CONJUGACY CLASSES IN FINITE PERMUTATION GROUPS
VIA HOMOMORPHIC IMAGES

ALEXANDER HULPKE

ABSTRACT. The lifting of results from factor groups to the full group is a standard technique for solvable groups. This paper shows how to utilize this approach in the case of non-solvable normal subgroups to compute the conjugacy classes of a finite group.

1. Introduction

The determination of all conjugacy classes of a finite permutation group $G$ has already been the subject of several studies [18, 2, 3, 5]. A first approach is to check random group elements for conjugacy until representatives for all conjugacy classes have been found. Despite the simplicity this approach works quite effectively if there are few and large conjugacy classes, for example if the group is almost simple. As soon as there are small conjugacy classes, however, the algorithm will not finish in reasonable time, because representatives from these classes will not be found. Thus one of the main objectives of any better conjugacy class algorithm has to be the creation of a list of group elements of sufficiently small size which contains representatives for all classes.

The inductive approach [3] considers elements as roots of their prime-order powers. Thus every element can be found in the centralizer of a $p$-element. These elements themselves will be found in the Sylow subgroups, which are usually smaller than the group and for which effective methods for the determination of conjugacy classes are known [8].

In some cases, however the sheer size of $p$-element centralizers and the number of rational classes in a Sylow subgroup may render this approach unusable. A typical example is the group

$$
\left( \frac{1}{2} \cdot S_{11}^2 \right)^2 = \langle (1, 2, 3), (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), (1, 2)(12, 13)
(1, 12)(2, 13)(3, 14)(4, 15)(5, 16)(6, 17)(7, 18)(8, 19)(9, 20)(10, 21)(11, 22) \rangle
$$

of degree 22. In this group, for example, the element $(1, 2, 3)$ has a centralizer of size $2^{14}3^75^37^211$ and the 2-Sylow subgroup has 460 classes which fuse to only 70 classes of 2-elements in the full group.
Definition 1. A group $H$ which is the direct product of simple groups of the same type, $H \cong T \times \cdots \times T$ ($T$ simple), is called elementary. If $T$ (and $H$) are abelian, $H$ is called elementary abelian.

If $G$ possesses an elementary abelian normal subgroup $N$, the problem can be reduced substantially: provided the classes of $G/N$ are known, the classes of $G$ can be lifted from these by an affine action on $N$. This leads to an efficient algorithm to determine all conjugacy classes of solvable groups given by PC presentations [15]. As a PC presentation can be computed easily for solvable permutation groups [19], we will only have to consider non-solvable permutation groups further on.

In fact, the affine method from [15] does not require the factor group to be solvable, and so we can in principle even restrict to groups with no solvable normal subgroup. In [5] it is shown that groups of this type have a large normal subgroup that is a direct product. In [5], the authors then suggest parametrizing the classes in this normal subgroup and using a random search outside. Again, however, groups like $\mathbb{Z}_2 \times S_\infty$ tend to possess some very small classes outside the direct product normal subgroup (data to support this claim can be found in Section 11). For groups like these, the random search is likely to take a very long time. This makes it desirable not only to increase the normal subgroup as far as possible but also to develop another way to find the outside classes.

With this aim, we will try to further investigate the structure of groups with a non-abelian normal subgroup to an extent that we can almost parametrize their classes.

Our strategy will be as follows. We will determine a normal series of $G$ with elementary factors: $G = N_0 > N_1 > \cdots > N_r = \{1\}$, $N_{i-1}/N_i \cong T_i$, $T_i$ simple. Section 2 recalls how to do this. As in the case of solvable groups we then proceed down this series, determining the conjugacy classes of $G/N_i$ from those of $G/N_{i-1}$. If $N_{i-1}/N_i$ is abelian, this can be done in the same way as for solvable groups [15]. This paper is therefore concerned only with the case of a non-abelian factor $N_{i-1}/N_i$. To attain our goal, we examine the extension theory of such factors more closely in Section 3 showing that these extensions can be built from subdirect products. Consequentially, Section 4 describes how to obtain the conjugacy classes of a subdirect product from the conjugacy classes of its factors.

Section 5 then shows how to deal with component permutations, and Section 8 shows how to obtain remaining conjugacy classes. Finally, in Section 10 we will briefly discuss implementational issues. A list of the symbols used in the main sections (Sections 3 to 8) is provided at the end.

1.1. Subdirect products. The decomposition of a group as subdirect product of two of its factor groups will be crucial to the procedure. The remainder of this section is therefore devoted to a short recall of its definition:

Suppose the group $G$ has two normal subgroups $N$ and $M$ with $N \cap M = \{1\}$. This gives rise to two projections, $\alpha: G \rightarrow G/N =: A$ and $\beta: G \rightarrow G/M =: B$. It is easily checked that the mapping $G \rightarrow A \times B$, $g \mapsto (g\alpha, g\beta)$ is an injective homomorphism, and we can therefore embed $G$ in the direct product of its factor groups $A$ and $B$. If we embed $G$ this way, $\alpha$ and $\beta$ are the homomorphisms induced by the two component projections of $A \times B$.

Conversely, a subgroup $G$ of $A \times B$ whose images under the component projections $\alpha: A \times B \rightarrow A$ and $\beta: A \times B \rightarrow B$ are $G\alpha = A$ and $G\beta = B$ has two normal
subgroups $\ker \alpha$ and $\ker \beta$ which intersect trivially; the factor groups of those kernels are isomorphic to $A$ and $B$. The factor group $G/(A, B)$ maps to factors of both $A$ and $B$; thus $A$ and $B$ have an isomorphic factor group.

In this situation we say that $G$ is an (inner) subdirect product of its factors $A$ and $B$; we write $G = A 1B$. If $A$ and $B$ are given, a subdirect product is determined up to isomorphism by an isomorphism of factor groups of $A$ and $B$, and any such isomorphism gives rise to a subdirect product:

**Definition 2.** Let $A$ and $B$ be groups, and let $D < A$, $E < B$ be such that there is an isomorphism $\chi: A/D \to B/E$. Denote the natural homomorphisms by $\delta: A \to A/D$ and $\epsilon: B \to B/E$. The (outer) subdirect product of $A$ and $B$ induced by $\chi$ is the subgroup

\[ \{(a, b) \in A \times B \mid a(\delta \chi) = be\} \leq A \times B. \]

(In the language of category theory, a subdirect product is the pull-back in the category of groups.)

A natural generalization is an iterated subdirect product which corresponds to a subgroup of a direct product of more than two groups whose component projections are all surjective. This corresponds to a set of normal subgroups whose intersection is trivial.

## 2. Computing a chief series

Given a permutation group $G$, we first have to obtain a normal series with elementary factors: let $H < G$ be a normal subgroup and suppose that a normal series of $G/H$ is known. (Initially, we know a normal series of $G/G$.) Using the methods for the computation of composition series \[16, 14, 1\], we obtain a subgroup $U < H$ such that $T := H/U$ is simple. Let

\[ L := \bigcap_{g \in G} U^g; \]

then $L < G$ (as intersection of a $G$-orbit) and $L < H$. Additionally $H/L$ is an iterated subdirect product of groups isomorphic $T$; thus $H/L$ is elementary of type $T$. We then proceed with $L$ in place of $H$ until the trivial subgroup is reached.

It takes little further work to obtain a series of normal subgroups that cannot be refined any further. (Such a series is normally called a chief series.) To achieve this some normal factors $H/L$ obtained so far might have to be refined. If $H/L$ is elementary abelian this can be done using standard MeatAxe methods \[17\]. If $H/L$ is not abelian, we claim that $H/L$ is already a chief factor:

**Lemma 3.** Let $T$ be non-abelian simple and $D = T^d$. Then every normal subgroup of $D$ is a direct product of some of the $d$ constituent copies of $T$.

**Proof.** Let $\langle 1 \rangle \neq N < D$ be a nontrivial normal subgroup and $1 \neq n \in N$. Considering $N$ as a direct product of $d$ copies of $T$, we write $n = (t_1, \ldots, t_d)$. Without loss of generality (renumber the components) we can suppose that $t_1 \neq 1$. As $T$ is simple non-abelian, there is an $s \in T$ such that $t_1^s \neq t_1$. Then the quotient (we write $a/b$ as a shorthand for $a \cdot b^{-1}$)

\[ v(s,1,\ldots,1)/n = (t_1^s/t_1, t_2/t_2, \ldots, t_d/t_d) = (t_1^s/t_1, 1, \ldots, 1) \]

has to be contained in $N$. By further conjugation we get elements $(r, 1, \ldots , 1)$ with $r$ running through the $T$-class of $t_1^s/t_1 \neq 1$. The subgroup of $T$ generated by these
elements is a nontrivial normal subgroup of $T$; thus it has to be the whole of $T$. Accordingly, $T_1 \leq N$

Iterated application of this argument shows that any normal subgroup contains every one of the $d$ direct factors isomorphic to $T$ on which it projects non-trivially. Thus it has to be a direct product of those factors.

Lemma 4. The factor $H/L$ is a chief factor.

Proof. Applying Lemma 3 with $D = H/L$, we see that $U/L$ has to be the direct product of all but one of the direct factors. The conjugates of $U$ then are direct products of other $(d - 1)$-combinations of the direct factors. The conjugates form one orbit; thus $G$ has to act transitively on the $d$ factors. Thus any normal subgroup of $G$ that contains one of these direct factors contains all.

An equivalent approach is mentioned in [13], constructing the series upwards instead of downwards. By modifying the composition series algorithm directly, instead of applying it as a subroutine, further improvements are possible [4].

3. Extensions with non-abelian elementary kernel

In the next sections we will describe how to compute the classes of $G/N_i$ based on knowledge of the classes of $G/N_{i-1}$ in the case that $N_{i-1}/N$ is elementary non-abelian. To simplify notation, however, we will (by proceeding to the factor group $G/N_i$) assume that $N_i$ is trivial and that $N = N_{i-1}/N_i$ is an elementary minimal normal subgroup. (The fact that we are computing in a factor group will not pose a problem for the actual calculations; see Remark 5.)

Let $G$ be a finite group with $N \triangleleft G$, $N \cong T^n$ for a non-abelian simple group $T$. Then we can write the elements of $N$ as an $n$-component vector $(t_1, \ldots, t_n)$ with $t_i \in T$. We consider the conjugation action $\varphi$ of $G$ on $N$. Its kernel is the centralizer $C = C_G(N)$. The intersection of $C$ and $N$ consists of those elements of $N$ that contain in every component an element of the centre $Z(T)$. As $T$ is non-abelian simple, $Z(T)$ is trivial; thus $Z(N) = C \cap N = (1)$. Figure 1 illustrates the situation.

Accordingly, $G$ can be considered as a subdirect product of the factor group $G/N$ with the factor $G/C$. This factor is isomorphic to the image $F := G\varphi$ of $G$ under $\varphi$, which is a subgroup of $\text{Aut}(N)$.

As seen in Lemma 3 every normal subgroup of $N$ is the direct product of some of the $n$ copies of $T$. Therefore these copies themselves form a class of normal subgroups of $N$ that is permuted by every automorphism. We denote this permutation

$$G/N \quad G \quad F \quad F\psi$$

$$\varphi$$

$$\langle N, C \rangle$$

$$N \quad C$$

$$\langle 1 \rangle$$

$$\langle 1 \rangle$$

Figure 1. Structure of an extension with non-abelian kernel
action by $\psi$. An element of $\text{Aut}(N)$ that fixes one component acts on this component like an element of $\text{Aut}(T)$. The action of $\text{Aut}(N)$ on $N$ thus is composed from an $\text{Aut}(T)$ action in the components and permutation of the components among each other. Conversely, every such composition forms an automorphism of $N$, so $\text{Aut}(N) \cong \text{Aut}(T) \wr S_n$.

The base group of this wreath product is $\text{Aut}(T)^n$, which in turn contains the elementary characteristic subgroup $E := T^n$. As $N \leq G$, we have in fact $E \leq F = G\varphi \leq \text{Aut}(N)$. If $N$ is a chief factor of $G$, by the same argument as in the proof of Lemma [4] the permutation image $F\psi$ is a transitive subgroup of $S_n$.

**Remark 5.** If $\text{Out}(T) = \langle 1 \rangle$ we actually get $F = T \wr (F\psi)$. Otherwise, still, by the proof of Schreier’s conjecture based on the classification of finite simple groups [10, Theorem 1.46] the index of $G\varphi$ in $\text{Aut}(T) \wr (F\psi)$ will be fairly small, provided that $n$ is not overly large.

If we take a transitive permutation representation of $T$ that extends to $\text{Aut}(T)$, we get a transitive action of $\text{Aut}(T) \wr S_d$ on $\deg(T) \cdot d$ points (the natural imprimitive action). In this action, the image $G\varphi$ is a transitive subgroup of the wreath product, and its base group $M := \ker \psi$ is an iterated subdirect product of a group containing $T$ and contained in $\text{Aut}(T)$. (Groups of this type have been classified in [11].)

Every extension of a factor $F$ with a kernel isomorphic to $T^d$ thus is isomorphic to a subdirect product of $F$ with a transitive imprimitive group of the described type. (One could also use the classification of those imprimitive groups together with a classification of subdirect products to get a classification of all extensions of a given factor group with elementary kernel of type $T$.)

**4. Conjugacy classes of a subdirect product**

The last section’s analysis shows that subdirect products are building blocks of group extensions with non-abelian elementary kernel. Thus a description of their conjugacy classes is essential to describe the classes of extensions. In this section we consider a group $M$ that is a subdirect product of its $n$ projections $M_\pi_i =: A_i$. We will use this procedure twice. The first time we will use it to compute the classes of the subgroup $M = \ker \psi \leq F$. Here all the $A_i$ are isomorphic to a subgroup $A \leq \text{Aut}(T)$. As we want $F$-classes, however, we will eventually have to modify the procedure. This will be described in Section [3].

The second use will be to compose the classes of $G$ from the classes of $G/N$ and $F$. In this situation $n = 2$ and the factor groups are usually not isomorphic.

We consider the iterated subdirect product $M$ as a subgroup of the direct product $D = \times_{i=1}^n A_i$. If the conjugacy classes of the $A_i$ are known (this will be the case either by induction or—when computing the classes of $M \leq F$—if the $A_i$ are almost simple and so the classical random search quickly yields the classes), the classes of $D$ are simply given by the Cartesian product of the sets of class representatives.

We will first refine these classes to the $M$-conjugacy classes on $D$ and then drop those representatives which are not contained in $M$. This obviously yields the conjugacy classes of $M$. In the cases considered here, even when $M$ is the subdirect product of more than two constituents, the index of $M$ in $D$ is not overly large due to Remark [5]. So this approach is feasible.
We concentrate first on dealing with \( M \)-conjugacy. Let \( g \) be an element of \( D \). As \( M \) projects surjectively on \( A_1 \), we can find \( m_1 \in M \) such that \( g^{m_1} \pi_1 \in A_1 \) is the fixed representative of its class. Keeping this image fixed restricts further conjugation to \( C_1 := (C_{A_1}(g^{m_1})^{\pi_1})^{\pi_1} \), where the inverse \( \pi_1^{-1} \) denotes taking the full preimage. We then consider the image \( g^{m_1} \pi_2 \). By conjugation with \( m_2 \in C_1 \) we can let \((g^{m_1}m_2)^{\pi_2} \) be a fixed representative of a \( C_1 \pi_2 \)-class in \( A_2 \). Leaving the first two projections images fixed, further conjugation is restricted to \( C_2 := C_{A_2}(g^{m_1}m_2^{\pi_2})(\pi_2|_{C_1}) \). We then proceed in a similar way for the other components, finally getting a “canonical” conjugate \( g^{m_1 \cdots m_n} \) and its centralizer.

So, we can build \( M \)-class representatives component by component in the form 
\[
(r_1, r_2, \ldots, r_d),
\]
where \( r_i \) is a representative of a \( C_i := C_{i-1} \pi_i \)-class on \( A_i \). These \( C_i \)-classes are refinements of the \( A_i \)-classes: if \( r \) is a representative of an \( A_i \)-class, representatives of the \( C_i \)-classes therein are given by \( r^{s_j} \) with \( s_j \) running through a set of representatives of the double cosets \( C_{A_i}(r) \backslash A_i / C_i \). The appropriate centralizers of the form \( C_{C_i}(r^{s_j}) \) can be obtained by intersecting the \( s_j \)-conjugate of the centralizer of \( r \) with \( C_i \). Taking the preimage under \( \pi_i|_{C_{i-1}} \) then yields the next \( C_i \).

If we want classes within \( M \) to have to drop all representatives that are not contained in \( M \) (as \( M \) is closed under \( M \)-conjugation). Whenever we encounter a partial representative \( (r_1, \ldots, r_i) \) with \( i \leq n \) which is not contained in the image \( M(\pi_1, \ldots, \pi_i) \) of \( M \) when projected to the first \( i \) components, we can discard this representative; all its descendants will not lie in \( M \) and it thus will not give rise to any valid representative of an \( M \)-class on \( M \).

This leads immediately to an algorithm that yields representatives of the \( M \)-classes on \( D \).

To reduce the number of centralizer preimages to consider (and thus also the number of double cosets to be computed) we further observe that \( M \) contains \( n \) normal subgroups that act on only one component, namely
\[
L_i := \bigcap_{j \neq i} \ker \pi_j.
\]

They generate a direct product, the group \( L := \langle L_i \mid 1 \leq i \leq n \rangle \). (In the situation of Section 3, we have for example \( T^n = E \leq L \); thus \( L \) can be considered to be comparatively big.)

We can conjugate with elements of \( L \) in one component without disturbing any of the other components. Thus in our above consideration, we can replace \( C_i \) by \( \langle C_i, L \rangle \) without introducing conjugation that would change previously chosen class representatives within some already considered projection images. The immediate benefit of this is not only that \( \langle C_i, L \rangle \) is often bigger than \( C_i \) and thus fewer double cosets arise (and the computation will proceed much faster), but also that actually many of the groups \( \langle C_i, L \rangle \) for different \( C_i \)’s will coincide. This greatly reduces the number of double coset computations needed. (In the extreme case of a direct product, \( \langle C_i, L \rangle \pi_{i+1} \) will be equal to \( A_{i+1} \). Thus the double cosets degenerate to simple cosets, and our procedure just enumerates the cartesian product of class representatives.)

If we proceed in this way, of course we obtain (for example) a new \( C_2 \) as \( C_2 := C_{A_2}(g^{m_1m_2}\pi_2)^{-1}(\pi_2|_{C_1, L}) \). Thus the final stabilizer \( C_n \) is not the centralizer of the
representative, but the closure of the centralizer with \( L \). But as computing images, preimages and double cosets in general is harder than computing centralizers, this loss of having to compute the centralizer separately is made up by the gain in the number of double coset computations needed.

By taking closures with \( L \), many of the centralizers will coincide. We may take advantage of this to reduce the number of double coset computations.

The algorithm follows.

Algorithm 6. (Classes of an iterated subdirect product based on classes of the constituents) Let \( M \), \( n \), \( A_i \), \( \pi_i \), \( D \) and \( L \) be as defined previously. We denote the canonical embedding from \( A_i \) into \( D \) by \( \varepsilon_i \), so we have \( m = \prod_i m_i \varepsilon_i \) for all \( m \in M \).

Let \( \text{number}_i \) be the number of classes and \( \{r_{i,j}\} \) a set of class representatives of \( A_i \). Let \( C_{i,j} := C_{A_i}(r_{i,j}) \). This algorithm computes a set of representatives of the conjugacy classes of \( M \). (We denote lists by square brackets.)

1. [Initialization] Let \( \text{reps} := [1_M]; \text{centralizers} := [M]; \text{cent\_index} := [1]. \)
2. [Components loop] For \( i \) from 1 to \( n \) execute steps 3-17:
3. [New component] Let \( \vec{w} := (\pi_1, \ldots, \pi_i); M\text{proj} := M \vec{w} \) and \( L\text{loc} := L \pi_i \). Let \( \text{new\_reps} := [\ ]; \text{new\_cent} := [\ ]; \text{new\_cent\_index} := [\ ]; \text{cent\_images} := [\ ]; \text{cent\_img\_index} := [\ ]. \)
4. [Fuse centralizers that become the same in \( A_i \)] For \( j \) from 1 to \( |\text{centralizers}| \) execute step 5
5. [Check whether two centralizers have the same image] Let \( C := \text{centralizers}_j \pi_i \). If \( C \in \text{cent\_images} \) then let \( p \) be its position, otherwise add \( C \) to \( \text{cent\_images} \) and let \( p := |\text{cent\_images}| \).
   In both cases add \( p \) to \( \text{cent\_img\_index} \).
6. [Consider all previous centralizers] For \( j \) from 1 to \( |\text{cent\_images}| \) execute steps 7-16:
7. [Determine all representatives belonging to this centralizer image] Let \( C := \text{cent\_images}_j \). Let \( \text{select\_cen} := \{ k \mid \text{cent\_img\_index}_k = j \} \) be the set of all centralizer indices with the same (enlarged) centralizer image. For \( k \) from 1 to \( \text{number}_i \) execute steps 8-16:
8. [Double cosets] Let \( \text{dc} \) be a set of representatives of the double cosets \( C_{i,j} \backslash A_i \backslash C \). For all \( t \) from \( \text{select\_cen} \) execute steps 9-14:
9. [Continue one partial representative] Let \( \text{cen} := \text{centralizers}_j \). Let \( \text{select} := \{ t \mid \text{cent\_index}_t = t \} \) be the set of indices of those representatives whose partial centralizer is \( \text{cen} \).
   Let \( \eta := \pi_i \mid \text{cen} \), \( \text{new\_cent\_local} := [\ ] \) and \( \text{new\_cent\_local\_index} := [\ ]. \) For all \( d \) from \( \text{dc} \) execute for all \( s \) from \( \text{select} \) the steps 10-13:
10. [New representative] Let \( \text{elm} := \text{reps}_s \cdot ((r_{i,k}^d)\varepsilon_i) \). If \( \text{elm} \notin M\text{proj} \) then go to step 13 as it may not lead to elements in \( M \).
11. [Compute new centralizer] Let \( \text{new\_cen} := C \eta \cap \langle L\text{loc}, C_{i,k}^d \rangle \). If \( \text{new\_cen} \notin \text{new\_cent\_local} \) then add \( \text{new\_cen} \) to \( \text{new\_cent\_local} \) and let \( p := |\text{new\_cent\_local}| \). Otherwise let \( p \) be the position of \( \text{new\_cen} \) in \( \text{new\_cent\_local} \).
12. [Store the new element] Add \( \text{elm} \) to \( \text{new\_reps} \) and add \( p \) to \( \text{new\_cent\_local\_index} \).
14. [Centralizer preimages] Let \( \text{shift} := [\ ] \). For each \( l \) from 1 to \( |\text{new\_cent\_local}| \) let \( P \) be the preimage of \( \text{new\_cent\_local}_l \) under \( \eta \). If \( P \notin \text{new\_cent} \) then add
1640 ALEXANDER HULPKE

P to new_cent and let shift := |new_cent|. Otherwise assign the position of
P in new_cent to shift.
15. [Move local centralizer indices to global] For each \( l \in new_cent\) local_index
add shift\(_l\) to new_cent_index.
16. End the loops from step 6,7 and 8.
17. [End of component loop] Replace centralizers by new_cent, cent_index by
new_cent_index, and reps by new_reps. End the loop of step 2.

When the algorithm terminates, the list reps contains representatives of the
classes of \( M \).

Remark 7. Of course – as seen before – the centralizer of the representatives can
be computed componentwise as well by keeping true centralizers \( C^i \) besides the \( C_i \).

We note that in the case of a transitive action on the \( A_i \) the class representatives
\( r_{i,j} \) and centralizers \( C_{i,j} \) only need to be computed once.

Remark 8. If \( M \) is a factor group of a larger group \( G \) (this happens when lifting the
classes via subdirect products as described in Section 3) and if we don’t yet have a
faithful representation of \( M \), we can instead of elements and subgroups of \( M \) work
with their representatives, respectively full preimages in \( G \). Tests for equality then
must take place modulo the normal subgroup.

Remark 9. For two components, the test \( elm \in Mproj \) in step 10 also can be re-
placed easily by checking whether \( (\text{reps}_s)_{\pi_1} \) and \( (r_{i,k}^d) \) both project correctly to the
common factor group that determines the subdirect product. (For more components,
subdirect products become too complicated for such a test.)

Furthermore, the algorithm does not really compute in the first component, but
just takes preimages of its classes/centralizers.

5. Fusion under component permutation

We now return to the situation of Section 3. Let \( F \leq \text{Aut}(T) \rtimes S_n \), with \( M = \ker \psi \triangleleft F \). We aim to describe the conjugacy classes of \( F \). This will be done in
two steps: first we will show how to obtain representatives of the \( F \)-classes on \( M \).
This will be explained in this section.

In the second step (which will be carried out in Section 8) we will then describe
how to get the remaining classes outside \( M \).

The \( F \)-classes of elements in \( M \) are of course unions of \( M \)-classes. We therefore
could simply compute the \( M \)-classes first and then fuse them under the action of \( F \).
In many cases, however, a substantial fusion takes place; this approach would first
compute many \( M \)-classes and then do much work fusing them to comparatively
few \( F \)-classes. We will therefore modify Algorithm 6 to construct representatives
of \( F \)-classes in the first place by taking care of the component permutation induced
by \( F \).

We will suppose that we have obtained \( F \) by the action of \( G \) on a chief factor
of \( G \). So \( F \) acts transitively on the projections \( A_i \) of \( M \), and therefore each \( A_i \)
is isomorphic to a given group \( A \leq \text{Aut}(T) \). The action of \( F \) might further fuse
classes of one \( A_i \); this can be read off from the action of \( S := \left( \text{Stab}_{F\psi}(1) \right) \psi^{-1} \) on the
classes of \( A_1 \).
Remark 10. If $M$ is complemented in $F$ (for example by the Schur-Zassenhaus theorem), no such fusion might take place, because the further action of $F$ on $M$ is a pure component permutation.

To keep track of possible fusion, we now assign labels (which we call colours) to the classes of $A$, giving the same colour to classes if and only if they fuse under $S$. Furthermore we assign to every element of $D$ a list of labels (which we call a colour bar) by listing the colours of the components classes:

$$(a_1, \ldots, a_n) \mapsto (\text{ColourOfClass}(a_1), \ldots, \text{ColourOfClass}(a_n)).$$

The permutation action of $F$ will permute the colours in a bar while $M$ stabilizes the bars. To obtain representatives of the $F$-classes on $M$ it is thus sufficient to consider only such class representatives whose colour bars are from a set of representatives of the colour bars under the permuting action of $P := F \psi$. This in turn allows us to weed out all the representatives obtained in Algorithm 6 whose colour bars are not representatives.

We shall first consider the problem of computing representatives of the $P$-action on colour bars. In Section 7 we will then use the bar representatives to compute class representatives.

6. Action on colour bars

We observe that we can parametrize colour bars by the number of colours contained, by the frequency of the colours occurring, and finally by the contained colours themselves. (For example for $n = 5$ one class might consist of the bars containing three different colours, the first colour three times, the other colours both once; the colours being red, green and blue.) This parameterization is well suited to component permutation, because we can first compute representatives by number and frequency of the colour and can fill the actual colours in afterwards in arbitrary combinations.

If the acting group is the symmetric group this completely parameterizes the representatives, as we can sort each bar by “brightness” (i.e. an arbitrary total ordering) of the colours and the sorting permutation is contained in the symmetric group.

To deduce $P$-representatives from this we observe that if $c$ is a colour bar and $S = \text{Stab}_{S_n}(c)$, all bars containing the same colours and the same frequencies are in bijection to the right cosets $S\backslash S_n$, as they form just one $S_n$-class. (Note that $S$ is just a direct product of symmetric groups.) The $P$-orbits are in correspondence with the double cosets $S\backslash S_n/P$; we can obtain representatives by mapping $c$ with a set of representatives for the double cosets.

This leads to the following algorithm:

Algorithm 11 (Representatives of colour bars). Let $P$ be a permutation group on $n$ points and $\text{colours}$ a list of colours. This algorithm computes a set of representatives of the colour bars under the permuting action of $P$.

1. [Initialization] Let $\text{allowed-colourbars} := []$.
2. [Loop over the number of different colours in the bar] For $\text{number}$ from 1 to $\min(n, \text{number})$ execute steps 3–6 (we construct bars with exactly $\text{number}$ colours).
3. [Colour combinations] Let \( \text{colour}_{\text{comb}} \) be a set of the \( \binom{\text{colours}}{\text{number}} \) combinations of \text{number} entries from \text{colours}. For \text{pattern} running through a set of the ordered partitions of \( n \) with \text{number} cells execute steps 4–6 (the \text{pattern} indicates how often which colour will occur).

4. [Compute pattern stabilizer and double cosets] Let 
\[
\text{expanded}_{\text{pattern}} := [1, \ldots, 1, 2, \ldots, 2, \ldots, \text{number}, \ldots, \text{number}]_1 \text{pattern}_1 \text{pattern}_2 \ldots \text{pattern}_\text{number}.
\]

Let \( S \) be the stabilizer in \( S_n \) of \( \text{expanded}_{\text{pattern}} \). (This is the stabilizer of all bars that arrange their colours according to this pattern. Its computation is trivial as it is just a direct product of symmetric groups.)

Let \( \text{reps} \) be a set of representatives of the double cosets \( S \backslash S_n / P \). For each \( r \in \text{reps} \) execute step 5:

5. [Consider further permutations] Let \( \text{permuted}_{\text{pattern}} \) be the image of \( \text{expanded}_{\text{pattern}} \) under component permutation by \( r \). For each \( \text{comb} \) from \( \text{colour}_{\text{comb}} \) add the bar 
\[
[\text{comb}(\text{permuted}_{\text{pattern}}_1), \ldots, \text{comb}(\text{permuted}_{\text{pattern}}_\text{number})]
\]

6. End the loops.

At this point, \( \text{allowed}_{\text{colourbars}} \) is a set of representatives.

We remark that in general \( n \) (and thus \( P \)) will be very small. Therefore this algorithm’s runtime will be negligible in the overall cost and the potentially exponential algorithm need not be worried about.

The stabilizers of the colour bars in \( P \) are obtainable in step 5 as \( S^r \cap P \).

**Definition 12.** We call those bars **unlucky** whose stabilizers are actually larger than \( M \).

### 7. Representatives of the inner classes

In this section we will use the representatives of the colour bars to obtain representatives of the classes of \( M \) under \( F \).

We assume that we have computed the fusion of \( A \)-classes under the action of 
\[
\text{Stab}_{F^F}^{-1}(1) \psi
\]
and thus obtained a list of colours and assignment of colours to classes. We also assume knowledge of a set of representatives of the colour bars under action of \( P = F^\psi \).

When constructing representatives of \( F \)-classes, representative tuples from \( M \) with a colour bar not in the list of bar representatives may be discarded. (There is an image under \( F \) whose colour bar is in the list; we will also construct such a representative. As \( M \) fixes all bars, this does not interfere with any labelling scheme for \( M \)-representatives.)

Thus in Algorithm 6 for each partial representative tuple \( (r_1, \ldots, r_{i-1}) \) only representatives of those classes need to be added in the \( i \)-th position, whose colour is equal to an \( i \)-th colour in a bar starting with Colourbar\((r_1, \ldots, r_{i-1})\). Thus by attaching to each (partial) representative its (partial) colour bar we can simply restrict the number of representatives constructed while still covering all \( F \)-classes on \( M \). The test for the permitted colours can take place in step 8 of Algorithm 6. 


as at this point the colour bar of the image is already determined. We modify Algorithm 6 as follows:

**Algorithm 13** (Classes of a subdirect product under further component fusion). We assume that \( \text{ColourOfClass} \) gives the colour for a class and that the list \( \text{allowed}\_\text{colourbars} \) is the result from Algorithm 11. Furthermore we will keep a list \( \text{colourbar} \) to give the (partial) colourbars of the representatives.

3'. [New component] Let \( \pi := (\pi_1, \ldots, \pi_i) \); \( M_{\text{proj}} := M \pi \) and \( L_{\text{loc}} := L \pi_i \). Let \( \text{new}\_\text{reps} := [\ ]; \text{new}\_\text{cen} := [\ ]; \text{new}\_\text{cen}\_\text{index} := [\ ]; \text{cent}\_\text{images} := [\ ]; \text{cent}\_\text{img}\_\text{index} := [\ ]; \text{new}\_\text{colourbar} := [\ ].

7'. [Determine all representatives belonging to this centralizer image] Let \( C := \text{cent}\_\text{images}_j \). Let \( \text{select}\_\text{cen} := \{ k \mid \text{cent}\_\text{img}\_\text{index}_k = j \} \) be the set of all centralizer indices with the same (enlarged) centralizer image. Let \( \text{select} := \{ k \mid \text{cent}\_\text{index}_k \in \text{select}\_\text{cen} \} \) be the set of indices of those representatives that might be extended here.

7a. [Determine the addable colours] Let \( \text{possible}\_\text{colours} := \{ \} \). For \( k \in \text{select} \) let \( \text{bar} := \text{colourbar}_k \). Let

\[
\text{potential}\_\text{bars} := \{ b \in \text{allowed}\_\text{colourbars} \mid b_{[1, \ldots, i-1]} = \text{bar} \}
\]

and let

\[
\text{possible}\_\text{colours} := \text{possible}\_\text{colours} \cup \{ b_i \mid b \in \text{potential}\_\text{bars} \}.
\]

7b. [Run through the classes with correct colours] For those \( k \) from 1 to \( \text{number} \) whose colour is in \( \text{possible}\_\text{colours} \) execute steps 8-16.

9'. [Continue one partial representative] Let \( \text{cen} := \text{centralizers}_k \). Let \( \text{select} := \{ t \mid \text{cent}\_\text{index}_t = k \} \) be the set of indices of those representatives whose partial centralizer is \( \text{cen} \).

Let \( \eta := \pi_i(\text{cen}) \); \( \text{new}\_\text{cen}\_\text{local} := [\ ] \) and \( \text{new}\_\text{cen}\_\text{local}\_\text{index} := [\ ] \). For all \( d \) from \( \text{dc} \) execute for all \( s \) from \( \text{select} \) the steps 9a-13:

9a. [Test whether this colour may be added here] Let \( \text{bar} := \text{colourbar}_s \cup [\text{ColourOfClass}(k)] \).

If there is no bar \( c \in \text{allowed}\_\text{colourbars} \) for which \( c_{[1, \ldots, d]} = \text{bar} \) then go to step 13.

12'. [Store the new element] Add \( \text{elm} \) to \( \text{new}\_\text{reps} \), \( \text{bar} \) to \( \text{new}\_\text{colourbar} \) and add \( p \) to \( \text{new}\_\text{cen}\_\text{local}\_\text{index} \).

17'. [End of component loop] Replace \( \text{cent}\_\text{index} \) by \( \text{new}\_\text{cent}\_\text{index} \), \( \text{colourbar} \) by \( \text{new}\_\text{colourbar} \), and \( \text{reps} \) by \( \text{new}\_\text{reps} \). End the loop of step 2.

Finally, among the representatives obtained by the modified algorithm, some further fusion by \( F \) may take place, but only among those representatives with the same colour bars. Two representatives \( a, b \in M \) with the same colour bar \( c \) that are conjugate in \( F \) must be conjugate already in \( \text{Stab}_F(c) \). So conjugacy tests actually only need to be performed among those representatives with unlucky colour bars. In this situation the conjugacy test takes place in \( \text{Stab}_F(c) \), which is often a smaller group than \( F \):

18. [Prepare for further \( F \)-fusion] Let \( \text{new}\_\text{reps} := [\ ]; \) For every \( \text{bar} \in \text{colourbar} \) execute step 19

19. [Fuse among classes with colour bar \( \text{bar} \)] Let \( \text{cand} \) be the set of those representatives in \( \text{reps} \) which have colour bar \( \text{bar} \). Using conjugacy tests, compute
a set \( \text{fuse} \) of \( \text{Stab}_F(\text{bar}) \)-representatives among the elements in \( \text{cand} \). Append \( \text{fuse} \) to \( \text{new_reps} \).

At the end of this loop \( \text{new_reps} \) contains representatives of the \( F \)-classes of elements in \( M \).

8. Obtaining the outer classes

We now can compute the \( F \)-classes of elements in \( M \). What remains to be done is the classes of elements of \( F \) not contained in \( M \), the \emph{outer classes}.

In contrast to the classes within \( M \), \( F \) usually has very few outer classes (see the examples in Section 11). Some of these outer classes can be relatively small, however, and so a pure random search is not feasible.

To obtain these outer classes, we recall the general idea of [15]: Let \( \text{M} := F/\text{M} \), and \( \psi : F \to P \) the natural homomorphism. Let \( r \) be the representative of a class of \( P \). Then representatives of those classes of \( F \) that map under \( \psi \) into the class of \( r \) can be found as representatives of the \( Z := (\text{C}_P(r\psi))^{-1} \)-orbits on the coset \( r\text{M} \). For \( z \in Z \) we have

\[
(rm)^z = r^zm^z = r[r, z]m^z
\]

with \( [r, z] \in M \), this induces an action on \( M \) by

\[
m \mapsto [r, z]m^z.
\]

Here, in general, however, \( M \) is too big to pursue this actual action.

As in Section 4 we will now first restrict the conjugating group acting on \( D \) from \( Z \) to \( M \leq Z \). As we may embed \( F \) in \( \text{Aut}(T) \), the action of \( M \) on \( M \) is a restriction of the action of \( M \) on \( D \) given by

\[
d \mapsto [r, m]d^m.
\]

Let \( M = (m_1, \ldots, m_n) \in M \), \( d = (a_1, \ldots, a_n) \in D \), \([r, m] := ([r, m]_1, \ldots, [r, m]_n)\), the cartesian decomposition. Then the action (14) becomes

\[
d \mapsto [r, m]d^m = ([r, m]_1a_1^{m_1}, \ldots, [r, m]_na_n^{m_n}).
\]

This action thus is the composition of the action of the components \( A_i \) by

\[
a_i \mapsto [r, m]_ia_i^{m_i}.
\]

Now, to obtain representatives of the \( M \)-orbits on \( M \) under this action, we again proceed componentwise: we first compute the \( M \)-orbits on \( A_1 \). Then for each representative \( r_1 \) we compute the \( \text{Stab}_M(r_1) \) orbits on \( A_2 \), and so on. We thus obtain orbit representatives in the form \((r_1, \ldots, r_n)\). As above, to obtain the orbits on \( M \) we have to discard those representatives that are not contained in \( M \) (as \( r \) normalizes \( M \), the action (14) fixes \( M \) set-wise), and we can simply discard all partial representatives \((r_1, \ldots, r_i)\) which are not contained in the projection image \( M(\pi_1, \ldots, \pi_i) \).

As long as the size of the \( A_i \) is small we can simply run a standard orbit/stabilizer algorithm. For a large \( A_i \) (as in the example in the introduction for which \( A_i \cong S_{11} \)), however, this is not possible as we cannot store all elements. A standard technique in such cases is the use of \emph{tadpoles} [7]. Let \( U \) be a subgroup of the acting group. Whenever elements are encountered a “standard” representative under action of \( U \) is computed (the iterative process usually used here gives rise
to the name “tadpole”) and only this representative is stored. This permits us to reduce the number of elements to store roughly by $|U|$. 

In our situation we will take $U$ to be a subgroup that centralizes $r$. (Any larger group would change $r$ and thus leave the operations domain.) Then for $U$ the action (15) reduces to a simple conjugation action, and we can take the “standard” $U$-representatives to be computed representatives of the $U$-classes on $A_r$. (Again, as above, these classes can be computed by conjugating class representatives $a$ with representatives of the double cosets $C_{A_r}((a)A_i/U)$. This permits us to compute the $U$-classes a priori and to simplify the orbit algorithm to fusion of the $U$-orbits. Section 12 explains how to proceed to obtain representatives of the $M$-orbits for these.

To get the largest possible subgroup we take $U$ in each step (when computing possible representatives in the $(i+1)$-th component with entries $r_1, \ldots, r_i$ in the prior components) to be the intersection of the centralizer $C_F(r)$ of $r$ with the acting group $\text{Stab}_M(r_1, \ldots, r_i)$. Note that we have $|M|$ choices for $r$. To minimize the number of $U$-orbits, we try to maximize $C_F(r)$, which is fulfilled if the order of $r$ is minimal possible. Let $o$ be the order of $r \psi$. Then we need $r^o = ()$, which means that $\langle r \rangle$ intersects trivially with $M$.

**Lemma 16.** For each cycle in the action of $M \psi$ on the components pick one component index. Let $I$ be the set of these indices. If there is an $m \in M$ with $m \pi_i = (r^o) \pi_i$ for all $i \in I$ and $m \pi_j = ()$ for all $j \notin I$, then $\hat{r} := r/m$ is an element with $\hat{r} \psi = r \psi$ and $\hat{r}^o = ()$.

**Proof.** By considering the orbits on the components separately, we can assume without loss of generality that $r$ acts transitively and $I = \{1\}$. Let $j$ be a point from the first component. Then $j^{(r^o)}$ ($l < o$) is in a component on which $m$ does act trivially. Accordingly

$$j^{(r^o)} = j^{(rm^{-1} \cdots rm^{-1})} = j^{(r^o/m)} = j$$

by the definition of $m$, as $j$ is in the first component. So $\hat{r}^o$ acts trivially on the first component. As every point $k$ is the image $k = j^{(\hat{r}^l)}$ ($l < o$) for a $j$ in the first component, we have similarly

$$k^{(\hat{r}^o)} = (j^{(\hat{r}^l)})^{(\hat{r}^o)} = j^{(\hat{r}^l+o)} = (j^{(\hat{r}^o)})^{(\hat{r}^l)} = j^{(\hat{r}^i)} = k,$$

and thus $\hat{r}^o = ()$. \hfill \Box

If no suitable element of $M$ can be found, we simply select an $r$ with a large centralizer.

To continue in the next component we also need the stabilizer in $U$ of the orbit representative (which will be the intersection of $U$ with the stabilizer mentioned above). This stabilizer is simply (the full preimage of) the centralizer in $U$.

**Remark 17.** If $U$ is not normal in $G$ and several orbits of $U$ get fused, however, these centralizers can be of different sizes. Therefore we sort the orbits of $U$ in ascending size and pick always the smallest one as representative for starting the $G$-orbit. This allows us to continue in the next component with as large a $U$ as possible.

Finally, we have to fuse the $M$-orbits to $Z$-orbits. As by Remark 10 $[F : M]$ is small, also $[Z : M]$ is small and thus not much fusion will take place. We also
observe that in general $M$ is already quite big and has large orbits; thus only few classes have to be fused. Therefore this fusion test is done in a straightforward way by testing conjugacy of the elements obtained so far. (Obviously we can restrict this fusion test to elements with the same cycle structure and centralizer orders.)

This finishes the description of the non-abelian lifting step which computes the classes of $G$, provided the classes of $G/N$ are known. To summarize: Assigning $F := G/C_G(N)$, we first compute the $F$-classes of $M$. This is done by Algorithm 13 and an additional fusion among representatives with the same unlucky colour bars. Afterwards we compute the $F$-classes outside $M$ as in Section 8. Finally, we combine the classes of $G/N$ and those of $F = G/C$ to classes of $G = (G/N) \cdot F$ using Algorithm 6. We then proceed inductively for the next chief factor.

### 9. An Orbit-Fusing Algorithm

(This section uses its own notation to simplify variable names.) Let $G$ be a (finite) group acting on $\Omega$ via the operation $\mu : \Omega \times G \to \Omega$. Furthermore let $U \leq G$ and $\nu := \mu|_U$ be the restriction of the operation. We assume that we know representatives $\{\omega_i\}$ of the $U$-orbits on $\Omega$ and their stabilizers, that we can find, for any $\delta \in \Omega$, in which $U$-Orbit it lies, and that we can compute $u \in U$ with $\delta^u = \omega_i$ for the suitable $i$. We want to obtain representatives and stabilizers of the $G$-orbits.

As we already know the $U$-orbits, we use a slightly modified orbit algorithm that will always add a full $U$-orbit. However we want to avoid mapping all elements of $\Omega$ with generators of $G$ but try to restrict ourselves (as far as possible, impossible in general) to mapping only the $\omega_i$: we store which $U$-orbits $\Im \omega^u_i$ are contained in the $G$-orbit. If we have mapped all $\omega_i$ with all generators of $G$ and the product of orbit length (the sum of the $|\Im \omega^u_i|$) and stabilizer size is smaller than $|G|$, we did not yet reach the full orbit or full stabilizer. In this situation we start mapping other points $\Im \omega^u_i (u \in U)$ until this condition is fulfilled.

#### Algorithm 18. (Fuse $U$-orbits under the action of $G \geq U$)

1. [Initialization] Let $\text{orb} := [\omega_1]$, $\text{stab} := \text{Stab}_U(\omega_1)$ and $\text{trans} := [()]$, let $\text{iterate} := \text{false}$ and $\text{pos} := 1$
2. [Orbit loop] Let $\omega := \text{orb}[\text{pos}]$. For every generator $\text{gen}$ of $G$ execute steps 3–7.
3. [Pick up further elements to map] If $\text{iterate}$ then let $u$ be a random element of $U$ and set $\text{gen} := u \cdot \text{gen}$.
4. [Compute image and its canonical $U$-representative] Compute $\Im := \mu(\omega, \text{gen})$. Let $j$ be the index for which $\omega_j$ is in the $U$-orbit of $\Im$. Compute $v \in U$ with $\nu(\omega_1, v) = \omega_j$ and let $\text{gen} := \text{gen} \cdot v$. Let $\text{transgen} := \text{trans}[\text{pos}] \cdot \text{gen}$. (Now $\text{transgen}$ maps $\omega_1$ to $\omega_j$.)
5. [Did we extend the orbit?] If $\omega_j \in \text{orb}$ then go to step 6. Otherwise add $\omega_j$ to $\text{orb}$ and add $\text{transgen}$ to $\text{trans}$. For each generator $s$ of $\text{Stab}_U(\omega_j)$ let $\text{stab} := (\text{stab}, \text{transgen} \cdot s \cdot \text{transgen}^{-1})$. Go to step 7.
6. [Extend stabilizer] Assume that $\text{orb}[i] = \omega_j$. Let $\text{stab} := (\text{stab}, \text{transgen} \cdot \text{trans}[i]^{-1})$
7. End the $\text{gen}$-loop of step 2
8. [New orbit index] Increment $\text{pos}$. If $\text{pos} > |\text{orb}|$ then set $\text{pos} := 1$ and $\text{iterate} := \text{true}$.
9. [Test for full orbit/stabilizer] If $|\text{stab} \cdot \sum_{\omega \in \text{orb}} |\Im \omega^u_i| < |G|$ then go to step 2.
When the algorithm terminates, \( orb \) contains those \( \omega_i \) whose \( U \)-orbits form the \( G \)-orbit and \( \text{stab} = \text{Stab}_G(\omega_1) \). Iterated application gives all \( G \)-orbits.

**Proof.** Observe that the elements added in steps 5 and 6 all stabilize \( \omega_1 \). By the orbit theorem the algorithm will terminate if we have found the full orbit and full stabilizer. By iterating we will finally (if the random function is really random) have mapped all orbit elements, so to show correctness it is sufficient to prove that the stabilizer generators added in step 6 finally will generate all Schreier generators of the stabilizer.

This is seen easily if we define the transversal \( T \) (which gives rise to all these Schreier generators) based on the computed transversal \( \text{trans} \). That is, every element of \( T \) is of the form \( t u \) with \( t \in \text{trans} \) and \( u \in U \). The random element \( u \) in step 3 will finally give rise to all elements of \( T \).

Suppose that we map \( t \cdot u \in T \) with the generator \( gen \) and that the transversal element for the image is \( t' u' \). (We have \( t = \text{trans}\{\text{pos}\} \) and \( t' = \text{trans}\{\text{pos}\} \).) This gives rise to the Schreier generator \( s := t \cdot u \cdot \text{gen} \cdot (u')^{-1}(t')^{-1} \). On the other hand, the element \( v \) computed in step 4 will have \( t \cdot u \cdot \text{gen} \cdot v \) mapping the representative to an image corresponding to the transversal element \( t' \). In step 6 then we add the element \( s' := t \cdot u \cdot \text{gen} \cdot v \cdot (t')^{-1} \) to the stabilizer. As \( u' v \in \text{Stab}_U(\omega_1) \) and thus \( v^{-1}(u')^{-1} = s_1 \cdot \ldots \cdot s_l \) for suitable generators \( s_i \) of \( \text{Stab}_U(\omega_1) \), we have that

\[
\begin{align*}
  s &= t \cdot u \cdot \text{gen}(v \cdot v^{-1})(u')^{-1} \cdot (t')^{-1} = t \cdot u \cdot \text{gen} \cdot v((t')^{-1} t')(v^{-1} \cdot (u')^{-1})(t')^{-1} \\
  &= (t \cdot u \cdot \text{gen} \cdot v \cdot (t')^{-1} t'(s_1 \cdot \ldots \cdot s_l))(t')^{-1} = s' \prod_{i=1}^l s_i^{(t')^{-1}}.
\end{align*}
\]

The \( s_i^{(t')^{-1}} \) have been added already in step 5. So we have finally \( s \in \text{stab} \), which is what had to be proved.

When picking the random element in step 3 it is sufficient to pick a \( u \) which is not in \( U \cap U(\text{gen}^{-1}) \), because otherwise \( u \cdot \text{gen} = \text{gen} \cdot \tilde{u} \) with \( \tilde{u} \in U \) and so the image will not lie in a different \( U \)-orbit. If no such elements exist, the generator can be ignored when iterating.

**Remark 19.** As we know the size of the acting group and the orbit lengths, we may sometimes complete orbits without having computed all images: the length of the orbit must divide the order of the acting group. Suppose we have computed a partial orbit \( \text{orb} \) of length \( l = \sum \omega_i \in \text{orb} \lfloor \omega_i^U \rfloor \) and stabilizer subgroup \( \text{stab} \). If for the smallest divisor \( d \) of \( |G| \) which is not smaller than \( l \) (\( d = \min\{e \mid |G| \mid e \geq d\} \)) we have that \( d \cdot |\text{stab}| = |G| \), we know that \( \text{stab} \) is the stabilizer and that the full orbit is of length \( d \). If there is only one possibility to add up the sizes of remaining \( U \)-orbits to obtain \( d - l \) these orbits have to be added to \( \text{orb} \) to obtain the full \( G \)-orbit. This can be particularly helpful if one small \( U \)-orbit is missing.

Usually the algorithm will finish with none or very few iterations and terminate much quicker than when running through all \( U \)-images \( v(\omega, u) \) in a more orderly way.

If some of the \( U \)-orbits are small in average, it is relatively unlikely that they will be hit. Therefore it is preferable to pick \( \omega_1 \) to be the representative with the smallest \( U \)-orbit. As a side effect, this permits one to start with the largest possible stabilizer and is consistent with the choice needed by Remark 17.
10. **Implementational issues**

The whole algorithm now stands as described in the introduction. We compute a series of normal subgroups with elementary factors: \( G = N_0 > N_1 > \cdots > N_r = \{1\} \), and inductively compute the classes of \( G/N_i \) from those of \( G/N_{i-1} \). For an abelian \( N_{i-1}/N_i \) the algorithm of [15] is used, and in the non-abelian case the method described in Sections 3 to 8 is used.

10.1. **Factor representations.** The algorithm will require us to compute in factor groups. Following Remark 8 however, it is often not necessary to construct a concrete representation of the factor groups, but computation can proceed using coset representatives of full preimages of subgroups.

For the factor groups for which we need to compute a representation, however, small faithful permutation representations can easily be found:

The factor groups \( F = G/C \) are obtained in Section 3 by action on chief factors. As shown there, \( F \) can be embedded into \( \text{Aut}(T) \cdot S_n \). Using a faithful permutation representation for \( \text{Aut}(T) \) (such a representation is of relatively small degree, as \( T \) is simple non-abelian), we obtain a moderate degree permutation representation for the wreath product and thus for the factor \( F \).

We now consider the lifting situation of Section 3 (\( G \) being a subdirect product of \( G/N \) and \( F \)) with \( G \) being a factor group of a (probably larger) group \( H \). We thus want to defer computations in \( G \) to computations in \( H \). As we can find a good permutation representation for \( F \), the full preimage of \( C = C_G(N) \) in \( H \) is just the kernel of the homomorphism \( H \to F \), and is computable as such. We also may assume by induction that we already have class representatives and full preimages in \( H \) of the centralizers in \( G/N \). Then (by Remark 8) we can use these preimages in Algorithm 6 and obtain representatives in \( H \) of class representatives for \( G \).

Similarly we obtain (preimages in \( H \) of) the centralizers in \( G \) by Remark 7 from the centralizers in \( G/N \) and \( F \). Furthermore we may assume \( F \) to be the second component of the subdirect product. Then by Remark 9 all computations for the algorithm will take place in \( F \), which has a good permutation representation.

This yields representatives for the classes and preimages of the centralizers, required inductively if \( G \) becomes a \( \hat{G}/\hat{N} \) in the next step.

For abelian chief factors, the methods from [15] mainly require us to compute the affine action on the vector space. This can be computed without a factor representation by identifying cosets.

10.2. **Choice of the normal series.** The choice of the normal series can affect the algorithm’s performance. In general the series should be chosen to have in each non-abelian step as small as possible a common factor group in the subdirect product of \( G/N \) with \( F \). (The best case would be a direct product that is factor size 1, for which the classes are simply obtained as a cartesian product.)

Conversely, it might happen (especially in the first steps of the algorithm) that \( C = 1 \) and thus \( G/N \) is isomorphic to a factor of \( F \). In this situation, of course, the classes of \( G \) are just the classes of \( F \) and \( G/N \), and the fusion step is discarded.

11. **Examples**

The described algorithm has been implemented by the author in GAP 4 [9]. We give some results of this experimental implementation in Table 1. All runtimes are seconds on a 200MHz Pentium Pro machine running under Linux. The nonstandard
usually is small compared with the total number of classes, but some classes are very
independent of it. Then the results of the computation do not have a solvable normal
subgroup. Therefore all other examples selected do not have a solvable normal
subgroup.

Table 1. Results of conjugacy class computations

<table>
<thead>
<tr>
<th>Group F</th>
<th>[F]</th>
<th>[F:M]</th>
<th>C</th>
<th>Out</th>
<th>part</th>
<th>timestd</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_8 \times S_3$  [ \langle C_{02} \rangle ]</td>
<td>$2^{17} \cdot 3^2 \cdot 5 \cdot 7$</td>
<td>240</td>
<td>111</td>
<td>287</td>
<td>122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_8 \times S_3$  [ \langle F_{22} \rangle ]</td>
<td>$2^{17} \cdot 3^3 \cdot 5$</td>
<td>106</td>
<td>1920</td>
<td>401</td>
<td>355</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{1}{2}[S_3^2] \times V_4 \coprod A_6 \cdot V_4$  [ $2^{15} \cdot 3^2 \cdot 5^2$</td>
<td>30</td>
<td>1591</td>
<td>1161</td>
<td>708</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{1}{2}[PGL_2(9^3)] \cdot S_3$  [ (impr.) ]</td>
<td>$2^{12} \cdot 3^7 \cdot 5^3$</td>
<td>30</td>
<td>218</td>
<td>86</td>
<td>139</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{1}{2}[PGL_2(9^3)] \cdot S_3$  [ (prd.) ]</td>
<td>$2^{12} \cdot 3^7 \cdot 5^3$</td>
<td>1000</td>
<td>218</td>
<td>194</td>
<td>180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{1}{2}[S_3^2] \cdot V_4 \coprod S_3^2 \cdot S_3$  [ (impr.) ]</td>
<td>$2^{17} \cdot 3^7 \cdot 5$</td>
<td>30</td>
<td>6115</td>
<td>—</td>
<td>2458</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_{12} \cdot V_4 \coprod A_6 \cdot V_4 \coprod S_3 \cdot V_4$  [ $2^{17} \cdot 3^9 \cdot 5 \cdot 7$</td>
<td>43</td>
<td>2286</td>
<td>2396</td>
<td>1177</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column “deg” gives the degree of the permutation representation used, “C” gives the
total number of classes, column “timestd” is the runtime for the standard conjugacy
class algorithm following [3, 20] as implemented in GAP 4, and column “time” the
runtime for the author’s implementation. A “—” indicates that the computation did not
finish in three times the time used by the other algorithm.

Table 2. Results for subgroups of wreath products

<table>
<thead>
<tr>
<th>Group F</th>
<th>[F]</th>
<th>[F:M]</th>
<th>C</th>
<th>Out</th>
<th>part</th>
<th>timestd</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2}[S_3^2] \times S_4$  [ $2^{16} \cdot 3^5 \cdot 7^2 \cdot 11^2$</td>
<td>22</td>
<td>2</td>
<td>874</td>
<td>58</td>
<td>$2.5 \cdot 10^{-8}$</td>
<td>1061</td>
<td>363</td>
</tr>
<tr>
<td>$\frac{1}{2}[S_3^2] \times S_4$  [ $2^{16} \cdot 3^5 \cdot 7^2 \cdot 11^2$</td>
<td>22</td>
<td>2</td>
<td>868</td>
<td>54</td>
<td>$1.3 \cdot 10^{-6}$</td>
<td>1026</td>
<td>944</td>
</tr>
<tr>
<td>$\frac{1}{2}[S_3^2] \times S_5$  [ $2^{17} \cdot 3^7 \cdot 5^2$</td>
<td>46</td>
<td>2</td>
<td>1600</td>
<td>74</td>
<td>$1.3 \cdot 10^{-7}$</td>
<td>4248</td>
<td>2914</td>
</tr>
<tr>
<td>$\frac{1}{2}[A_4 \times 2] \cdot S_4$  [ $2^{17} \cdot 3^7 \cdot 5^2$</td>
<td>20</td>
<td>24</td>
<td>118</td>
<td>68</td>
<td>$3.6 \cdot 10^{-5}$</td>
<td>66</td>
<td>70</td>
</tr>
<tr>
<td>$\frac{1}{2}[A_5 : 2] \cdot S_3$  [ $2^{12} \cdot 3^4 \cdot 7^3$</td>
<td>24</td>
<td>6</td>
<td>135</td>
<td>50</td>
<td>$2.9 \cdot 10^{-4}$</td>
<td>64</td>
<td>66</td>
</tr>
<tr>
<td>$\frac{1}{2}[A_5 : 2] \cdot D(4)$  [ $2^{12} \cdot 3^4 \cdot 7^3$</td>
<td>24</td>
<td>8</td>
<td>1645</td>
<td>513</td>
<td>$1.5 \cdot 10^{-9}$</td>
<td>2282</td>
<td>1158</td>
</tr>
<tr>
<td>$A_7 \cdot C_4$  [ $2^{17} \cdot 3^4 \cdot 5^4 \cdot 7$</td>
<td>24</td>
<td>8</td>
<td>1728</td>
<td>63</td>
<td>$5.2 \cdot 10^{-8}$</td>
<td>1330</td>
<td>428</td>
</tr>
<tr>
<td>$\frac{1}{2}[A_7 : 2] \cdot C_3$  [ $2^{17} \cdot 3^4 \cdot 5^4 \cdot 7$</td>
<td>24</td>
<td>8</td>
<td>1728</td>
<td>63</td>
<td>$5.2 \cdot 10^{-8}$</td>
<td>1330</td>
<td>428</td>
</tr>
<tr>
<td>$\frac{1}{2}[S_7 : 2] \cdot C_5$  [ $2^{17} \cdot 3^4 \cdot 5^4 \cdot 7$</td>
<td>24</td>
<td>8</td>
<td>1728</td>
<td>63</td>
<td>$5.2 \cdot 10^{-8}$</td>
<td>1330</td>
<td>428</td>
</tr>
<tr>
<td>$\frac{1}{2}[PSL_2(11)] : 2 \cdot C_3$  [ $2^{17} \cdot 3^4 \cdot 5^4 \cdot 7$</td>
<td>24</td>
<td>8</td>
<td>1728</td>
<td>63</td>
<td>$5.2 \cdot 10^{-8}$</td>
<td>1330</td>
<td>428</td>
</tr>
</tbody>
</table>

In addition to the data given in Table 1, column “Out” gives the number of classes
outside M, and “part” the ratio of the smallest outer class within all outer classes.

names refer to constructions as explained in [6]. If solvable normal subgroups exist,
the use of homomorphic images always improves the performance. The table gives
only one example of this kind, as the behaviour of other such groups does not vary
much from the given one and as the method used has been well-known for a couple
of years; extensive runtime studies are given in [5]. Therefore all other examples
selected do not have a solvable normal subgroup.

The group $\frac{1}{2}[PGL_2(9)] \cdot S_3$ is the same group isomorphism type given in two
different permutation representations (imprimitive and product action). While the
standard algorithm depends substantially on the degree of the representation, the
new algorithm (which performs all hard calculations in the factor group F for which
a small degree permutation representation is computed) is fairly independent of it.

The building blocks of the new part of the algorithm are the subgroups of wreath
products, described as F in Section 3. We list a few extra results for a random
selection of them in Table 2. Observe that, as promised, the number of outer classes
usually is small compared with the total number of classes, but some classes are very
small, so a random search for outer classes as proposed in [3] will not necessarily work. We also see that these groups usually have very many classes in total, so even a modified random search along the lines of [12] is therefore not suitable.

A notable exception among the otherwise satisfactory runtimes is \(1/2[S_{11}^2]^2\), for which the new algorithm yields only a minimal improvement. In this group (the wreath product \(S_{11} \wr 2\) has 3 normal subgroups of index 2 namely \(1/2[S_{11}^2]^2\), \(1/2[S_{11}^2]^2\) and \(S_{11}^2\); see [3] for explanation of the names) the block action \(C_2\) is not “bodily”, that is, the group does not split over its normal subgroup \(S_{11}^2\). Consequently, the centralizer \(C_F(r)\) is relatively small and thus has many small orbits which are hard to fuse. However the loss in comparison to the standard algorithm is only due to the small size, as the similarly built \(1/2[S_{12}^2]^2\) shows.

In general, we deduce from comparison with the standard algorithm that the group must be large enough to make the new algorithm worth using; otherwise the setup of the series and the subdirect products takes too long.

On the other hand, the algorithm is capable of dealing with groups which the standard algorithm is incapable of, and may become useful in these cases.

12. Closing remarks

The algorithm presented in this note does not rely on \(G\) being a permutation group. All parts of the calculations which depend on the representation are kept in routines called (mainly: composition series, Double cosets and Homomorphisms). As soon as such routines are provided for other types of group representations, the algorithm becomes available there.

The parametrization of classes used here may also be used to run through the classes of a large group without storing all class representatives at once.

The author would like to thank Steve Linton for helpful comments on a first draft.

13. List of symbols

\[\begin{array}{ll}
A_i & \text{constituent groups of } M \\
C & C_G(N) \\
C_i & \text{pre-image in } M \text{ of centralizer in } M \pi_i \\
D & \text{direct product } \times \prod_{i=1}^n A_i \\
d & \text{element of } D \\
E & \text{image } N \varphi \leq F \\
F & G \varphi \cong G/C \\
G & \text{Group for which we want to compute the conjugacy classes} \\
L_i & \bigcap_{j \neq i} \ker \pi_i < M, \text{ subgroup acting on one component only} \\
L & \langle L_i \rangle \\
M & \text{subdirect product, for example } \ker \psi \\
m & \text{element of } M \\
N & \text{elementary (minimal) normal subgroup of } G \\
\varphi & : G \to F, \text{ action of } G \text{ on } N \\
\pi_i & : M \to A_i, \text{ component projection} \\
\omega_i & = (\pi_1, \ldots, \pi_i), \text{ projection on first } i \text{ components} \\
\psi & : F \to P, \text{ action on the components of } N. \\
\psi^{-1} & \text{full preimage under } \psi \\
N & \text{“dimension” of } N \\
P & F/M \cong F/\psi \\
r & \text{representative in } F \text{ of } P\text{-class} \\
S & \text{component stabilizer} \\
(\text{Stab}_{F/\psi}(1))^{-1} & \\
T & \text{simple group, } N \cong T^n \\
U & \text{subgroup that centralizes } r \\
Z & \text{full centralizer preimage} \\
(C_F(r\psi))^{-1} & .
\end{array}\]
REFERENCES


SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES, UNIVERSITY OF ST. ANDREWS, THE NORTH HAUGH, UK-ST ANDREWS, FIFE KY16 9SS, SCOTLAND

E-mail address: ahulpke@dcs.st-and.ac.uk