \ldots, r\}$ with $k \neq i, j$ and $s_k > 1$, this follows from Lemma 6. If that is not the case, we are in the situation of $r-2$ looplets with $s_i > 1$, $s_j > 1$ and $s_k = 1$ for $k \neq i, j$. Invoking Lemma 6 for X_{ik} and for X_{kj}, and using $[X_{ik}, X_{kj}] = X_{ij}$, we show $X_{ij} \in \overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$. ■

3.8 Sharper bounds

To fully understand the situation, we yet have to find out whether the lower bound $S(v)$ from Theorem 3 is already the full group $\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$. It turns out that this is indeed the case.

Information on the upper bound of the image of the stabilizer subgroup can be obtained in the more general setting of congruence subgroups of level 2:

Theorem 4 *Let $U \leq F_r$, $r \geq 3$, be a subgroup with $\Gamma_2 \leq B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$, and let*

$$v_i = \begin{cases} 0, & \text{if } \pi(g_i) \text{ is an even permutation} \\ 1, & \text{if } \pi(g_i) \text{ is an odd permutation.} \end{cases}$$

Then,

$$\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))} \leq S(v).$$

PROOF Let $\gamma \in \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$. For a generator g_i, $i \in \{1, \dots, r\}$, the permutation $\pi(\gamma(g_i))$ has the same parity as $\pi(g_i)$, by Lemma 2. Therefore, the number of odd permutations in the word $\gamma(g_i)$ equals v_i modulo 2:

$$\sum_{j=1}^{r} \#_{g_j} \gamma(g_i) \cdot v_j \equiv v_i \pmod 2$$

This condition can be written as $v \cdot \overline{B(\gamma)} = v$, hence $\overline{B(\gamma)} \in S(v)$. ■

This is an interesting result, as the subgroups $S(v)$ are maximal, but far from the only maximal groups.

Corollary 2 *For a loop subgroup $U \leq F_r$, $r \geq 3$, with at most $r-2$ looplets, we have*

$$S(v) = \overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$$

where

$$v_i = \begin{cases} 0, & \text{if } s_i \text{ odd} \\ 1, & \text{if } s_i \text{ even.} \end{cases}$$

PROOF Theorem 4 applies because $\Gamma_2 \leq \overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))}$ by Lemma 7 and the permutation $\pi(g_i)$ is odd if and only if s_i is even. Theorem 3 provides the inclusion in the other direction. ■

3.9 The excluded case

In the previous sections, we have always excluded loop subgroups with exactly $r-1$ looplets. These subgroups have some special properties that make them break rank and therefore, they are handled separately here.

Let U be a loop subgroup with exactly $r-1$ looplets and assume without loss of generality that $s_1 \neq 1$. The subgroup can now be written as

$$U = \{w \in F_r \mid \#_x w \equiv 0 \pmod{s_1}\}.$$

This fact is easily seen by tracing a word in the coset graph of U: Only occurrences of x advance on the graph, and x must occur a multiple of s_i times to end up at the node U.

Let $A\colon F_r \to \mathbb{Z}^r$ be the surjective abelianization map defined by $w \mapsto (\#_{g_i} w)_{i=1,\dots,r}$. Then $U = A^{-1}(U')$ with $U' := \{v \in \mathbb{Z}^r \mid v_1 \equiv 0 \mod s_1\} \leq \mathbb{Z}^r$. This allows us to calculate $B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$ using the following, more general observation:

Lemma 12 *Let $\varphi\colon G \to G'$ be a surjective group homomorphism with its kernel characteristic in G. Let $H \leq G$ and $H' \leq G'$ be subgroups such that $H = \varphi^{-1}(H')$. Let $\psi\colon \mathrm{Aut}(G) \to \mathrm{Aut}(G')$ be the map induced by φ. Then*

$$\psi(\mathrm{Stab}_{\mathrm{Aut}(G)}(H)) = \mathrm{Stab}_{\mathrm{Aut}(G')}(H') \cap \psi(\mathrm{Aut}(G)).$$

PROOF For $\gamma \in \mathrm{Aut}(G)$, $y \in G'$ the map $\psi\colon \mathrm{Aut}(G) \to \mathrm{Aut}(G')$ is defined by $\psi(\gamma)(y) := \varphi(\gamma(x))$ for an $x \in \varphi^{-1}(y)$. This is well-defined because the kernel of φ is characteristic. Thus

$$\begin{aligned}
\psi(\mathrm{Stab}_{\mathrm{Aut}(G)}(H)) &= \psi(\{\gamma \in \mathrm{Aut}(G) \mid \gamma(H) = H\}) \\
&= \{\psi(\gamma) \mid \gamma \in \mathrm{Aut}(G),\ \forall x \in H : \gamma(x) \in H\} \\
&= \{\psi(\gamma) \mid \gamma \in \mathrm{Aut}(G),\ \forall x \in H : \varphi(\gamma(x)) \in \varphi(H)\} \\
&= \{\psi(\gamma) \mid \gamma \in \mathrm{Aut}(G),\ \forall y \in H' : \varphi(\gamma(\varphi^{-1}(y))) \in \varphi(H)\} \\
&= \{\psi(\gamma) \mid \gamma \in \mathrm{Aut}(G),\ \forall y \in H' : \psi(\gamma)(y) \in H'\} \\
&= \{\gamma' \in \mathrm{Aut}(G') \mid \forall y \in H' : \gamma'(y) \in H'\} \cap \psi(\mathrm{Aut}(G)) \\
&= \mathrm{Stab}_{\mathrm{Aut}(G')}(H') \cap \psi(\mathrm{Aut}(G))
\end{aligned}$$

■

Proposition 1 *For the $s_1/1/\dots/1$ loop subgroup U, $s_i \in \mathbb{N}$, of F_r we have*

$$\Gamma_{s_1} \leq B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)) = \mathrm{Stab}_{\mathrm{GL}_r(\mathbb{Z})}(U') \leq \mathrm{GL}_r(\mathbb{Z})$$

where

$$U' := \{v \in \mathbb{Z}^r \mid v_1 \equiv 0 \mod s_1\}.$$

PROOF As noted before, $U = A^{-1}(U')$. B is surjective, because preimages of a generating set of $\mathrm{GL}_r(\mathbb{Z})$ are given by τ_1 and the automorphisms in the proof of Lemma 6. Therefore, Lemma 12 gives us

$$B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)) = \mathrm{Stab}_{\mathrm{GL}_r(\mathbb{Z})}(U').$$

Since every matrix in Γ_{s_1} stabilizes U', $B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$ contains Γ_{s_1}. ■

As a matter of fact, s_1 is the level of the congruence subgroup $B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$. The difference to the other loop subgroups is evident: While here we easily reach any congruence subgroup level, in the other cases we do not exceed level 2.

3.10 Index two subgroups of F_r

The property of subgroups of F_r that the image of their stabilizer subgroups under $\bar{B}$ is one of the groups

$$S(v) = \{M \in \mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) \mid v \cdot M = v\}$$

for $v \in (\mathbb{Z}/2\mathbb{Z})^r$ is not limited to loop subgroups. In this section, we will see that it holds for subgroups of F_r of index 2 as well. These are in general not loop subgroups.

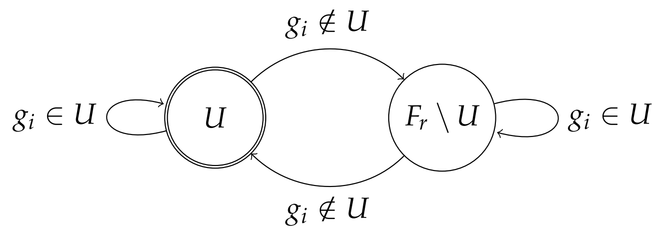
Theorem 5 *Let $U < F_r$ be a subgroup of index two. We have*

$$\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))} = S(v)$$

where

$$v_i = \begin{cases} 1, & \text{if } g_i \notin U \\ 0, & \text{if } g_i \in U. \end{cases}$$

PROOF As shown e.g. in [Fre08], Lemma 3.7, a subgroup $U < F_r$ of index 2 is fully determined by the generators of the free group that are contained in U, since $g_i^2 \in U$ for all generators g_i. The coset graph of U is:



As can be seen from the graph, we actually have $g_i^2 \in \mathrm{NT}(U)$. By the argument of Section 3.2.1, preimages of the generators X_{ij}^2, $i \neq j$, and T_i of Γ_2 in the stabilizer subgroup of U can be given explicitly by the automorphisms $\gamma_{ij} : g_j \mapsto g_i^2 g_j$, $g_k \mapsto g_k$, $k \neq j$, and τ_i. This proves $\Gamma_2 \leq B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))$ and we can actually project this group into $\mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z})$ without losing information.

For a word $w \in F_r$, only the number of generators with $g_i \notin U$ in the word decide whether it is in U, as only those have an effect when tracing the word in the coset graph. Due to our definition of v, this leads to the equation

$$w \in U \iff \sum_{j=1}^{r} \#_{g_j} w \cdot v_j \equiv 0 \pmod 2$$

and thus for all $i \in \{1, \dots, r\}$

$$\sum_{j=1}^{r} \#_{g_j} \gamma(g_i) \cdot v_j \equiv v_i \pmod 2.$$

We first prove $\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))} \geq S(v)$. For a matrix $M \in S(v)$, take any preimage $\gamma \in \bar{B}^{-1}(M)$. For a word $w \in U$ we have, using the two previous equations,

$$\begin{aligned}
\sum_{j=1}^{r} \#_{g_j} \gamma(w) \cdot v_j &= \sum_{j=1}^{r} \left(\sum_{i=1}^{r} \#_{g_j} \gamma(g_i) \cdot \#_{g_i} w \right) \cdot v_j \\
&= \sum_{i=1}^{r} \#_{g_i} w \cdot \left(\sum_{j=1}^{r} \#_{g_j} \gamma(g_i) \cdot v_j \right) \\
&\equiv \sum_{i=1}^{r} \#_{g_i} w \cdot v_i \pmod 2 \\
&\equiv 0 \pmod 2,
\end{aligned}$$

showing $\gamma(w) \in U$ and thus $\gamma \in \mathrm{Stab}_{\mathrm{Aut}(F_r)}(U)$.

The other inclusion, $\overline{B(\mathrm{Stab}_{\mathrm{Aut}(F_r)}(U))} \leq S(v)$, follows from Theorem 4 and the fact that $\pi(g_i)$ is even if and only if $g_i \in U$. ■

3.11 Discussion and further directions

The study of images of stabilizer subgroups under the map B was motivated by the analogy to origamis, which are restricted to the rank $r = 2$. Comparing these settings, we see that higher ranks cause the objects to become simpler. In [HL05], Theorem B and Conjecture 1, the size of the orbit of an $L(a,b)$-origami with n squares under the action of $\mathrm{Out}^+(F_2) \cong \mathrm{SL}_r(\mathbb{Z})$ is calculated for even n and conjectured for odd n. This number, which is also the index of the Veech group of the origami, grows without bound as n increases. The F_2 subgroups induced by L-origamis are loop subgroups in our sense, but here we find that, no matter what the index of the loop subgroup (with not $r-2$ looplets) is, the index of the image of its stabilizer subgroup in $\mathrm{GL}_r(\mathbb{Z})$ with $r > 2$, is bounded by $[\mathrm{GL}_r(\mathbb{Z}/2\mathbb{Z}) : S(v)] = 2^r - 1$.

This bound is caused by applying the map B, as the index of the stabilizer subgroups in $\mathrm{Aut}(F_r)$ is still unbounded, as shown in Lemma 3. Thus a lot of information about these subgroups is lost in the process, making this approach less useful if one is primarily interested in the situation in $\mathrm{Aut}(F_r)$.

While our main result has been reached now, we want to give some thoughts about possible further directions.

So far, the loops of the loop subgroups were pairwise disjoint, with exception of the node U itself. This was a crucial part of the argument, as it allowed us, for a pair of generators g_i and g_j, to find another generator g_k, $k \neq i,j$, such that the permutations $\pi(g_j)$ and $\pi(g_k)$ were in the shape required by Lemma 4. But to find such a generator g_k, it is not really necessary that *all* other loops are disjoint, it is sufficient to have *one* such loop. Therefore, one can expect a possible generalization of Corollary 2 to require just this. Unfortunately, for non-loop-subgroups, it is not easy to see that we have pre-images of a matrix T_i or any other matrix with determinant -1.

Example 8 One such group can be obtained by taking the 3/3/1/3 subgroup, and merging the nodes g_4^1U and g_4^2U with x^1U resp. x^2U. The resulting coset graph is shown in Figure 4. The similarity to Example 6 is obvious. Preimages to the elementary matrices are found the same way. We also see that T_2 has the preimage τ_2, thus the image of the stabilizer subgroup is again the whole linear group.

Example 9 But also the more convoluted case of the subgroup with the coset graph from Figure 5, which is obtained by merging the cosets of the 3/3/3/3/3 loop subgroup of F_5, has the property that for generators i, j, there is a third generator k that the permutations $\pi(g_j)$ and $\pi(g_k)$ are in the shape required for Lemma 4. Therefore, preimages to the elementary matrices exist. The automorphism τ_1 is not a stabilizer, so a preimage to T_1 is not easily given. Our results so far tell us that the image of the stabilizer subgroup contains the full special linear group, but to decide whether it is the full group, additional arguments are required. Unfortunately, the

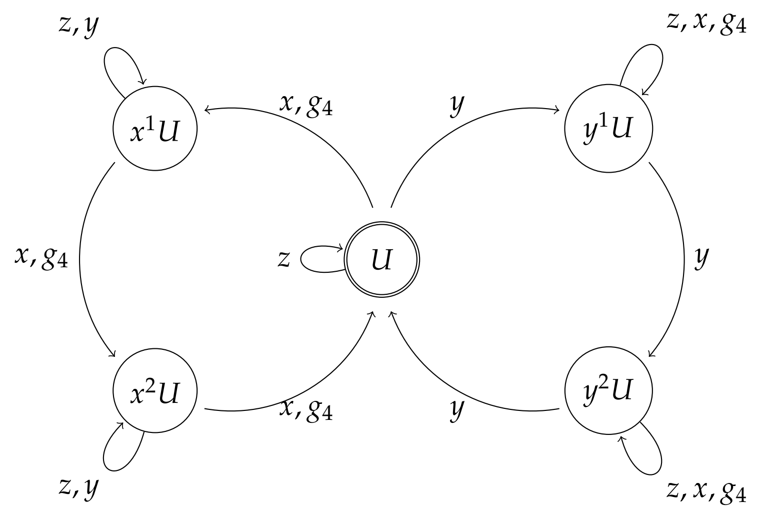


Figure 4: A subgroup obtained from a 3/3/1/3 loop subgroup by merging nodes.

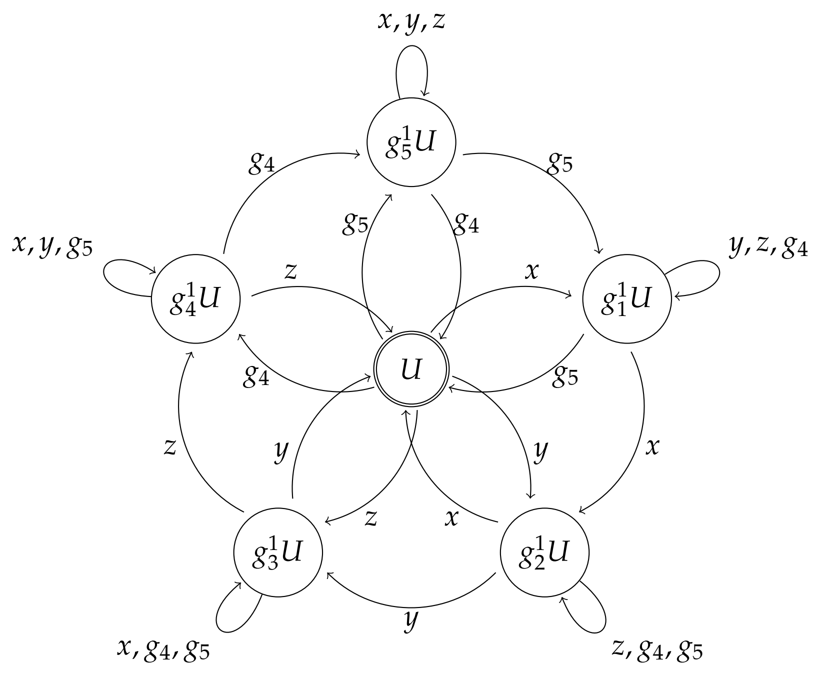


Figure 5: A supergroup of the 3/3/3/3/3 loop subgroup.

code described in Chapter 2 was not able to calculate the stabilizer subgroup in reasonable time.

Table 9 gives, for a given rank of the the free group and a given maximum index the number of conjugacy classes of subgroups, the number of loop subgroups where Corollary 2 is applicable and the number of subgroups where for each pair of generators there is a third generator with permutations as required for Lemma 4. It is calculated using [GAP08] using the script printed on page 63.

Our proofs are based on the fact that we have the whole alternating group A_N as commutators of $\langle \pi(g_j), \pi(g_k) \rangle$, and that the permutations $\pi(g_j)$ in the odd case, $\pi(g_j^2)$ in the level calculation resp. $\pi(g_j g_l)$ in the even case are all even permutations. But the essence of the argument is that these permutations lie in $[\langle \pi(g_j), \pi(g_k) \rangle, \langle \pi(g_j), \pi(g_k) \rangle]$, even if that is might not be the whole group A_N. Using this argument might lead to an even larger class of subgroups where the argument above runs through. Furthermore, we are not limited to two generators to build the commutators from, we just must not use g_i in it, possibly allowing for further generalization.

If a subgroup $U \leq F_r$ is not in a shape accessible by these methods, one can try to obtain a suitable shape by using another basis of F_r. This is equivalent to turning to another subgroup $\gamma(U)$ in the orbit of U under the action of $\mathrm{Aut}(F_r)$. All subgroups in the same orbit have the same stabilizer subgroup (up to conjugation).

It might therefore be fruitful to study to which larger class of subgroups of F_r we can generalize our argument.

rank	max. index	conjugacy classes	conjugacy classes with loop subgroups	conjugacy classes suitable for Lemma 4
3	1	1	1	1
	2	8	1	1
	3	49	4	2
	4	653	11	9
	5	14406	23	33
	6	518659	41	95
4	1	1	1	1
	2	16	1	1
	3	251	7	14
	4	14371	23	197
	5	1727216	54	1866
5	1	1	1	1
	2	32	1	1
	3	1393	11	91
	4	335969	41	3651
6	1	1	1	1
	2	64	1	1
	3	8051	16	481

Table 9: Conjugacy classes containing loop subgroups

Counting loop subgroups

The following GAP script is used to calculate Table 9 on page 61:

```
# Convenience function
everySecond := function (list)
    return list{[2, 4 .. Length(list)]};
end;

# Detecting a permutation with a single cycle
isCycle := function (perm)
    local struct;
    struct := CycleStructurePerm(perm);
    return Length(struct) = 0 or
        (Number(struct) = 1 and struct[Length(struct)] = 1);
end;

# These cases are calculateable in bearable time
cases := [
    [3,1],[3,2],[3,3],[3,4],[3,5],[3,6],
    [4,1],[4,2],[4,3],[4,4],[4,5],
    [5,1],[5,2],[5,3],[5,4],
    [6,1],[6,2],[6,3],
];

for case in cases do
    rank := case[1];
    index := case[2];
    F := FreeGroup(rank);
    countSG := 0;
    countLoopSG := 0;
    countGeneralLoopSG := 0;
    for sg in LowIndexSubgroupsFpGroupIterator(F,index) do
        countSG := countSG + 1;
        # The list of permutations
        perms := List(everySecond(CosetTable(F,sg)), PermList);

        # We want all permutations to be cycles
        if not ForAll(perms, isCycle) then; continue; fi;

        properLoops := Filtered(perms, p −> p <> ());
        if properLoops = [] # The trivial subgroup
        then
            countLoopSG := countLoopSG + 1;
            countGeneralLoopSG := countGeneralLoopSG + 1;
            continue;
        fi;
        # For the loop subgroup, there is one coset moved by all proper loops
        pointsMovedByAll := Intersection(List(properLoops, MovedPoints));
        if Length(pointsMovedByAll) = 1
        then
            # Detect a loop subgroup. Every other point must be moved at most once
            identity := pointsMovedByAll[1];
            if ForAll([1..index], i −>
                i = identity or Number(properLoops, p −> iˆp <> i) <= 1
            ) then
                countLoopSG := countLoopSG + 1;
            fi;
        fi;
        # Detect a general loop subgroup. Every proper loop needs two other proper
        # loops to be almost disjunct from
        if ForAll([1..Length(properLoops)], i −>
            1 < Number( [1..Length(properLoops)], j −>
                i<>j and
                Length(IntersectionSet(
                        MovedPoints(properLoops[i]),
                        MovedPoints(properLoops[j])
                )) = 1
            )
        ) then
            countGeneralLoopSG := countGeneralLoopSG + 1;
        fi;
    od;

    Print("Rank: ", rank, ", Index: ", index,"\n");
    Print("Subgroups: ", countSG, "\n");
    Print("Loop Subgroups: ", countLoopSG, "\n");
    Print("General Loop Subgroups: ", countGeneralLoopSG,"\n");
od;
QUIT;
```

Bibliography

[AFV08] Heather Armstrong, Bradley Forrest, and Karen Vogtmann, *A presentation for* $\mathrm{Aut}(F_n)$, Journal of Group Theory **11** (2008), no. 2, 267–276.

[BBN59] Gilbert Baumslag, W. W. Boone, and B. H. Neumann, *Some unsolvable problems about elements and subgroups of groups.*, Mathematica Scandinavica **7** (1959), 191–201.

[Bos06] Siegfried Bosch, *Algebra*, Springer-Verlag, Berlin, 2006, 6th Edition.

[Cox36] H. S. M. Coxeter, *An Abstract Definition for the Alternating Group in Terms of Two Generators*, J. London Math. Soc. **s1-11** (1936), no. 2, 150–156.

[EM09] Jordan S. Ellenberg and D. B. McReynolds, *Every curve is a Teichmüller curve*, preprint at arXiv:0909.1851 [math.GT], 2009.

[Fre08] Myriam Freidinger, *Stabilisatorgruppen in* $\mathrm{Aut}(F_z)$ *und Veechgruppen von Überlagerungen*, Diplomarbeit, Universität Karlsruhe (TH), 2008.

[GAP08] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.4.12*, 2008.

[HL05] Pascal Hubert and Samuel Lelièvre, *Noncongruence subgroups in* $\mathcal{H}(2)$, International Mathematics Research Notices **2005:1** (2005), 47–64.

[Lee63] John Leech, *Coset enumeration on digital computers*, Mathematical Proceedings of the Cambridge Philosophical Society **59** (1963), no. 02, 257–267.

[Men64] Nathan Saul Mendelsohn, *An algorithmic solution for a word problem in group theory*, Canadian Journal of Mathematics **16** (1964), 509–516.

[Sch05] Gabriela Schmithüsen, *Veech groups of origamis*, Ph.D. thesis, Universität Karlsruhe (TH), 2005.

[Ser97] Ákos Seress, *An introduction to computational group theory*, Notices Amer. Math. Soc **44** (1997), 671–679.

[Sim94] Charles C. Sims, *Computation with finitely presented groups*, Encyclopedia of mathematics and its applications, vol. 48, Cambridge University Press, New York, 1994.

[Sur03] B. Sury, *The congruence subgroup problem*, Hindustan Book Agency, New Delhi, 2003.

[TC36] J. A. Todd and H. S. M. Coxeter, *A practical method for enumerating cosets of a finite abstract group*, Proceedings of the Edinburgh Mathematical Society (Series 2) **5** (1936), no. 01, 26–34.

Erklärung

HIERMIT erkläre ich, Joachim Breitner, dass ich die vorliegende Diplomarbeit selbständig und ausschließlich unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe.

Ort, Datum Unterschrift