# Computing Automorphism Groups and Isomorphism Testing in Finite Groups

by

**David J A Howden**

**Thesis**

Submitted to The University of Warwick

for the degree of

**Doctor of Philosophy**

**Mathematics Institute**

April, 2012

# Table of contents

# List of Algorithms

# Acknowledgments

First and foremost I would like to thank my supervisor, Derek Holt, for a great deal of help and encouragement throughout my time at Warwick. He has been incredibly generous with his time and a constant source of enthusiasm and sound advice.

I would also like to thank John Cannon and the MAGMA group in Sydney for inviting me to work with them for 5 weeks in April-May 2011. In particular, I would like to mention Dan Roozemond who assisted me in improving the automorphism group algorithm for $p$-groups, and Bill Unger who helped me make changes to the internal MAGMA system which allowed for better integration with implementations of various algorithms in this thesis. I thank them all for their hospitality which made for a productive and greatly enjoyable trip.

I am also grateful to numerous fellow students, in particular Janosch Ortmann, Mikołaj Sierżega, Paul Cadman, Ben Sharp, Barinder Banwait and Rupert Swarbrick for many inspiring and interesting conversations over the years. Special thanks to Rupert for proof reading sections of this thesis.

Thank you to my family for their constant love and support. And to all those who have made it a fun few years, especially Damon, Matt, David and Martha, and most of all Anna.

This work was supported by EPSRC.

# Declaration

The author declares that the material contained in this thesis is entirely his own work. As is standard practice, the subject matter developed builds on existing theory, and clear citations and references are provided where necessary.

Although material contained herein may be submitted for publication at a later date, the author has not published any work which forms part of this thesis.

No part of this thesis has been submitted, for the purposes of a degree or otherwise, to any other university or educational institution.

## Abstract

We outline a new method for computing automorphism groups and performing isomorphism testing for soluble groups. We derive procedures for computing polycyclic presentations for soluble automorphism groups, allowing for much more efficient calculations. Finally, we demonstrate how these methods can be extended to tackle some non-soluble groups. Performance statistics are included for an implementation of these algorithms in the MAGMA [BCP97] language.

# Notation and displayed procedures

For a group $G$ the action of an automorphism $\alpha \in \mathrm{Aut}(G)$ on some element $g \in G$ will be denoted $g^{\alpha}$, though we may revert to writing $\alpha(g)$ where expressions become complicated. Note that $g^{\alpha\beta} = \beta(\alpha(g))$. An automorphism $\alpha \in \mathrm{Aut}(G)$ is said to fix some subgroup $H \leq G$ if $H^{\alpha} = H$ and we say that $\alpha$ centralises $H$ if $h^{\alpha} = h$ for all $h \in H$. If $H^{\alpha} = H$ for every $\alpha \in \mathrm{Aut}(G)$ then $H$ is said to be a *characteristic* subgroup of $G$, which will be denoted $H \operatorname{char} G$. Given a group $G$, a subgroup $H \leq G$, and group of automorphisms $A$ of $G$, we define $A_H = \{\alpha \in A : H^{\alpha} = H\}$.

We define $\overline{g} : G \to G$ by $\overline{g} : x \mapsto g^{-1}xg$, the inner automorphism of $G$ induced by $g \in G$, and $\overline{H} \leq \mathrm{Aut}(G)$ for some subgroup $H \leq G$ is defined as $\overline{H} = \{\overline{h} : h \in H\}$. For two elements $g, h \in G$, $g^h = h^{-1}gh$, the conjugation of $g$ by $h$. The identity automorphism of $G$ will be denoted by $\mathrm{Id}_G : G \to G$ where $\mathrm{Id}_G : g \mapsto g$ for all $g \in G$. Finally, we will use $1_G$ to denote the identity element of $G$ and $1 = \{1_G\} \leq G$, the trivial subgroup. Finally, we define natural numbers as $\mathbb{N} = \{1, 2, 3, \dots\}$.

In pseudocode, algorithms or function names will be written in small caps; for example AutomorphismGroupPGroup. Constants are written as true, false, null. If a variable is passed to a function and is prefixed with a $\sim$, then this means that any operations made to the argument in the scope of the function are actually performed on the variable which is passed. In other words, the function can change the value of the argument.

Aside from group theoretic functions which will be detailed later, we will also refer to the following functions and algorithmic constructs as standard:

- [ ] an empty list
- $[1, 2, 3]$ an ordered list
- [1..5] shorthand for the ordered list [1,2,3,4,5]
- [10..1 by -3] shorthand for the ordered list [10,7,4,1]

The function Append($\sim l$, $x$) will append the element $x$ to the list $l$, and for two lists, $L_1$ and $L_2$, $L_1$ cat $L_2$ is their concatenation.

# Chapter 1

# Introduction

Computing automorphism groups for general finite groups is a hard problem. Over the last 40 years or so there has been a lot of effort to develop algorithms which will construct $\mathrm{Aut}(G)$ for specific types of group $G$.

The earliest successful work on this problem was carried out by Felsch and Neubüser in 1970 [FN67]. Their approach was based on constructing a lattice of subgroups and then choosing a particular generating set for the group as well as a list of maps defined on this generating set. A brute-force search of this list produced all the automorphisms of the group. Later, in 1976, another approach was developed by Robertz [Rob76], which constructed the automorphism group as a permutation representation acting on a union of conjugacy classes of the group. The performance of this algorithm relied heavily on the structure of the conjugacy classes of the group, which meant that its effectiveness diminished and became extremely unpredictable with increasing order.

With the more recent development of efficient algorithms for computing detailed structural information of groups, a new generation of automorphism group algorithms has evolved. A general approach to calculations with finite groups which has proved particularly useful for finite soluble groups $G$ defined by power-conjugate presentations, is to lift computations through a normal series with elementary abelian layers.

For finite soluble groups $G$, Smith introduced an algorithm in [Smi94], since improved by Slattery, which computes the automorphism group of $G$. The strategy is to first construct a characteristic series for $G$ with elementary abelian factors. The automorphism group of the top quotient is then calculated, and lifted through successive quotients of the series with $G$. This approach has been adapted by Holt and Cannon in [CH03] for use on general finite groups. They begin by computing the largest normal soluble subgroup $L$ of the finite group $G$, and then construct a characteristic series for $L$ with elementary abelian factor groups. The automorphism group $\mathrm{Aut}(G/L)$ is determined using a combination of pre-computed group databases, and other elementary methods; then, as in the soluble algorithm, a lifting process runs through quotients of $G$ with successive terms of the computed characteristic series of $L$.

For finite $p$-groups, there is a separate algorithm which was first introduced by O'Brien in [O'B92] and has since been further improved by O'Brien, Leedham-Green and Eick in [ELGO02]. Again, it follows a similar lifting process, but using quotients of the lower central exponent-$p$ series of the given $p$-group. Having been tailored to the $p$-group structure, it considerably outperforms the more general lifting algorithms in most cases as the lifting process can be reduced to a simpler problem.

There are various advantages and disadvantages to using any of these three approaches. Each generally performs better on its designated group type, and is therefore preferred, though it is not very hard to find examples where the more general routines finish in a shorter time. This makes absolute performance statements unreliable, so we try to avoid making generalisations of this type. When considering the performance of each algorithm, we also need to take note of the representation type which is required of the input group. The more general algorithm for finite groups relies on the input group being given as a permutation or matrix group, and this places immediate limitations on its usefulness for very large examples where a low-degree permutation representation or small matrix group representation can not be easily constructed. The soluble and $p$-group algorithms require that the input group be represented by a power-conjugate presentation, which allow more easily for the construction and computation with large groups. It is relatively routine to convert permutation or matrix groups to power-conjugate presentations if they are known to be soluble.

All three algorithms have been implemented in MAGMA [BCP97]; the soluble group procedure is written in 'C', and the general and $p$-group algorithms in the MAGMA language. The soluble group and $p$-group algorithms are also available as GAP [GAP08] packages (`autag` in GAP3, now integrated into GAP4, and `AutPGrp` [EO09] respectively). Given a finite group $G$ as input, expressed in an appropriate representation, they each return a group of automorphisms $A$. In both systems, $A$ is represented by a list of automorphism maps which generate the automorphism group. Elements of $A$ can be composed, inverted and their orders can be evaluated. The order of $A$, which is determined during its construction, is stored as an attribute of $A$.

To carry out structural calculations on a group of automorphisms $A$, it is necessary to find an appropriate representation. In general, this means attempting to construct a permutation representation of $A$, which is impractical when $G$ is large. If $A$ is soluble, it would be ideal to directly construct a power-conjugate presentation, without having to first find a permutation representation.

The recent development of databases of finite groups allows for more effective benchmarking of existing algorithms, and in particular leads to readily available collections of large groups for which the algorithms perform poorly. We will concentrate on the following databases:

- **Small Groups** of order up to 2000, described in [BEO02]. In general, both the soluble and finite group algorithms work well on examples taken from the small groups database of orders up to 2000. However, there are a large number of notable "hard cases" where computing the automorphism group of soluble groups can take up to 10000 times longer than other groups of the same order, particularly when using the soluble algorithm.

- **Transitive Groups** up to degree 32 (degrees up to 11 are given in [BM83], degree 12 in [Roy87], degrees 14 and 15 in [But93], all degrees up to 30 in [Hul05], and degree 32 in [CH08]). This database gives rise to a great many examples, particularly groups of degree 24, for which the current algorithms can take up to 20 hours to complete.

- **Primitive Groups**, recently updated to include primitive groups up to degree 4095 [CQRD11]. This database provides several soluble examples where the finite group automorphism algorithm struggles (particularly those groups with degree above 2000).

There are other notable hard examples which can be easily constructed. For instance, direct products such as $\times_{i=1}^{n} \mathrm{Sym}(3)$ and $\times_{i=1}^{n} D_6$ can take hours to finish when $n > 5$ (where $D_m$ is the dihedral group of order $2m$ for some $m \in \mathbb{N}$).

In this thesis we describe new methods for computing automorphism groups and performing isomorphism testing in finite groups, which can be applied to both soluble, and some non-soluble finite groups. For finite soluble groups, we also provide a new approach to finding power-conjugate representations of soluble automorphism groups, which avoids creating a permutation representation of the whole group as an intermediary step.

The motivation behind much of this work stems from the many examples of soluble groups in the catalogue of small groups of orders up to 2000, for which the standard soluble group automorphism group algorithm performs very badly. In general, the approach given here relies on the computation of an automorphism group of some large $p$-group $P \in \mathrm{Syl}_p(G)$ of the soluble group $G$. When the computation of $\mathrm{Aut}(P)$ is fast relative to the normal time required to construct $\mathrm{Aut}(G)$, our algorithm performs better than the soluble and finite group algorithms. In practice, we have found that almost all examples which are hard cases for the existing soluble group algorithm are computed very quickly using our method. Indeed, our approach works consistently on almost all small groups in the database, completing in under 1 second for almost every entry.

We have implemented both the automorphism group computation and isomorphism testing algorithms for soluble groups in the MAGMA language, and this is available on request.

In Chapters 2 and 3, we give background material and preliminary theory and algorithms, which will be necessary for later results. The theory behind our approach is outlined in Chapter 4, and Chapters 5 and 6 adapt this to construct algorithms for computing automorphism groups and performing isomorphism testing for finite soluble groups, respectively. We devote Chapter 7 to exploring different ways of extending our general approach to some finite non-soluble groups, making use of algorithms for soluble groups developed earlier, and many of the ideas behind them. Finally, in Chapter 8 we give some benchmarks of our implementation in MAGMA, and compare it to both the general algorithm for finite groups and the current soluble group algorithm.

# Chapter 2

# Background material

In this chapter we give a summary of the background material required for the results that follow in later chapters.

**Definition 2.1.** A group $G$ is said to be *soluble* (or *solvable*) if it has a finite subnormal series $G = G_0 \geq G_1 \geq \ldots \geq G_n = 1$ such that each factor group $G_i/G_{i+1}$ for $1 \leq i < n$ is abelian.

**Definition 2.2.** Let $G$ be a finite group. Then the *soluble radical* of $G$, denoted $O_\infty(G)$, is defined to be the largest normal soluble subgroup of $G$.

It is clear that such a subgroup exists, since if $A, B \leq G$ are any two soluble normal subgroups of a group $G$, then $AB$ is soluble and $AB \trianglelefteq G$. From the definition it is also clear that $O_\infty(G)$ is a characteristic subgroup of $G$.

The next few results give various structural details which will be used later. The following two results are used to detect direct products.

**Proposition 2.3.** *Let $X$ be a group with subgroups $U, V \leq X$ such that $X = UV$, $U \cap V = 1$ and $[U, V] = 1$. Then $X \cong U \times V$.*

*Proof.* Define a mapping $\phi : U \times V \to X$ by $\phi : (u, v) \mapsto uv$ for $u \in U$, $v \in V$. Then for $u_1, u_2 \in U$ and $v_1, v_2 \in V$ we have

$$\begin{aligned} ((u_1, v_1)(u_2, v_2))^{\phi} &= (u_1u_2, v_2v_2)^{\phi} \\ &= u_1u_2v_1v_2 \\ &= u_1v_1u_2v_2 \\ &= (u_1, v_1)^{\phi}(u_2, v_2)^{\phi} \end{aligned}$$

since $u_2 \in C_U(V)$ and hence $\phi$ is a homomorphism. For $u \in U, v \in V$ with $(u, v)^{\phi} = 1_X$ gives $uv = 1_X$ and hence $u, v \in U \cap V = 1$, and so $\phi$ is injective. For any $x \in X$, we have $x = uv$ for some $u \in U$, $v \in V$. Then $(u, v)^{\phi} = x$ and $\phi$ is surjective. □

**Proposition 2.4.** *Let $X$ be a group with subgroups $U, V \leq X$ such that $V \lhd X$, $X = UV$ and $U \cap V = 1$. Then $[U, V] = 1$ if and only if $U \lhd X$.*

*Proof.* Suppose that $C_U(V) = U$ and hence $C_V(U) = V$. Then $V \leq N_X(U)$ and $X = UV = N_X(U)$, so $U \lhd X$. Now suppose that $U \lhd X$ and consider the element $u^{-1}v^{-1}uv \in X$ for $u \in U$ and $v \in V$. Clearly, $(u^{-1}v^{-1}u)v \in V$ and $u^{-1}(v^{-1}uv) \in U$, hence $u^{-1}v^{-1}uv \in U \cap V$. So $u^{-1}v^{-1}u = v^{-1}$, and $C_U(V) = U$. □

**Lemma 2.5.** *Let $G$ be a cyclic group. Then* $\mathrm{Aut}(G)$ *is abelian.*

*Proof.* Let $|G| = n$. Every automorphism of $G$ is of the form $\alpha_k : G \to G$ where $\alpha_k : g \mapsto g^k$ for some $k$ with $(k, n) = 1$. Thus the product of any two automorphisms is $\alpha_k\alpha_l = \alpha_l\alpha_k : g \mapsto g^{k+l}$. □

**Proposition 2.6.** *Let $G$ be a group and $H \leq G$. Then $N_G(H)/C_G(H) \cong I$ for some subgroup $I \leq \mathrm{Aut}(H)$.*

*Proof.* Define a homomorphism $\phi : G \to \mathrm{Aut}(G)$ by $\phi : g \mapsto \overline{g}$. For each $g \in N_G(H)$ note that $g^{\varphi}|_H$ is an automorphism of $H$, and so we can define $\varphi : N_G(H) \to \mathrm{Aut}(H)$ as $\varphi = (\phi|_{N_G(H)})|_H$. Given $g \in N_G(H)$, $g \in \mathrm{Ker}\,\varphi$ if and only if $\overline{g}|_H = \mathrm{Id}_H$ and hence $g^{-1}hg = h$ for all $h \in H$. Thus $g \in \operatorname{Ker}\varphi$ if and only if $g \in C_G(H)$. By the second isomorphism theorem

$$\frac{N_G(H)}{C_G(H)} \cong \operatorname{Im}\varphi,$$

and, as noted earlier, $\operatorname{Im}\varphi \leq \operatorname{Aut}(H)$. Hence $\operatorname{Im}\varphi \cong I$ for some subgroup $I \leq \operatorname{Aut}(H)$. □

## 2.1 Sylow subgroups and Hall subgroups

In this section we outline the well-known results of Sylow and Hall regarding the existence of prime power or composite prime order subgroups in finite and finite soluble groups respectively. For a more detailed exposition of this theory, consult [Rot95].

**Definition 2.7.** Let $G$ be a finite group with $|G| = p^\alpha t$ where $\operatorname{hcf}(p^\alpha, t) = 1$. Then a subgroup $P \leq G$ is a Sylow $p$-subgroup of $G$ if $|P| = p^\alpha$. We denote the set of all Sylow $p$-subgroups of $G$ by $\operatorname{Syl}_p(G)$.

**Theorem 2.8** (Sylow, 1872)**.** *Let $G$ be a finite group with $|G| = p^\alpha t$ for some prime $p$ and $t$ with $\operatorname{hcf}(t, p) = 1$. Then*

1. *For each $0 < \gamma \leq \alpha$ there exists a subgroup $P_\gamma$ such that $|P_\gamma| = p^\gamma$ and any such subgroup is contained in a Sylow $p$-subgroup;*

2. *$|\operatorname{Syl}_p(G)| \equiv 1 \pmod{p}$, and*

3. *Any two Sylow $p$-subgroups of $G$ are conjugate in $G$.*

**Definition 2.9.** Let $\pi$ be a function defined as

$$\pi(n) = \{p \in \mathbb{N} : p \text{ is prime}, p|n\}.$$

For a finite group $G$, we set $\pi(G) = \pi(|G|)$. Note that in this context we define $p'$ (or the $p$ complement) to be the set $p' = \{q \in \pi(G) : q \neq p\}$, where $p \in \pi(G)$ for a finite group $G$. For a given set of primes $\pi$, a group $G$ is a $\pi$-group if $\pi(G) \subseteq \pi$. Similarly, $H \leq G$ is called a $\pi$-subgroup of a given finite group $G$ if $\pi(H) \subseteq \pi$ for some $\pi \subseteq \pi(G)$.

We now state the result of Hall detailing the existence of $\pi$-subgroups for soluble groups.

**Theorem 2.10** (Hall, 1928)**.** *If $G$ is a soluble group of order $ab$, where* $\mathrm{hcf}(a, b) = 1$*, then $G$ contains a subgroup of order $a$. Moreover, any two subgroups of order $a$ are conjugate.*

**Definition 2.11.** Let $G$ be a finite group, then a *Hall subgroup $H$* of $G$ is a subgroup whose order and index are relatively prime; that is $\mathrm{hcf}(|H|, |G : H|) = 1$. We use $\mathrm{Hall}_n(G)$ to denote the set of Hall subgroups of order $n$ in $G$.

Observe that if $G$ is a finite group and $n$ is a prime dividing $|G|$, then $\mathrm{Hall}_n(G) = \mathrm{Syl}_n(G)$. For a given subset $\pi \subseteq \pi(G)$ define $\mathrm{Hall}_\pi(G)$ to be the set of Hall $\pi$-subgroups of $G$. Note that, by definition, each $H \in \mathrm{Hall}_\pi(G)$ has order $|H| = p_1^{n_1} p_2^{n_2} ... p_r^{n_r}$ such that $p_i^{n_i+1} \nmid |G|$ for each $p_i \in \pi$. In addition, if $G$ is soluble then $\mathrm{Hall}_\pi(G)$ is non-empty by Theorem 2.10 and all Hall $\pi$-subgroups are conjugate in $G$.

The conjugation property of Sylow gives rise to the well know "Frattini Argument" given here with proof. We also include a corollary stating the equivalent result reformulated for Hall subgroups of finite soluble groups.

**Theorem 2.12** (Frattini Argument)**.** *Let $G$ be a group, $K \lhd G$ and $P \in \mathrm{Syl}_p(K)$. Then $G = N_G(P)K$.*

*Proof.* For any $g \in G$, $g^{-1}Pg \leq g^{-1}Kg = K$ since $K \lhd G$, and so there exists $k \in K$ such that $g^{-1}Pg = k^{-1}Pk$ since all Sylow $p$-subgroups of $K$ are conjugate in $K$. Hence $gk^{-1} \in N_G(P)$, and $g = (gk^{-1})k$. □

**Corollary 2.13** (Extended Frattini). *Let $G$ be a group and let $K \lhd G$ be a soluble subgroup. If $H \in \mathrm{Hall}_\pi(K)$ for some subset $\pi \subseteq \pi(G)$ then $G = N_G(H)K$.*

### 2.1.1 Useful $\pi$-subgroups

Observe that for a finite group $G$ and a given prime $p \in \pi(G)$, we can construct a normal $p$-subgroup of $G$ by taking the intersection of all the Sylow $p$-subgroups. In fact this formulation gives rise to the $p$-core of $G$, denoted $O_p$, which is the largest normal $p$-subgroup of $G$. This principle can also be applied to Hall $\pi$-subgroups in soluble groups, and generalised further for $\pi$-subgroups of finite groups.

**Definition 2.14.** Let $G$ be a finite group and $\pi \subseteq \pi(G)$. Then we define $O_\pi(G)$ to be the subgroup of $G$ generated by all normal $\pi$-subgroups of $G$.

**Lemma 2.15.** *Let $G$ be a finite group, and $\pi \subseteq \pi(G)$. Then $O_\pi(G)$ is the largest normal $\pi$-subgroup of $G$; and $O_\pi(G)$ is characteristic in $G$.*

*Proof.* Firstly we note that $O_\pi(G)$ is a $\pi$-group as it is a product of $\pi$-subgroups of $G$. Now suppose that $H \lhd G$ and $H$ is a $\pi$-subgroup of $G$. Then $|H| \leq |HO_\pi(G)| = |O_\pi(G)|$ and hence $|H| \leq |O_\pi(G)|$. Therefore $O_\pi(G)$ is the largest normal $\pi$-subgroup. For any $\alpha \in \mathrm{Aut}(G)$, $O_\pi(G)^\alpha$ is a normal $\pi$-subgroup of $G$ thus $O_\pi(G)^\alpha \leq O_\pi(G)$, and further $|O_\pi(G)^\alpha| = |O_\pi(G)|$. Therefore $O_\pi(G)^\alpha = O_\pi(G)$, and $O_\pi$ is characteristic in $G$. □

**Lemma 2.16.** *Let $G$ be a finite group and $H \in \mathrm{Hall}_\pi(G)$ for some subset $\pi \subseteq \pi(G)$. Then $H \lhd G$ if and only if $|\,\mathrm{Hall}_\pi(G)| = 1$. Furthermore, if $H \lhd G$ then $H \operatorname{char} G$.*

*Proof.* Suppose that $H \lhd G$. Then $H = O_\pi(G)$ and hence $H = K$ for any $K \in \mathrm{Hall}_\pi(G)$, since otherwise $|HK| > |H|$. Conversely, suppose that $|\,\mathrm{Hall}_\pi(G)| = 1$. Then for $H \in \mathrm{Hall}_\pi(G)$ we have $|H^\alpha| = |H|$ for any $\alpha \in \mathrm{Aut}(G)$, and thus $H$ is characteristic, and hence normal. Now note that for any $\alpha \in \mathrm{Aut}(G)$, $|H^\alpha| = |H|$, and so $H^\alpha \in \mathrm{Hall}_\pi(G)$. Hence, if $H \lhd G$, we have that $|\,\mathrm{Hall}_\pi(G)| = 1$, and $H^\alpha = H$. □

We note that given a finite group $G$ and a Sylow $p$-subgroup $P \in \mathrm{Syl}_p(G)$ for some $p \in \pi(G)$, then $|\mathrm{Syl}_p(G)| = 1$ if and only if $P \,\mathrm{char}\, G$ (an important special case of Lemma 2.16).

### 2.1.2 Automorphisms acting on $\mathrm{Syl}_p(G)$ and $\mathrm{Hall}_\pi(G)$

In this section we give two results which describe the action of automorphisms $\alpha \in \mathrm{Aut}(G)$ on $\mathrm{Syl}_p(G)$ for finite groups $G$ with $p \in \pi(G)$, and analogous actions on $\mathrm{Hall}_\pi(G)$ for finite soluble groups $G$ with $\pi \subseteq \pi(G)$.

**Lemma 2.17.** *Let $G$ be a finite group, $P \in \mathrm{Syl}_p(G)$ for some $p \in \pi(G)$ and $\alpha \in \mathrm{Aut}(G)$. Then $\alpha$ permutes $\mathrm{Syl}_p(G)$ and $P^\alpha = g^{-1}Pg$ for some $g \in G$.*

*Proof.* Clearly $|P^\alpha| = |P|$, and so by definition $P^\alpha \in \mathrm{Syl}_p(G)$. Furthermore, all Sylow $p$-subgroups are conjugate in $G$ hence there exists some $g \in G$ such that $P^\alpha = g^{-1}Pg$. □

**Lemma 2.18.** *Let $G$ be a finite group, $H \in \mathrm{Hall}_\pi(G)$ for some subset $\pi \subseteq \pi(G)$, and $\alpha \in \mathrm{Aut}(G)$. Then $\alpha$ permutes $\mathrm{Hall}_\pi(G)$ and if $G$ is soluble then $H^\alpha = g^{-1}Hg$ for some $g \in G$.*

*Proof.* Clearly $|H^\alpha| = |H|$, and so by definition $H^\alpha \in \mathrm{Hall}_\pi(G)$. Furthermore if $G$ is soluble, then all Hall $\pi$-subgroups are conjugate in $G$ hence there exists some $g \in G$ such that $H^\alpha = g^{-1}Hg$. □

## 2.2 Polycyclic groups

A group $G$ is said to be *polycyclic* if it has a descending chain of subgroups

$$G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$$

in which each $G_{i+1}$ is normal in $G_i$ and each quotient subgroup $G_i/G_{i+1}$ is cyclic. Any finite soluble group is a polycyclic group, and any polycyclic group is soluble. Polycyclic presentations are a form of finite presentation used to express polycyclic groups which allow for practical, and often efficient, computations with their underlying groups. For a broader background on the theory of polycyclic groups, we refer the reader to Segal's book [Seg83]. Here we will define polycyclic sequences and polycyclic presentations, and explore some of their elementary properties and uses (we are basing this brief introduction on [HEO05, 8.1]).

### 2.2.1 Polycyclic sequences

Let $G$ be a polycyclic group with a polycyclic series $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$. As $G_i/G_{i+1}$ is cyclic, there exist elements $x_i \in G$ with $\langle x_i G_{i+1} \rangle = G_i/G_{i+1}$ for every $i$.

**Definition 2.19.** A sequence of elements $X = [x_1, \ldots, x_n]$ such that $\langle x_i G_{i+1} \rangle = G_i/G_{i+1}$ for $1 \leq i \leq n$ is a *polycyclic sequence* for $G$.

**Definition 2.20.** Let $X$ be a polycyclic sequence for $G$. The sequence $R(X) := (r_1, \ldots, r_n)$ defined by $r_i := |G_i : G_{i+1}| \in \mathbb{N} \cup \infty$ is called the *sequence of relative orders* for $X$. The set $\{i \in \{1..n\} \,|\, r_i \text{ finite}\}$ is denoted by $I(X)$.

The sequence $R(X)$ and the set $I(X)$ depend on the underlying polycyclic series $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$ and $X$ only. Also note that $G$ is finite if and only if $I(X) = \{1, \ldots, n\}$, and when this is the case $|G| = r_1 \ldots r_n$. We now derive a structure for defining a normal form to elements of $G$.

**Lemma 2.21.** *Let $X = [x_1, \ldots, x_n]$ be a polycyclic sequence for $G$ with the relative orders $R(X) = (r_1, \ldots, r_n)$. Then for every $g \in G$ there exists a unique sequence $(e_1, \ldots, e_n)$, with $e_i \in \mathbb{Z}$ for $1 \leq i \leq n$ and $0 \leq e_i < r_i$ if $i \in I(X)$, such that $g = x_1^{e_1} \cdots x_n^{e_n}$.*

**Definition 2.22.** The expression $g = x_1^{e_1} \cdots x_n^{e_n}$ of Lemma 2.21 is called the *normal form* of $G$ with respect to $X$. The sequence $(e_1, \ldots, e_n)$ is the *exponent vector* of $g$ with respect to $X$. We shall write $\exp_X(g) = (e_1, \ldots, e_n)$.

We remark that it is possible to refine any polycyclic sequence to one in which each $r_i$ of $R(X) = (r_1, \ldots, r_n)$ is prime.

### 2.2.2 Polycyclic presentations

**Definition 2.23.** A presentation $\langle\, x_1, \ldots, x_n \,|\, R \,\rangle$ is called a *polycyclic presentation* if there exists a sequence $S = (s_1, \ldots, s_n)$ with $s_i \in \mathbb{N} \cup \{\infty\}$ and integers $a_{i,k}$, $b_{i,j,k}$, $c_{i,j,k}$ such that $R$ consists of the following relations:

$$\begin{aligned} x_i^{s_i} &= R_{i,i} \text{ with } R_{i,i} := x_{i+1}^{a_{i,i+1}} \ldots x_n^{a_{i,n}} \text{ for } 1 \leq i \leq n \text{ with } s_i < \infty, \\ x_j^{-1} x_i x_j &= R_{i,j} \text{ with } R_{i,j} := x_{j+1}^{b_{i,j,j+1}} \ldots x_n^{b_{i,j,n}} \text{ for } 1 \leq j < i \leq n, \\ x_j x_i x_j^{-1} &= R_{j,i} \text{ with } R_{j,i} := x_{j+1}^{c_{i,j,j+1}} \ldots x_n^{c_{i,j,n}} \text{ for } 1 \leq j < i \leq n \text{ where } j \notin I(X). \end{aligned}$$

These relations are called *polycyclic relations.* The first type are the *power relations* and the second and third types are the *conjugate relations.* $S$ is called the sequence of *power exponents* of the presentation.

Conjugation relations of the form $x_j^{-1} x_i x_j = x_i$ or $x_j x_i x_j^{-1} = x_i$ are called *trivial polycyclic relations.* They are often omitted from a polycyclic presentation to simplify the notation, and this means that polycyclic presentations have to be distinguished from arbitrary finite presentations. For this purpose we denote them with

$$\mathrm{Pc}\langle x_1, \ldots, x_n | \, R \,\rangle.$$

A group that is defined as and represented by a polycyclic presentation is known as a *PC-group.*

**Theorem 2.24.** *Every polycyclic sequence is the generating set of a unique poly-*

*cyclic presentation. Thus every polycyclic group can be defined by a polycyclic presentation.*

## 2.3 Computing orbits and stabilisers

Let $G$ be a finite group acting on some finite set $\Omega$. A common process in computational group theory, which will be used in later algorithms, is to determine the orbit of some point $\alpha \in \Omega$ under the action of $G$, denoted $\mathrm{Orbit}_G(\alpha)$. Let us assume that $G = \langle X \rangle$ for some finite sequence of generators $X = [x_1, x_2, \ldots, x_n]$ such that $x_i \in \mathrm{Sym}(\Omega)$ for each $i$. Further, we shall assume that $\alpha^g$, the image of the action of an element $g \in G$ on a point $\alpha \in \Omega$, can be determined, and that it is possible to test if two elements $\alpha, \beta \in \Omega$ are equal. Computing the orbit is then straightforward.

Beginning with $\Delta = \{\alpha\}$ for some $\alpha \in \Omega$, we apply each of the generators $x_i \in G$ to the elements of $\Delta$. If $\alpha^{x_i} \notin \Delta$, then $\alpha^{x_i}$ is added to $\Delta$. We repeat this process until the images of all orbit points are contained in $\Delta$.

Often, we need to retain a list of elements of $G$ which correspond to each orbit element, so for every $\delta \in \Delta$ we have an element $g \in G$ such that $\alpha^g = \delta$. In practice, this proves to be very useful when constructing the stabiliser of $\alpha$ in $G$, denoted $\mathrm{Stab}_G(\alpha)$. For example, suppose that $\delta^x \in \Delta$ for some $\delta \in \Delta$, $x \in G$. Then there exists $g, h \in G$ such that $\alpha^g = \delta$ and $\alpha^h = \delta^x$, and hence $gxh^{-1} \in \mathrm{Stab}_G(\alpha)$. Therefore it is usual for the orbit and stabiliser to be computed together.

We give a formal outline of the full orbit stabiliser procedure in Algorithm 2.1, and define $\Delta^* \subseteq \Omega \times G$, a set which stores orbit elements as pairs: $(\beta, u_\beta)$ where $\alpha^{u_\beta} = \beta$.

As an alternative to keeping the explicit group element $g \in G$ which maps $\alpha^g = \beta$, one can store a list of indices instead, which allow the element to be constructed when needed and greatly reduces the amount of memory needed to complete the computation. This data structure is called a Schreier vector, and is defined as

**Algorithm 2.1:** OrbitStabiliser

```
  Input: $(\alpha, X) : \alpha \in \Omega, X = [x_1, \ldots, x_n], x_i \in \mathrm{Sym}(\Omega)$ with $\langle X \rangle = G$
  Output: $(\Delta^*, Y)$ : The orbit $\Delta^*$ of $\alpha$ under $G$, and $\langle Y \rangle = \mathrm{Stab}_G(\alpha)$
1 $\Delta^* := [(\alpha, 1_G)]$;
2 $Y := [\ ]$;
3 for $(\beta, u_\beta) \in \Delta^*, x_i \in X$ do
4     if $\beta^{x_i} \notin \Delta^*$ then
5         Append($\sim \Delta^*, (\beta^{x_i}, u_\beta x_i)$);
6     else
7         Append($\sim Y, u_\beta x_i (u_{\beta^{x_i}})^{-1}$);
8 return $\Delta^*, Y$;
```

follows.

**Definition 2.25** (Schreier Vector)**.** A *Schreier vector* for $\alpha \in \Omega$ is a sequence $v$, indexed by $\Omega$, which adheres to the following rules:

- $v[\alpha] = -1$.
- For $\gamma \in \mathrm{Orbit}_G(\alpha)$, $v[\gamma] = i$ where $\gamma$ is appended to $\Delta$ as $\beta^{x_i}$, see Line 5 of OrbitStabiliser (Algorithm 2.1).
- $v[\beta] = 0$ for $\beta \notin \mathrm{Orbit}_G(\alpha)$.

It is straightforward to now modify OrbitStabiliser to use a Schreier vector instead of storing each of the group elements with their respective orbit element. The group element required to map $\alpha \mapsto \beta$ for $\beta \in \Delta$, can then be computed, if needed, by following the orbit construction described by the Schreier vector.

### 2.3.1 Polycyclic orbit stabiliser

When computing orbits and stabilisers for the action of a finite polycyclic group $G$ on a set $\Omega$, the algorithm can be altered slightly from the one described above, to take advantage of the extra structure afforded by the presentation.

This is best demonstrated in pseudocode, which can be found in PolycyclicOrbitStabiliser (Algorithm 2.2). This is taken from [HEO05, page 304], and we refer the reader here for a more detailed explanation of how the the polycyclic structure of $G$ is used to construct the orbit and stabiliser.

**Algorithm 2.2:** PolycyclicOrbitStabiliser

```
   Input: $(X, \alpha) : X = [x_1, \dots, x_n]$ is a polycyclic sequence for $G$ with
          $R(X) = (r_1, \dots, r_n)$ the corresponding relative orders
   Output: $(Y, \Delta)$ : An induced polycyclic sequence $Y$ for $\mathrm{Stab}_G(\alpha)$ and $\Delta$
           the orbit of $\alpha$ under $G$

 1 $\Delta := [\alpha]$;
 2 $Y := [\ ]$;
 3 for $i \in [n..1\ by\ -1]$ do
 4     $\beta := \alpha^{x_i}$;
 5     if $\beta \in \Delta$ then
 6         Find $g \in G_2$ with $\beta = \alpha^g$;
 7         $Y := [x_i g^{-1}] \cup Y$;
 8     else
 9         $\Gamma := [\ ]$;
10         for $j \in [1..r_j - 1]$ do $\Gamma := \Gamma$ cat $[\delta^{x_i^j} : \delta \in \Delta]$;
11         $\Delta := \Delta$ cat $\Gamma$;
12 return $\Delta, Y$;
```

It remains to show how to find the element $g$ which maps $\alpha^g = \beta$ (Line 6). Since the orbit $\Delta$ increases by a known amount, i.e. $r_i|\Delta|$, when $\alpha^{x_i} \neq \alpha$ for some $i$, we only need to record those $i$ for which $x_i$ acts non-trivially on $\alpha$. The element $g$ can then be computed from its position in $\Delta$ using modular arithmetic.

## 2.4 Computations in permutation groups

Machinery for handling computations in permutation groups is very well developed. Introduced by Charles C. Sims in [Sim71] and [Sim70], the so-called "Base and Strong Generating Set" construction allows for detailed calculations to be carried out in permutation groups of degrees up to about $10^7$. We will give a very brief overview of the basic setup here, and refer the reader to [CH03] for a more detailed introduction.

Let $G$ be a finite permutation group acting on some finite set $\Omega = \{1, \ldots, n\}$, and suppose that $S$ is a finite set of elements of $\mathrm{Sym}(n)$ which generate $G$. A *base* is some sequence of elements $X = x_1, x_2, \ldots, x_k$ with each $x_i \in \Omega$, such that the only element of $G$ fixing every $x_i$ is the identity. Define $G^{(i)} := \mathrm{Stab}_G(x_1, x_2, \ldots, x_{i-1})$ for $1 \leq i \leq k+1$, then $G = G^{(1)}$, $G^{(k+1)} = 1$ and $G^{(i+1)} \leq G^{(i)}$ for each $i$. Thus we have a chain of subgroups

$$G = G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(k)} \geq G^{(k+1)} = 1.$$

The set $S$ is called a *strong generating set* relative to a base $X$ if it includes generators for each stabiliser $G^{(i)}$, that is $G^{(i)} = \langle S^{(i)} \rangle$ where $S^{(i)} = S \cap G^{(i)}$. It is common to refer to the pair $(X, S)$ as a *BSGS* for $G$ if $X$ is a base and $S$ is a strong generating set relative to $X$.

Assume that we have some base $X$ and a strong generating set $S$ as outlined above. Define $\Delta^i := \mathrm{Orbit}_{G^{(i)}}(x_i)$ and compute a right transversal of $G^{(i+1)}$ in $G^{(i)}$. By the Orbit Stabiliser Theorem, we have $U^{(i)} = \{u_x^{(i)} : x \in \Delta^i\}$ where $u_x^{(i)}$ maps $x_i$ to $x$. Now we have a convenient normal form for elements of $G$ since every $g \in G$ can be expressed as a unique product $g = u_k u_{k-1} \ldots u_2 u_1$ where each $u_i \in U^{(i)}$. Immediately this yields useful information: membership testing is possible via the normal form, and the order of the group is easily determined from the transversals since

$$|G| = |U^{(1)}||U^{(2)}| \dots |U^{(k-1)}||U^{(k)}|.$$

The Schreier-Sims algorithm constructs a base and strong generating set for a given permutation group, and we refer the reader to [CH03] for a full discussion of the procedure. There are some computational issues to be considered in this construction. When computing the $U^{(i)}$, it may be advantageous to avoid storing the transversal representatives explicitly, instead using a more compact form like Schreier vectors (recall Definition 2.25). In practice this will slow down the construction of the base and strong generating set, but crucially it greatly lowers the space requirement for the algorithm to complete successfully. For instance, it is generally impractical to explicitly store the transversals when the underlying permutation group has degree more than 1000.

## 2.5 Computations in matrix groups

There are several different models which are used for computing with matrix groups over finite fields. The most established approach is to construct a BSGS using the action of the given matrix group on its underlying vector space. Let $G \leq \mathrm{GL}(d, p)$ and suppose that $V$ is a $d$-dimensional vector space over the field $\mathbf{F}_p$. The Schreier-Sims algorithm can then be applied to $G$ and constructs a BSGS for the action of $G$ on the vectors and subspaces of $V$. In practice this can be very unwieldy as very large orbits are typical in many standard matrix groups, making routine calculations impractical in general.

An alternative approach for computing with large matrix groups, is to use the Composition Tree (C-T), or "Large Matrix Group" (LMG), method which is summarised in [HLGO11]. This strategy spawned from the *matrix group recognition project*, which uses understanding of different subgroups of $G \leq \mathrm{GL}(n, p)$ to exploit their structure for calculations. The process begins by computing a composition tree of the matrix group and writing of elements of $G$ as words from some specified set $Y$ with $\langle Y \rangle = G$.

## 2.6 Computing automorphism groups

Here we give a brief summary of the current methods used to compute automorphism groups of finite groups.

The soluble group algorithm, developed by Smith [Smi94] and further improved by Slattery, begins by constructing a series of characteristic subgroups

$$G = N_1 > N_2 > \cdots > N_{r-1} > N_r = 1$$

such that $N_i/N_{i+1}$ is elementary abelian for each $i$. The automorphism group $\mathrm{Aut}(G/N_2) \cong \mathrm{GL}(d, p)$ for some $d \in \mathbb{N}$ and prime $p \in \mathbb{N}$ is computed, and then lifted through each of the elementary abelian layers $G/N_3, \ldots, G/N_{r-1}$, finally giving $\mathrm{Aut}(G/N_r) = \mathrm{Aut}(G)$.

A more general approach, outlined in [CH03], constructs the automorphism group of a finite group $G$ by constructing a similar characteristic series to the soluble case,

$$G \geq L = N_1 > N_2 > \cdots > N_{r-1} > N_r = 1$$

but here $L = O_\infty(G)$, the soluble radical of $G$ (recall Definition 2.2). The automorphism group of $\mathrm{Aut}(G/L)$ is then computed using databases of precomputed automorphism groups or elementary methods. Once $\mathrm{Aut}(G/L)$ has been found, this is then lifted through each elementary abelian layer, just as in the soluble case, successively computing $\mathrm{Aut}(G/N_i)$ for $i = 2, \ldots r$, until $\mathrm{Aut}(G/N_r) = \mathrm{Aut}(G)$ is constructed.

**Definition 2.26.** The *lower central exponent-p series* for a finite $p$-group $P$, is a descending sequence of subgroups

$$P = \gamma_1(P) \geq \gamma_2(P) \geq \cdots \geq \gamma_{c-1}(P) \geq \gamma_c(P) = 1,$$

defined by $\gamma_1(P) = P$ and $\gamma_{i+1}(P) = [\gamma_i(P), P]\gamma_i(P)^p$ for $i \geq 1$. If $\gamma_c(P) = 1$, and $c$ is the smallest such integer, then $P$ is said to be of *exponent-$p$ class $c$*, or just *class $c$*.

In the construction of $\mathrm{Aut}(P)$ for a finite $p$-group $P$, the algorithm described in [O'B92] and [EO09], constructs $\mathrm{Aut}(P/\gamma_2(P)) \cong \mathrm{GL}(d, p)$ where $|P/\gamma_2(P)| = p^d$, and then proceeds by induction down the lower exponent-$p$ central series; that is, successively computes $\mathrm{Aut}(P_i)$ for the quotients $P_i = P/\gamma_i(P)$.

In all three algorithms the lifting procedure accounts for most of the computational work done in the majority of cases. The lifting problem in the more specialised $p$-group algorithm can be reduced to finding a subspace stabiliser in a matrix group, which is a much simpler calculation than is required for the more general lifting problem used in the soluble and finite group algorithms. Thus, the $p$-group algorithm is almost always considerably faster than the more general approaches.

### 2.6.1 Constructing permutation representations of automorphism groups

In later chapters we will need to construct permutation representations for automorphism groups of finite groups. Given a finite group $G$ and a group of automorphisms $A \leq \mathrm{Aut}(G)$, the general approach is to construct a union of conjugacy classes $U$ that are closed under the action of $A$ such that the normal closure of $U$ in $G$, i.e. the largest normal subgroup of $G$ containing $U$, is $G$. Then the permutation action of $A$ on $U$ is faithful.

## 2.7 Other useful algorithms

Throughout this thesis, we will assume that the following algorithms are available. We give a brief listing here only detailing the input/output of each procedure. For more details, please refer to [HEO05].

- SylowSubgroup$(G, p)$ : A finite group $G$ and a prime $p \in \pi(G)$.

  Returns a Sylow $p$-subgroup of $G$.

- HallSubgroup$(G, \pi)$ : A finite soluble group $G$ and a subset $\pi \subseteq \pi(G)$.

  Returns a Hall $\pi$-subgroup of $G$.

- pCore$(G, p)$ : A finite group $G$ and a prime $p \in \pi(G)$.

  Returns $O_p(G)$, the $p$-core of $G$.

- Core$(G, \pi)$ : A finite group $G$ and a subset $\pi \subseteq \pi(G)$.

  Returns $O_\pi(G)$, the largest normal $\pi$-subgroup of $G$.

- Normaliser$(G, H)$ : A finite group $G$, and a subgroup $H \leq G$.

  Returns $N_G(H)$.

- Centraliser$(G, H)$ : A finite group $G$, and a subgroup $H \leq G$.

  Returns $C_G(H)$.

- IsConjugate$(Z, x, y)$ : A finite group $X$ with $Z \leq X$ and elements $x, y \in X$.

  Returns (true, $z$) if there exists an element $z \in Z$ such that $x^z = y$, or (false, null) otherwise.

- IsHomomorphism$(G, H, \alpha)$ : Finite groups $G, H$ and a mapping $\alpha : G \to H$.

  Returns true if $\alpha$ is a homomorphism, or false if not.

- InnerAutomorphism$(G, g)$ : A finite group $G$ and $g \in G$.

  Returns the inner automorphism, so $\overline{g} : x \mapsto g^{-1}xg$ for $x \in G$.

We assume that all of these algorithms are available for use with each group structure that we pass them.

# Chapter 3

# Preliminary results and algorithms

In this chapter we outline results and algorithms which will prove useful when constructing practical implementations of the algorithms in later chapters. In particular, we will give details of how to exploit the structure of automorphism groups of $p$-groups to allow for more efficient computations.

Aside from the problem of constructing an automorphism group of a finite group $G$, it is often impossible to calculate a good representation for $\mathrm{Aut}(G)$ thus hugely limiting its use in further calculations. Typically, an automorphism group is defined by a generating set of automorphism maps. Although this is sufficient for carrying out simple operations (i.e. those which only rely on the evaluation of elements under the generators), more complicated structural computations such as testing membership of random elements and determining normalisers of subgroups, often prove to be impractical. Hence before any complicated computations can be attempted, it is necessary to first construct a more useful representation of the automorphism group.

Constructing manageable representations of groups is a frequent limiting factor throughout computational group theory. Therefore we devote some time here to discussing the various forms of group representation used for computation, and how they are selected based on the calculations that need to be performed. We refer the interested reader to [HEO05] for a more detailed discussion of the different data structures used to represent groups and their suitability to different algorithms. For the purposes of this thesis, we consider several different types of group representations commonly used for computations, and outline our preferences in a brief summary here.

- **Polycyclic Group** We prefer to use polycyclic presentations when the group can be determined to be soluble. In general, polycyclic groups allow for efficient calculations and are not as limiting (in terms of both space and computation time) as other group types.

- **Matrix Group** In cases where a $p$-group is known to be elementary abelian and calculations are to be evaluated in $\mathrm{Aut}(P)$, we construct $\mathrm{Aut}(P)$ as a matrix group and perform operations in the matrix group where appropriate. In particular, we make use of GLNormaliser described in [Cou11], to compute normalisers in subgroups of $\mathrm{GL}(n, p)$, and recent work by Holt, O'Brien and Leedham-Green on the "Large Matrix Group" summarised in [HLGO11], which is implemented in MAGMA, and provides a wide range of fast algorithms for computing in large matrix groups.

- **Permutation Group** Where no better representation can be found, we construct a permutation representation. Unfortunately, this is sometimes impractical and often acts as the main stumbling block to most of the procedures which follow. Despite the fact that computation in permutation groups is a very well developed area, calculations are commonly limited by the degree of the permutation groups involved. Finding low degree permutation representations is a hard problem and, in practice, continuing with larger degree representations is often best.

We begin by presenting algorithms for handling automorphism groups of $p$-groups.

## 3.1 Computing with automorphism groups of $p$-groups

In this section we give a brief analysis of the data output of the automorphism group algorithm for $p$-groups introduced by O'Brien in [O'B92], and further improved by O'Brien, Eick and Leedham-Green in [ELGO02]. In particular, we describe the typical output of the algorithm, and show how this can be manipulated to construct efficient representations of the resulting automorphism group.

There are several approaches which can be used to construct practical representations of automorphism groups of $p$-groups. Following our preferences for group representations outlined at the beginning of this chapter, we will describe how to construct polycyclic presentations for soluble automorphism groups of $p$-groups and how to use matrix groups $\mathrm{GL}(n,p)$ to construct $\mathrm{Aut}(P)$ for elementary abelian $p$-groups $P$. As discussed earlier, where no better alternative can be found, we revert to using permutation representations.

Recall Definition 2.26, and for a finite $p$-group $P$ define $P_i = P/\gamma_i(P)$ for $1 \leq i \leq c$ where $P$ has exponent-$p$ class $c$ for some $c \in \mathbb{N}$. Hence we have the following series of epimorphisms:

$$P = P_c \to P_{c-1} \to \dots P_2 \to P_1 = 1.$$

For an automorphism $\alpha \in \mathrm{Aut}(P)$, we will use $\alpha|_{P_i}$ to denote the induced action of $\alpha$ on $P_i$, where $1 \leq i \leq c$.

The output of the $p$-group automorphism group algorithm for a $p$-group $P$ returns a set of generators $A$, such that $\langle A \rangle = \mathrm{Aut}(P)$. This set has a so-called "hybrid" structure, with each generator being one of two distinct types:

- **Type 1** act non-trivially on $P_2$ and generate a subgroup $M$ with $M|_{P_2}$ isomorphic to a subgroup of $\mathrm{GL}(d,p)$ where $|P_2| = p^d$; or

- **Type 2** act trivially on $P_2$ and generate $A_p = \langle \alpha \in A : \alpha|_{P_2} = \mathrm{Id}_{P_2} \rangle$, a normal $p$-subgroup of $\mathrm{Aut}(P)$.

We note that there may be a normal $p$-subgroup of $\mathrm{Aut}(P)$ larger than $A_p$, so we don't always have $A_p = O_p(\mathrm{Aut}(P))$.

We may now consider the output of AutomorphismGroupPGroup$(P)$ to be of the form $(M, A_p)$ where $M$ is the group generated by **Type 1** automorphism generators and $A_p$ is generated by **Type 2** automorphism generators of $\mathrm{Aut}(P)$.

Given this construction, it is natural to define a homomorphism $\phi : \mathrm{Aut}(P) \to M|_{P_2}$ which is defined by the induced action on $P_2$: thus $\phi : \alpha \mapsto \alpha|_{P_2}$ for $\alpha \in \mathrm{Aut}(P)$. Then $\mathrm{Ker}\,\phi \cong A_p$, and $\mathrm{Aut}(P)$ is the extension of $M|_{P_2}$ by $A_p$. In particular, if $|A_p| = 1$, then $\mathrm{Aut}(P) \cong M|_{P_2}$ and we can use the matrix group as a representation for $\mathrm{Aut}(P)$.

For further details, we refer the reader to notes for the AutPGrp package [EO09] from the GAP computer algebra system [GAP08]. The Magma version of the algorithm produces an equivalent structure as output.

### 3.1.1 Determining solubility and computing PC presentations

In this section we briefly outline how to construct PC presentations of $\mathrm{Aut}(P)$ for a finite $p$-group $P$ of exponent-$p$ class $c$ where $c \in \mathbb{N}$. The strategy is to use the "hybrid" structure of the automorphism group algorithm output detailed in the previous section, and we begin by determining the solubility of $\mathrm{Aut}(P)$. Define $(M, A_p) := \mathrm{Aut}(P)$ as in the previous section. Then $\mathrm{Aut}(P)$ is soluble if and only if $M|_{P_2}$ is soluble, since it is an extension of $M|_{P_2}$ by $A_p$.

To construct a PC representation of $\mathrm{Aut}(P)$, we first build a PC presentation of $A_p$. Using existing methods, we then obtain a PC presentation for the matrix group $M|_{P_2}$, and extend this by the PC presentation of $A_p$ to get a PC representation of $\mathrm{Aut}(P)$. We make several assumptions about the structure of the output of the algorithm and generators of $P$, which are specific to the implementation we are using.

Firstly, we assume that we can readily compute a PC presentation of $P$ for which consecutive subsequences of generators in the polycyclic sequence of $P$ generate the consecutive quotient groups $P_2, P_3, \dots P_c$. In other words, if we define $\phi_i : P \to P_i$ to be the natural maps for $2 \leq i \leq c$, and assume that $X = [x_1, \dots, x_n]$ is a polycyclic sequence for $P$ with this property, then:

$$\begin{aligned}
P_2 &= \langle x_1, \dots, x_{r_2} \rangle^{\phi_2} \\
P_3 &= \langle x_1, \dots, x_{r_2}, x_{r_2+1}, \dots, x_{r_3} \rangle^{\phi_3} \\
&\vdots \\
P_{c-1} &= \langle x_1, \dots, x_{r_2}, x_{r_2+1}, \dots, x_{r_3}, x_{r_3+1}, \dots, x_{r_{c-1}} \rangle^{\phi_{c-1}} \\
P_c &= \langle x_1, \dots, x_{r_2}, x_{r_2+1}, \dots, x_{r_3}, x_{r_3+1}, \dots, x_{r_{c-1}}, x_{r_{c-1}+1}, \dots, x_{r_c} \rangle^{\phi_c} = P,
\end{aligned}$$

where each $r_i$ is the smallest possible value, and $r_c = n$. We can therefore split the polycyclic generating sequence of $P$ into generators of specific layers; thus we define

$$X_i := \{x_1, \dots, x_{r_i}\}, \tag{3.1}$$

where $\langle X_i \rangle^{\phi_i} = P_i$ for $2 \leq i \leq c$. Note that $X_1$ is undefined.

If the $p$-group automorphism group algorithm is given a $p$-group with such a presentation, then the resulting automorphism group generators can similarly be classified into layers which correspond to their action on the generators of $P$. So we define $L_{(i)}$, which denotes the set of all $i$-th layer automorphisms of $P$, as follows:

$$L_{(i)} = \left\{ \alpha \in \mathrm{Aut}(P) : \alpha|_{P_j} = \mathrm{Id}_{P_j} \text{ for } 0 \leq j < i \text{ and } \alpha|_{P_i} \neq \mathrm{Id}_{P_i} \right\},$$

for $1 \leq i \leq c$. Note that $\langle\, L_{(k)} \,\rangle / \langle\, L_{(k+1)} \,\rangle$ is elementary abelian for $3 \leq k \leq c-1$.

If $A$ is the generating set returned by the algorithm, then we can define $A_{(i)}$, which denotes the $i$-th layer generators, as $A_{(i)} = A \cap L_{(i)}$. In particular, note that $A_{(1)}$ is empty, $A_{(2)}$ is the set of generators of $A$ corresponding to the group $M|_{P_2}$ which act non-trivially on the elementary abelian $p$-group $P_2$; and

$$A_p = \left\langle \bigcup_{i=3}^{c} A_{(i)} \right\rangle .$$

Unfortunately, it is not always true that $\langle L_{(i)} \rangle = \langle A_{(i)} \rangle$ for each $i$.

Given an arbitrary automorphism $\alpha \in \mathrm{Aut}(P)$ where $\alpha \neq \mathrm{Id}(P)$, we can now find $k \in \mathbb{N}$ such that $\alpha \in L_{(k)}$. If we assume that $\alpha|_{P_k}$ can be expressed as a product of automorphisms from $A_{(k)}$, then we have

$$\alpha|_{P_k} = (\beta_{k,1} \cdots \beta_{k,n_k})|_{P_k}$$

where each $\beta_{k,i} \in A_{(k)}$ for $1 \leq i \leq n_k$ (we will address the problem of constructing this product later). Following a similar process for $\alpha(\beta_{k,1} \ldots \beta_{k,n_k})^{-1} \in L_{(l)}$ for some $l > k$, we can construct a product of generators from the sets $A_{(k)}, A_{(k+1)}, \ldots, A_{(c)}$ which is equivalent to the action of $\alpha$ on $P$. Thus we have

$$\alpha = (\beta_{k,1} \cdots \beta_{k,n_k})(\beta_{k+1,1} \cdots \beta_{k+1,n_{k+1}}) \cdots (\beta_{c,1} \cdots \beta_{c,n_c}) \tag{3.2}$$

where each $\beta_{j,i} \in A_{(j)}$ for $1 \leq i \leq n_j$, and $k \leq j \leq c$.

We can now construct a PC presentation for $A_p$ as follows:

1. Compute power relations for each $\alpha \in A_{(i)}$ for $3 \leq i \leq c$:
   (a) Find $k$ such that $\alpha^p \in L_{(k)}$, where $k > i$.
   (b) Find a product of automorphisms from $A_{(k)}$, $A_{(k+1)}, \ldots, A_{(c)}$ whose action on $P$ is equivalent to that of $\alpha^p$, as in (3.2). This is the right hand side

of the relation.

2. Compute conjugation relations $\alpha_i^{\alpha_j}$ where $\alpha_i \in A_{(i)}$, $\alpha_j \in A_{(j)}$ and $\alpha_i \neq \alpha_j$ for $3 < i \leq j < c$.

   (a) Find $k$ where $\alpha_i^{\alpha_j} \in L_{(k)}$ for $k \geq j$.

   (b) Find a product of automorphisms from $A_{(k)}, A_{(k+1)}, \dots, A_{(c)}$ whose action on $P$ is equivalent to that of $\alpha_i^{\alpha_j}$, as in (3.2). This is the right hand side of the relation.

If we find an automorphism $\alpha \in L_{(k)}$ which cannot be represented by the automorphism generators in $A_{(k)}$ for some $3 \leq k \leq c$, then we append $\alpha$ to $A_{(k)}$ and continue, ensuring that new power and conjugation relations are added accordingly.

Recall that $\langle A_{(2)} \rangle \cong M|_{P_2}$. Since we can construct the matrix group $M|_{P_2}$, we use existing methods to compute a PC presentation for it. Then we begin to combine this with the PC presentation for $A_p$ that has already been determined. First, we extend each PC relation $R(\alpha_1, \alpha_2, \dots, \alpha_n)|_{P_2} = \mathrm{Id}_{P_2}$ of the PC representation of $M|_{P_2}$, where $\alpha_1, \dots, \alpha_n \in A_{(2)}$, so that they are made to be consistent on the whole of $P$. Hence:

1. Compute $\rho := R(\alpha_1, \dots \alpha_n)$.

2. Find $k$ such that $\rho^{-1} \in L_{(k)}$.

3. Find a product of automorphisms from $A_{(k)}, A_{(k+1)}, \dots, A_{(c)}$ whose action on $P$ is equivalent to that of $\rho^{-1}$, as in (3.2)

4. Then append this product to $R(\alpha_1, \dots \alpha_n)$, and the relation is now consistent on $P$.

Similarly, we compute the conjugation relations between $A_{(2)}$ and $A_{(i)}$ for $3 \leq i \leq c$ by following the same method.

One of the more complex computational challenges in this calculation is to construct a product of automorphisms, as in (3.2), for a given $\alpha \in L_{(k)}$. In general, this is a very hard problem but the groups involved here enjoy special properties which allow us to reduce the calculation into simple linear algebra.

When considering the action of a given automorphism $\alpha \in \mathrm{Aut}(P)$ on some $P_k$ for $3 \leq k \leq c$, it is sufficient to consider the image of $X_2$ under $\alpha$, since $P_2 = P/\Phi(P)$ and so $P = \langle X_2 \rangle$. Further, as both $P_2$ and $P_{k-1}/P_k$ are elementary abelian $p$-groups, they can be regarded as vector spaces over $\mathbf{F}_p$, and hence any $\alpha|_{P_k}$ can be thought of as an element of the vector space $\mathrm{Hom}(P_2, P_{k-1}/P_k)$.

So for each generating set $A_{(k)}$ construct a vector space $V_k = \mathrm{Hom}(P_2, P_{k-1}/P_k)$ over $\mathbf{F}_p$ where $\dim V_k = |X_2| \cdot (|X_k| - |X_{k-1}|) = d_k$. Temporarily relabelling generators for simplicity, suppose that $X_2 = \{a_1, \ldots, a_r\}$, and $X_k \setminus X_{k-1} = \{b_1, \ldots b_s\}$. We can now define a basis $\{e_{i,j}\}$, where $1 \leq i \leq r$ and $1 \leq j \leq s$, for $V_k$ in the obvious way, so

$$e_{i,j} : \begin{cases} a_i \mapsto b_j \\ a_l \mapsto 0 \quad l \neq i \end{cases}$$

and $e_{i,j}$ corresponds to an automorphism of $P_k$ which maps $a_i \mapsto a_i b_j$ and $a_l \mapsto a_l$ for $l \neq i$.

Each mapping $\alpha \in A_{(k)}$ now has a corresponding vector $v_\alpha \in V_k$ and so we can define a subspace $S_k = \langle v_\alpha : \alpha \in A_{(k)} \rangle$. Given an arbitrary automorphism $\alpha \in L_{(k)}$ we compute $v_\alpha \in V_k$, and if $v_\alpha \in S_k$ then we can easily find an expression for $v_\alpha$ as a linear combination of the basis vectors of $S_k$. Thus we have now constructed a product of elements of $A_{(k)}$ which give the equivalent action of $\alpha$ on $P_k$.

We now give a brief worked example to demonstrate some of these ideas in action. We begin by constructing a suitable 2-group `P` of order 64, from the small groups database [BEO02], and use the `pRanks` function to determine the layers of the generators of `P`. Note that in general we can use the MAGMA function `SpecialPresentation` to construct a presentation for `P` which has the properties that we assume. However, every $p$-group constructed directly from the small groups database already has such a presentation, so we don't include it in this calculation.

```
> SetSeed(1);
> P := SmallGroup(64, 10);
> pRanks(P);
[ 2, 4, 6 ]
```

The $i$-th generator of the group `P` is denoted by `P.i` in Magma. So this output implies that `P.1` and `P.2` are layer 2; `P.3` and `P.4` are layer 3; and `P.5` and `P.6` are layer 4 generators. We now compute the automorphism group of `P`, and output the automorphism generators that are returned as a result.

```
> A := AutomorphismGroupPGroup(P);
> [ < i, A.i > : i in [1..Ngens(A)] ];
[
    <1, Automorphism of GrpPC : P which maps:
        P.1 |--> P.1 * P.2
        P.2 |--> P.2
        P.3 |--> P.3 * P.6
        P.4 |--> P.3 * P.4 * P.6
        P.5 |--> P.5 * P.6
        P.6 |--> P.6>,
    <2, Automorphism of GrpPC : P which maps:
        P.1 |--> P.1 * P.4
        P.2 |--> P.2
        P.3 |--> P.3 * P.5 * P.6
        P.4 |--> P.4 * P.5
        P.5 |--> P.5
        P.6 |--> P.6>,
    <3, Automorphism of GrpPC : P which maps:
        P.1 |--> P.1
        P.2 |--> P.2 * P.6
        P.3 |--> P.3
        P.4 |--> P.4
        P.5 |--> P.5
        P.6 |--> P.6>,
    <4, Automorphism of GrpPC : P which maps:
        P.1 |--> P.1 * P.6
        P.2 |--> P.2
        P.3 |--> P.3
        P.4 |--> P.4
        P.5 |--> P.5
        P.6 |--> P.6>,
    <5, Automorphism of GrpPC : P which maps:
        P.1 |--> P.1
        P.2 |--> P.2 * P.3
        P.3 |--> P.3 * P.5
        P.4 |--> P.4
        P.5 |--> P.5
        P.6 |--> P.6>,
    <6, Automorphism of GrpPC : P which maps:
        P.1 |--> P.1 * P.3 * P.6
        P.2 |--> P.2
        P.3 |--> P.3 * P.6
        P.4 |--> P.4 * P.5 * P.6
        P.5 |--> P.5
        P.6 |--> P.6>
]
```

By inspection, we see that `A.1` $\in A_{(2)}$; `A.2`, `A.5`, `A.6` $\in A_{(3)}$; and `A.3`, `A.4` $\in A_{(4)}$. We now start to construct power relations. It is easy to check that `A.2^2` and `A.6^2` are both the identity. Now,

```
> A.5^2;
Automorphism of GrpPC : P which maps:
    P.1 |--> P.1
    P.2 |--> P.2 * P.5 * P.6
    P.3 |--> P.3
    P.4 |--> P.4
    P.5 |--> P.5
    P.6 |--> P.6
```

and A.5^2 $\in L_{(4)}$ but can't be represented as a product of A.3 and A.4, thus we add A.5^2 to $A_{(4)}$. Continuing in this manner we determine the remaining power relations and conjugation relations.

We note that construction of the vector spaces $V_k$ and $S_k \subseteq V_k$ is straightforward using the action of the generators A.i on P.1 and P.2. For instance, $V_3$ is a $2 \cdot 2$ dimensional space over $\mathbf{F}_2$, and a mapping of automorphisms in $A_{(3)}$ to a basis for $S_3 \subseteq V_3$ would be:

$$\begin{aligned} \texttt{A.2} &\mapsto (\ 0\ 1\ 0\ 0\ ) \\ \texttt{A.5} &\mapsto (\ 0\ 0\ 1\ 0\ ) \\ \texttt{A.6} &\mapsto (\ 1\ 0\ 0\ 0\ ) \end{aligned}$$

Given some arbitrary automorphism, a $\in L_{(3)}$, defined as:

```
> a;
Automorphism of GrpPC : P which maps:
    P.1 |--> P.1 * P.3 * P.4 * P.5
    P.2 |--> P.2 * P.3 * P.6
    P.3 |--> P.3
    P.4 |--> P.4 * P.6
```

```
P.5 |--> P.5

P.6 |--> P.6
```

we can compute the associated vector: ( 1 1 1 0 ), and thus deduce that `a` = `A.2 * A.5 * A.6`.

## 3.2 Fix subgroup calculations

Let $G$ be a finite group and suppose that $H$ is a subgroup of $G$. Define $\mathrm{Aut}(G)_H = \{\alpha \in \mathrm{Aut}(G) : H^\alpha = H\}$ and assume that $\mathrm{Aut}(G)$ has been computed and we want to find $\mathrm{Aut}(G)_H$. In this section we outline several different approaches, beginning with the most general and adding refinements or giving alternative algorithms where some known structure of $\mathrm{Aut}(G)$ allows for more efficient computation of the result.

We include a pseudocode prototype for the full resulting algorithm FixSubgroup (Algorithm 3.1) below, for ease of reference later.

**Algorithm 3.1:** FixSubgroup

```
Input: $(A, H)$ : A group of automorphisms $A$ of a group $G$, and $H$ a
       subgroup of $G$
Output: $A_H$ : The subgroup of $A$ fixing $H$
```

### 3.2.1 Orbit stabiliser

The most basic approach to this problem involves computing the orbit of the given subgroup $H$ under the action of $\mathrm{Aut}(G)$. We follow the standard OrbitStabiliser procedure as given in Algorithm 2.1. Due to the processing time and memory needed to maintain the list of orbit elements, this approach is only viable when the orbit of $H$ under $\mathrm{Aut}(G)$ is small. However, situations where large orbits arise can be predicted - for instance if $|G : H|$ is large - and there are a few tricks which can be applied to reduce the size of the orbit before it is computed (we refer the reader to Section 3.2.2). We use Schreier vectors to avoid storing accompanying automorphisms with orbit elements.

In cases where $\mathrm{Aut}(G)$ is soluble and we have a polycyclic presentation for $\mathrm{Aut}(G)$, we can use PolycyclicOrbitStabiliser (as shown earlier in Algorithm 2.2) to take advantage of the PC structure of $\mathrm{Aut}(G)$. Again, we need to avoid computing large orbits if possible, so apply the same orbit reduction techniques for the standard orbit stabiliser algorithm where appropriate.

### 3.2.2 Handling large orbits

There are cases where the orbits produced in orbit-stabiliser calculations are very large - for instance if $|G : H|$ is large. In practice this can be extremely problematic due to the amount of time and memory required to identify and store each orbit element. In this situation, it is desirable to reduce the orbit as much as possible before performing the orbit-stabiliser algorithm. Recall the definition of $\overline{H}$ for a subgroup $H \leq G$ of a group $G$.

**Lemma 3.1.** *Let $G$ be a finite group, $H \leq G$. Then $\mathrm{Aut}(G)_H \leq N_{\mathrm{Aut}(G)}(\overline{H})$. Furthermore, if $Z(G) = 1$ then $\mathrm{Aut}(G)_H = N_{\mathrm{Aut}(G)}(\overline{H})$.*

*Proof.* Firstly note that for $\alpha \in \mathrm{Aut}(G)$, $g \in G$ and $h \in H$ we have

$$g^{\overline{h}^{\alpha}} = g^{\alpha^{-1}\overline{h}\alpha} = (h^{-1})^{\alpha}(g^{\alpha^{-1}})^{\alpha}h^{\alpha} = g^{\overline{h^{\alpha}}}. \tag{3.3}$$

Now take $\alpha \in \mathrm{Aut}(G)_H$. Since $\alpha$ fixes $H$, $\overline{h^{\alpha}} \in \overline{H}$, and thus $\alpha \in N_{\mathrm{Aut}(G)}(\overline{H})$. Now suppose that $Z(G) = 1$, and hence $H \cong \overline{H}$. Take $\alpha \in N_{\mathrm{Aut}(G)}(\overline{H})$, and for $h \in H$ we have $\overline{h}^{\alpha} \in \overline{H}$, and so $\overline{h}^{\alpha} = \overline{h^{\alpha}} \in \overline{H}$. Since $Z(G) = 1$, $\alpha$ fixes $H$. □

In cases where $G$ has trivial centre performing a full orbit stabiliser calculation is equivalent to evaluating a normaliser in $\mathrm{Aut}(G)$. Of course, this whole procedure relies on having constructed a manageable representation for $\mathrm{Aut}(G)$, and we refer to earlier comments about this at the beginning of the chapter. As we will mostly be performing this calculation on $\mathrm{Aut}(P)$ for some $p$-group $P$, this result can only be used to reduce the size of the orbit (since $Z(P) > 1$ for any $p$-group $P$). However, practice has shown that this trick does increase the scope of the algorithms in this thesis.

### 3.2.3 Elementary abelian $p$-groups

Suppose now that $G$ is an elementary abelian $p$-group. Then $\mathrm{Aut}(G) = \mathrm{GL}(n,p)$ and constructing a manageable (degree $< 10000$) permutation representation for $\mathrm{Aut}(G)$ becomes difficult even for relatively straightforward automorphism groups (recall that $|GL(n,p)| = (p^n-1)(p^n-p)(p^n-p^2)\dots(p^n-p^{n-1})$). However, if we construct $\mathrm{Aut}(G)$ directly as $\mathrm{GL}(n,p)$, we can represent $G$ as a vector space $V$ over the field $\mathbf{F}_p$ which is acted on by $GL(n,p)$. Given any subgroup $H \leq G$, we can find the equivalent subspace, say $W \subseteq V$, and our problem is reduced to finding the stabiliser of $W$ in $\mathrm{GL}(n,p)$.

In the next result, we assume that $\mathrm{GL}(n,p)$ is generated by two matrices $A_{n,p}$ and $B_{n,p}$, and we use $E_{i,j}$ to denote the matrix with $1_{\mathbf{F}_p}$ in the $i^{\text{th}}$ row and $j^{\text{th}}$ column.

**Lemma 3.2.** *Let $V$ be an $n$-dimensional vector space over a finite field $K$, and suppose that $W \subset V$ is a subspace of $V$ of dimension $r$. Then the subgroup of $\mathrm{GL}(n,K)$ fixing $W$, after basis change, is generated by*

$$\begin{pmatrix} A_{r,p} & 0 \\ 0 & I_s \end{pmatrix}, \begin{pmatrix} B_{r,p} & 0 \\ 0 & I_s \end{pmatrix}, \begin{pmatrix} I_r & 0 \\ 0 & A_{s,p} \end{pmatrix}, \begin{pmatrix} I_r & 0 \\ 0 & B_{s,p} \end{pmatrix}, \text{ and } \begin{pmatrix} I_r & 0 \\ E_{r+s,1} & I_s \end{pmatrix}.$$

*Proof.* Choose a basis $w_1, \ldots, w_r$ of $W$ and extend this to a basis of $V$, say

$$w_1, \ldots, w_r, v_{r+1}, \ldots, v_{r+s}.$$

Now in the new basis we can write down generators for the subgroup of $\mathrm{GL}(n, p)$ fixing $W$, namely those which act on $W$ as $\mathrm{GL}(r, p)$ and fix the rest of $V$, those which fix $W$ and act on the rest of $V$ as $\mathrm{GL}(s, p)$.

Applying change of basis matrices to revert back to the original basis of $V$ gives the result. □

## 3.3 Find conjugating element

Given a group $G$ with subgroups $X, Y \leq G$ and an automorphism $\alpha \in \mathrm{Aut}(Y)$ we need to test if $\alpha$ is an inner automorphism of $Y$ induced by an element $x \in X$, and if so determine $x$. The full procedure, FindConjugatingElement, is outlined in Algorithm 3.2, and is taken from [CH03, 2.4]. We may refer to this function in later pseudocode as returning only one value, the element $x$, when we know that the action of $\alpha$ is conjugation.

## 3.4 Find mapping automorphism

Given two subgroups $H, K \leq G$ with $|H| = |K|$, is there an automorphism $\alpha \in \mathrm{Aut}(G)$ such that $H^\alpha = K$? We follow the same approach as in the OrbitStabiliser algorithm: compute the orbit of $H$ under the action of $\mathrm{Aut}(G)$, stopping if we find $K$ in the orbit. If $K$ is not in the orbit of $H$ under $\mathrm{Aut}(G)$ then there is no such $\alpha$. Again, we use Schreier vectors to minimise the storage requirement of the algorithm in the normal case, and if $\mathrm{Aut}(G)$ is polycyclic then we can use the more efficient PolycyclicOrbitStabiliser in place of the standard orbit stabiliser procedure.

**Algorithm 3.2:** FindConjugatingElement

```
Input: $(X, Y, \alpha)$ : $X$, $Y$ are groups such that $X$ acts on $Y$ by conjugation,
       and $\alpha \in \mathrm{Aut}(Y)$
Output: $(b, x)$ : A boolean $b$, true if a conjugate is found, false otherwise,
        and an element $x \in X$ such that $y^\alpha = y^x$ for all $y \in Y$, or null if
        no such element exists

$Z := X$;
$x := 1_X$;
// Iterate over the generators of $Y$
for $y \in$ Generators($Y$) do
    $b, c :=$ IsConjugate($Z, y^x, y^\alpha$);
    if $b \neq$ true then
        return false, null;
    $x := xc$;
    $Z := C_Z(y^\alpha)$;
return true, $x$;
```

We include the pseudocode prototype for FindMappingAutomorphism (Algorithm 3.3) here, for ease of reference later.

**Algorithm 3.3:** FindMappingAutomorphism

```
Input: $(A, H, K) : A$ is a group of automorphisms of $G$ and $H, K \leq G$
Output: $(b, \alpha) : b$ a boolean value, true if a map is found, false otherwise,
        and an automorphism $\alpha \in \mathrm{Aut}(G)$ such that $H^\alpha = K$ if such a
        map exists, null otherwise
```

# Chapter 4

# General methods

In this chapter we outline general methods which form the basis of our algorithms for constructing automorphism groups and performing isomorphism testing for soluble groups (see Chapter 5 and Chapter 6) and further to some non-soluble examples (see Chapter 7).

We use two general strategies for both computing automorphism groups and performing isomorphism testing:

- Construct an embedding into a product of smaller groups and combine the result of the calculation on each factor.
- Solve the problem for a subgroup, or a corresponding pair of subgroups in the isomorphism case, and then extend the result to the whole group.

We begin by outlining the results for automorphism groups and then adapt these ideas to perform isomorphism testing.

## 4.1 Automorphism group

In this section we derive results which will be used later to construct automorphism groups. We begin by stating the result that forms the basis of our general approach.

**Theorem 4.1.** *Let $G$ be a group and $P \in \mathrm{Syl}_p(G)$ for some prime $p \in \pi(G)$. Define $\Gamma = \{\alpha \in \mathrm{Aut}(G) : P^\alpha = P\}$ and let $I = \mathrm{Inn}(G)$. Then* $\mathrm{Aut}(G) = \Gamma I$.

*Proof.* Take any $\alpha \in \mathrm{Aut}(G)$. Then using the result of Lemma 2.17, $P^\alpha = g^{-1}Pg$ for some $g \in G$ and thus $\alpha\overline{g^{-1}} \in \Gamma$. Hence we express $\alpha$ as a product $\alpha = \alpha\overline{g^{-1}}\,\overline{g}$, and the result follows. □

**Definition 4.2.** Let $G$ be a finite group and suppose that $H, K \leq G$. Define $\Gamma = \{\alpha \in \mathrm{Aut}(G) : H^\alpha = H, K^\alpha = K\}$. Then we say that $\mathrm{Aut}(G)$ is *determined by restriction to the action on $H$ and $K$* if

$$\mathrm{Aut}(G) = \langle \Gamma, \mathrm{Inn}(G) \rangle.$$

In each of the following sections we give specific structural criteria for a finite group $G$ and show how to compute the automorphism group $\mathrm{Aut}(G)$.

### 4.1.1 Direct product

In situations where a group $X$ is known to be a direct product of characteristic subgroups $U, V \leq G$, computing $\mathrm{Aut}(X)$ can be reduced to two sub-problems: computing $\mathrm{Aut}(U)$ and $\mathrm{Aut}(V)$, as demonstrated in the following theorem.

**Remark.** For groups $X, Y$ we will regard $\mathrm{Aut}(X) \times \mathrm{Aut}(Y)$ as a subgroup of $\mathrm{Aut}(X \times Y)$. Given automorphisms $\chi \in \mathrm{Aut}(X)$ and $\upsilon \in \mathrm{Aut}(Y)$ we construct the combined mapping denoted $(\chi, \upsilon) : X \times Y \to X \times Y$ defined by $(\chi, \upsilon) : (x, y) \mapsto (x^\chi, y^\upsilon) \in \mathrm{Aut}(X \times Y)$ for $x \in X, y \in Y$.

**Theorem 4.3.** *Let $X$ be a group with characteristic subgroups $U$ and $V$ such that $X = UV$ and $U \cap V = 1$. Then $X \cong U \times V$ and* $\mathrm{Aut}(X) \cong \mathrm{Aut}(U) \times \mathrm{Aut}(V)$.

*Proof.* Firstly, $U, V \,\mathrm{char}\, X$ and hence $U, V \lhd X$, so by Proposition 2.4 and Proposition 2.3, $X \cong U \times V$. For each $x \in X$ there are elements $u \in U$ and $v \in V$

such that $x = uv$, and this expression is unique. Consider the action of an automorphism $\alpha \in \mathrm{Aut}(X)$ on an element $x \in X$. We have $x^\alpha = (uv)^\alpha = u^\alpha v^\alpha$, and as both $U$ and $V$ are characteristic subgroups of $X$, $u^\alpha \in U$ and $v^\alpha \in V$. In addition we have $\alpha|_U \in \mathrm{Aut}(U)$, $\alpha|_V \in \mathrm{Aut}(V)$, and $\{\alpha|_U : \alpha \in \mathrm{Aut}(X)\} \leq \mathrm{Aut}(U)$ and similarly $\{\alpha|_V : \alpha \in \mathrm{Aut}(X)\} \leq \mathrm{Aut}(V)$. Hence we can define a mapping $\Phi : \mathrm{Aut}(X) \to \mathrm{Aut}(U) \times \mathrm{Aut}(V)$ by

$$\Phi : \alpha \mapsto (\alpha|_U, \alpha|_V),$$

which is clearly a homomorphism. Now suppose that $\alpha^\Phi = 1_{\mathrm{Aut}(U)\times\mathrm{Aut}(V)}$ for some $\alpha \in \mathrm{Aut}(X)$. Then $\alpha|_U = \mathrm{Id}_U$ and $\alpha|_V = \mathrm{Id}_V$, and hence $\alpha = \mathrm{Id}_X$ since $X = U \times V$. Therefore $\Phi$ is injective.

Now take any $\overline{\alpha} \in \mathrm{Aut}(U) \times \mathrm{Aut}(V)$, and note that by the previous remark $\overline{\alpha}$ induces an automorphism $\alpha = (\mu, \nu) \in \mathrm{Aut}(U \times V)$ for some $\mu \in \mathrm{Aut}(U)$ and $\nu \in \mathrm{Aut}(V)$. Hence the mapping $\alpha : uv \mapsto u^\mu v^\nu$ is an automorphism of $X$ and $\Phi$ is surjective, and therefore an isomorphism. □

### 4.1.2 Subdirect product

We begin by introducing the notion of a subdirect product, which is defined as an embedding of a group $G$ into a direct product of two quotient groups of $G$. We outline the construction in the following lemma.

**Lemma 4.4.** *Let $X$ be a group with normal subgroups $U, V \leq X$ such that $U \cap V = 1$. Then the mapping $\psi : X \to X/U \times X/V$ defined by $\psi : x \mapsto (xU, xV)$ for $x \in X$ is a monomorphism.*

*Proof.* Clearly $\psi$ is a homomorphism, so suppose that $x^\psi = 1_{X/U\times X/V}$ for some $x \in X$. Then $(xU, xV) = (U, V)$, and hence $x \in U \cap V = 1$. Therefore $\mathrm{Ker}\,\psi = 1_X$. □

Now we can consider the situation where we need to compute $\mathrm{Aut}(X)$ for some group $X$ which is known to be a subdirect product of some characteristic subgroups $U, V \leq X$. Again, we follow the same basic approach as in the direct product case, and split into two sub-problems: computing $\mathrm{Aut}(X/U)$ and $\mathrm{Aut}(X/V)$, though we then need to apply a search to isolate the automorphisms in $\mathrm{Aut}(X/U) \times \mathrm{Aut}(X/V)$ which induce automorphisms of $X$, as detailed in the following theorem.

**Theorem 4.5.** *Let $X$ be a group with characteristic subgroups $U, V \leq X$ such that $U \cap V = 1$. Define a subdirect product $\psi : X \to X/U \times X/V$ by $\psi : x \mapsto (xU, xV)$ for $x \in X$, and let $A' = \{\overline{\alpha} \in \mathrm{Aut}(X/U) \times \mathrm{Aut}(X/V) : X^{\psi\overline{\alpha}} = X^{\psi}\}$. Then $\mathrm{Aut}(X) \cong A'$.*

*Proof.* Let $\alpha \in \mathrm{Aut}(X)$ and for $x \in X$ we define $\alpha_U \in \mathrm{Aut}(X/U)$ and $\alpha_V \in \mathrm{Aut}(X/V)$ to be $\alpha_U : xU \mapsto (xU)^\alpha = x^\alpha U$ and $\alpha_V : xV \mapsto (xV)^\alpha = x^\alpha V$ respectively. Define $\Psi : \mathrm{Aut}(X) \to \mathrm{Aut}(X/U) \times \mathrm{Aut}(X/V)$ by $\Psi : \alpha \mapsto (\alpha_U, \alpha_V)$ for $\alpha \in \mathrm{Aut}(X)$. Clearly this mapping is a homomorphism. Now suppose that $\alpha^\Psi = 1_{\mathrm{Aut}(X/U)\times\mathrm{Aut}(X/V)}$ for some $\alpha \in \mathrm{Aut}(X)$ and thus $\alpha_U = \mathrm{Id}_{X/U}$ and $\alpha_V = \mathrm{Id}_{X/V}$. Hence $(xU)^\alpha = x^\alpha U = xU$ and $(xV)^\alpha = x^\alpha V = xV$ so $x^{-1}x^\alpha \in U \cap V$ and $x = x^\alpha$. Therefore $\Psi$ is a monomorphism. Now given any $\alpha \in \mathrm{Aut}(X)$, let $\overline{\alpha} = \alpha^\Psi$. Then for $x \in X$, $(xU, xV)^{\overline{\alpha}} = ((xU)^\alpha, (xV)^\alpha) = (x^\alpha U, x^\alpha V) \in X^\psi$. Hence $\overline{\alpha}$ fixes $X^\psi$ so $\overline{\alpha} \in A'$ and $\mathrm{Im}\,\Psi \leq A'$.

Now take $\overline{\alpha} \in A'$. As $\overline{\alpha}$ fixes $X^\psi$ and $\psi$ is a monomorphism, given $x \in X$ we have $(xU, xV)^{\overline{\alpha}} = (x'U, x'V)$ for some $x' \in X$, which is uniquely determined by $x$. Thus $\overline{\alpha}$ induces a mapping $\alpha : X \to X$ defined by $\alpha : x \mapsto x'$. For $x, y \in X$

$$\begin{aligned}
(xU, xV)^{\overline{\alpha}}(yU, yV)^{\overline{\alpha}} &= (x^\alpha U, x^\alpha V)(y^\alpha U, y^\alpha V) \\
&= (x^\alpha y^\alpha U, x^\alpha y^\alpha V), \\
((xU, xV)(yU, yV))^{\overline{\alpha}} &= (xyU, xyV)^{\overline{\alpha}} \\
&= ((xy)^\alpha U, (xy)^\alpha V),
\end{aligned}$$

and since $(xU, xV)^{\overline{\alpha}}(yU, yV)^{\overline{\alpha}} = ((xU, xV)(yU, yV))^{\overline{\alpha}}$ we have $\psi((xy)^{\alpha}) = \psi(x^{\alpha}y^{\alpha})$. Therefore $(xy)^{\alpha} = x^{\alpha}y^{\alpha}$ for all $x, y \in X$ and so $\alpha$ is a homomorphism. Since $\overline{\alpha}$ is an automorphism, $\overline{\alpha}^{-1}$ induces a homomorphism $\alpha^{-1}$ which is both a right and left inverse for $\alpha$. Thus $\alpha$ is an automorphism, $\Psi(\alpha) = \overline{\alpha}$ and $\operatorname{Im}\Psi = A'$, giving $A' \cong \operatorname{Aut}(X)$. □

Unfortunately, using this result to construct an algorithm for computing $\operatorname{Aut}(X)$ for large groups $X$ can be very impractical. Having computed $\operatorname{Aut}(X/U)$ and $\operatorname{Aut}(X/V)$ we need to then construct the subgroup of $\operatorname{Aut}(X/U) \times \operatorname{Aut}(X/V)$ fixing $X^{\Psi}$, hence the whole calculation would rely on FixSubgroup routines. We note that orbit elements will be large (i.e. $|X|$), and the size of the orbit can potentially be huge, particularly when $|\operatorname{Aut}(X/U) \times \operatorname{Aut}(X/V)| >> |\operatorname{Aut}(X)|$. Thus this process could become an inhibiting factor for large examples.

In the next result we introduce a refinement to this process, avoiding this potentially large FixSubgroup computation.

**Theorem 4.6.** *Let $X$ be a group with characteristic subgroups $U, V \leq X$ such that $U \cap V = 1$. Define $\psi_U : X \to X/U$ and $\psi_V : X \to X/V$ to be the natural maps, and let*

$$A_U = \left\{\alpha_U \in \operatorname{Aut}(X/U) : V^{\psi_U\alpha_U} = V^{\psi_U}\right\},$$

$$A_V = \left\{\alpha_V \in \operatorname{Aut}(X/V) : U^{\psi_V\alpha_V} = U^{\psi_V}\right\}, \textit{and}$$

$$A_{\{U,V\}} = \left\{(\Lambda_U, \Lambda_V) \in A_U \times A_V : \left((xUV/U)^{\Lambda_U}\right)^{\psi_U^{-1}} = \left((xUV/V)^{\Lambda_V}\right)^{\psi_V^{-1}} \textit{ for all } x \in X\right\}.$$

*Then $A_{\{U,V\}} \cong \operatorname{Aut}(X)$.*

*Proof.* We claim that $A_{\{U,V\}} \cong A'$ from Theorem 4.5, and thus $A_{\{U,V\}} \cong \operatorname{Aut}(X)$. We begin by checking that $A' \leq A_{\{U,V\}}$. Let $\overline{\alpha} \in A'$ and recall that we write

$\overline{\alpha} = (\alpha_U, \alpha_V) \in \mathrm{Aut}(X/U) \times \mathrm{Aut}(X/V)$. For $x \in X$

$$\begin{aligned}
(xU, xV)^{\overline{\alpha}} &= ((xU)^{\alpha_U}, (xV)^{\alpha_V}) \\
&= (x^{\alpha}U, x^{\alpha}V) \\
&= (yU, yV),
\end{aligned}$$

where $\alpha = \Psi^{-1}(\overline{\alpha})$ and $y \in X$ such that $x^{\alpha} = y$. Now,

$$\begin{aligned}
((xUV)U, (xUV)V)^{\overline{\alpha}} &= ((xVU)^{\alpha_U}, (xUV)^{\alpha_V}) && \Longrightarrow \\
&= ((xV)^{\alpha}U, (xU)^{\alpha}V) && U, V \,\mathrm{char}\, X \;\Longrightarrow \\
&= (yUV, yUV).
\end{aligned}$$

So $(xUV, xUV)^{\overline{\alpha}} = (yUV, yUV)$ and observe that since $U, V \,\mathrm{char}\, X$ we have $\alpha_U \in A_U$ and $\alpha_V \in A_V$. Therefore $\overline{\alpha} \in A_{\{U,V\}}$.

Now suppose that $\overline{\alpha} \in A_{\{U,V\}}$, and $\overline{\alpha} = (\Lambda_U, \Lambda_V) \in A_U \times A_V$. Let $(xU)^{\Lambda_U} = aU$ and $(xV)^{\Lambda_V} = bV$ for some $a, b \in X$. Since $V^{\Lambda_U} = V$ and $U^{\Lambda_V} = U$, we get $(xUV)^{\Lambda_U} = aUV$ and $(xUV)^{\Lambda_V} = bUV$. By definition, $aUV = bUV$ and so $a \in bUV$. Therefore there exists $u \in U$ and $v \in V$ such that $a = buv$, and $\overline{\alpha} : (xU, xV) \mapsto (bvU, bV) = (bvU, bvV)$. So $\overline{\alpha} \in A'$.

□

### 4.1.3 Conjugation action

In the previous two sections we compute the automorphism group of a group $X$ by defining an embedding of $X$ into some product of smaller groups, and then recursing. In this section we consider a different approach which relies on computing the automorphism group of a given subgroup $U \leq X$, and extending automorphisms from some subgroup $\Delta \leq \mathrm{Aut}(U)$ to automorphisms of $X$.

We begin by introducing the notion of associating automorphisms of two subgroups $U, V \leq X$ for a group $X$, such that they are candidates for forming automorphisms of $X$ which lie in a given subgroup $\Lambda \leq \mathrm{Aut}(X)$.

**Definition 4.7.** Let $X$ be a group and suppose that $U, V$ are subgroups of $X$. Then a given $\mu \in \mathrm{Aut}(U)$ is *associated* to $\nu \in \mathrm{Aut}(V)$ if there exists an automorphism $\chi \in \Lambda \leq \mathrm{Aut}(X)$ such that $\chi|_U = \mu$ and $\chi|_V = \nu$, and we write $\mu \sim_\Lambda \nu$. Further, we say that $\mu$ *extends*, or *can be extended*, to an automorphism of $X$ fixing $V$.

In particular, we note the following consequence of this definition.

**Lemma 4.8.** *Let $X$ be a group with subgroups $U, V$ such that $X = UV$ and suppose that $\mu \in \mathrm{Aut}(U)$. Then $\mu$ extends to an automorphism of $X$ fixing $V$ if and only if there exists $\nu \in \mathrm{Aut}(V)$ such that $uv \mapsto u^\mu v^\nu$ for $u \in U, v \in V$ is an automorphism of $X$.*

*Proof.* Every element $x \in X$ can be expressed as a product $x = uv$ for $u \in U$, $v \in V$. By definition, there exists $\chi \in \mathrm{Aut}(X)$ with $\chi|_U = \mu$ and $\chi|_V = \nu$ if and only if $\chi : x \mapsto u^\chi v^\chi = u^\mu v^\nu$ is an automorphism. □

We now state a hypothesis which will be assumed throughout the rest of this section.

**Hypothesis 4.9.** Let $X$ be a group with subgroups $U, V, W \leq X$ such that $X = UV$, $U \cap V = 1$, $W \leq U$, $W \operatorname{char} X$ and $C_V(W) = 1$. Define $\Gamma = \{\alpha \in \mathrm{Aut}(X) : U^\alpha = U, V^\alpha = V\}$.

Using Definition 4.7 we can derive conditions for an automorphism $\mu \in \mathrm{Aut}(U)$ to be associated with some $\nu \in \mathrm{Aut}(V)$, and further, how to construct such a $\nu$ given $\mu$. Recall the definition of $\overline{H}$ for a subgroup $H \leq G$ of a group $G$.

**Lemma 4.10.** *The mapping $\gamma : V \to \overline{V}|_W$ defined by $\gamma : v \mapsto \overline{v}|_W$ for $v \in V$ is a monomorphism.*

*Proof.* As $W$ char $X$ it is easy to see that $v^\gamma \in \mathrm{Aut}(W)$ for each $v \in V$. Furthermore $\gamma$ is clearly a homomorphism, and so assume that $v^\gamma = \mathrm{Id}_W$ for some $v \in V$. Then $v^{-1}wv = w$ for all $w \in W$ and as $C_V(W) = 1$, $v = 1$, so $\gamma$ is a monomorphism. □

**Lemma 4.11.** *Let $\mu \in \mathrm{Aut}(U)$ be such that $\mu$ fixes $W$ and $\mu|_W \in N_{\mathrm{Aut}(W)}(\overline{V}|_W)$. Then the mapping $\nu : V \to V$ given by $\nu : v \mapsto v'$ where $(\overline{v}|_W)^{\mu|_W} = \overline{v'}|_W$, defines an automorphism of $V$.*

*Proof.* Assume that all maps are restricted to $W$. From assumption, given $v \in V$, $\overline{v}^{\,\mu} = \overline{v'}$ for some $v' \in V$, and by Lemma 4.10 $v'$ is uniquely determined by $v$. So $\nu$ is well defined and it is straightforward to check that $\nu$ is a homomorphism:

$$\overline{(v_1v_2)^\nu} \;= \mu^{-1}\overline{v_1v_2}\mu = \mu^{-1}\overline{v_1}\,\overline{v_2}\mu = \mu^{-1}\overline{v_1}\mu\,\mu^{-1}\overline{v_2}\mu \;= \overline{v_1^\nu}\,\overline{v_2^\nu} = \overline{v_1^\nu v_2^\nu}.$$

Further we note that since $\mu$ induces $\nu$ and $\mu$ is an automorphism, $\mu^{-1}$ exists, and induces a mapping $\nu^{-1} : V \to V$ which is both a right and left inverse for $\nu$. Thus $\nu$ is an automorphism. □

**Lemma 4.12.** *Let $\mu \in \mathrm{Aut}(U)$ be such that $\mu$ fixes $W$ and suppose that $\mu \sim_\Gamma \nu$ for some $\nu \in \mathrm{Aut}(V)$. Then $(\overline{v}|_W)^{\mu|_W} = \overline{v^\nu}|_W$ for all $v \in V$.*

*Proof.* Recall that $\mu \sim_\Gamma \nu$ implies that there exists $\alpha \in \Gamma$ with $\alpha|_U = \mu$ and $\alpha|_V = \nu$. So for all $w \in W$ and $v \in V$ we have $(w^v)^\alpha = (v^{-1})^\alpha w^\alpha v^\alpha$ and hence $(w^v)^\mu = (v^{-1})^\nu w^\mu v^\nu$. Thus if we apply the mapping $w \mapsto w^{\mu^{-1}}$ then $(v^{-1}w^{\mu^{-1}}v)^\mu = (v^{-1})^\nu w v^\nu$ and so $w^{\mu^{-1}\overline{v}\mu} = w^{\overline{v^\nu}}$ for all $w \in W$ and $v \in V$, and the result follows. □

**Lemma 4.13.** *Let $\alpha \in \Gamma$. If $\alpha|_W = \mathrm{Id}_W$ and $V^\alpha = V$ then $\alpha|_V = \mathrm{Id}_V$.*

*Proof.* For any $v \in V$, $w \in W$ we have

$$\begin{aligned} v^{-1}wv &= (v^{-1}wv)^\alpha \\ &= (v^{-1})^\alpha w^\alpha v^\alpha \\ &= (v^{-1})^\alpha w v^\alpha \end{aligned}$$

and therefore $v^\alpha v^{-1} \in C_V(W) = 1$, which gives $v^\alpha = v$. □

**Lemma 4.14.** *Let $\phi : \Gamma \to \mathrm{Aut}(U)$ be the mapping defined by $\phi : \alpha \mapsto \alpha|_U$ for $\alpha \in \Gamma$. Then $\phi$ is a monomorphism.*

*Proof.* Firstly note that $U^\alpha = U$ for $\alpha \in \Gamma$ by definition, and hence $\alpha^\phi \in \mathrm{Aut}(U)$ and $\phi$ is clearly a homomorphism. Let $\alpha \in \Gamma$ and suppose that $\alpha^\phi = \mathrm{Id}_U$. Then $\alpha$ centralises $U$ and fixes $V$, hence by Lemma 4.13, $\alpha = \mathrm{Id}_X$ and so $\phi$ is injective. □

### Semidirect product

Assume Hypothesis 4.9 with $U = W$. Then $X$ is a semidirect product $X = U \rtimes V$ and every $x \in X$ can be expressed uniquely as $x = uv$ for some $u \in U$ and $v \in V$, such that

$$u_1 v_1 u_2 v_2 = u_1 u_2^{v_1^{-1}} v_1 v_2, \tag{4.1}$$

for $u_1, u_2 \in U$ and $v_1, v_2 \in V$. We can construct $\mathrm{Aut}(X)$ directly from $\mathrm{Aut}(U)$ as follows.

**Theorem 4.15.** *Assume Hypothesis 4.9 with $U = W$ and define $\Delta = \{\mu \in \mathrm{Aut}(U) : \mu \in N_{\mathrm{Aut}(U)}(\overline{V}|_U)\}$. Then each $\delta \in \Delta$ can be extended to a unique automorphism of $X$ that fixes $V$.*

*Proof.* Uniqueness follows from Lemma 4.12. By Lemma 4.11, $\delta$ defines an automorphism $\nu : V \to V$ such that $\overline{v}^\mu|_W = \overline{v^\nu}|_W$ for $v \in V$. Thus $\mu\overline{v^\nu}|_W = \overline{v}\mu|_W$ and for $v_1^{-1} \in V$ and $u_2 \in U$ we have

$$(u_2^\mu)^{\overline{v_1^{-1\nu}}} = (u_2^{v_1^{-1}})^\mu$$

and therefore $v_1^\nu u_2^\mu = (u_2^{v_1^{-1}})^\mu v_1^\nu$. So for $u_1 \in U$ and $v_2 \in V$

$$\begin{aligned}(u_1^\mu v_1^\nu)(u_2^\mu v_2^\nu) &= u_1^\mu (u_2^{v_1^{-1}})^\mu v_1^\nu v_2^\nu \\ &= (u_1 u_2^{v_1^{-1}})^\mu (v_1 v_2)^\nu\end{aligned}$$

and hence using (4.1), $uv \mapsto u^\mu v^\nu$ is an endomorphism, and therefore an automorphism, of $X$. □

So, recall that $\Gamma = \{\alpha \in \mathrm{Aut}(X) : U^\alpha = U, V^\alpha = V\}$, and define

$$\Delta = \left\{\mu \in \mathrm{Aut}(U) : \mu \in N_{\mathrm{Aut}(U)}(\overline{V}|_U)\right\}. \tag{4.2}$$

**Proposition 4.16.** *There exists an isomorphism $\phi : \Gamma \to \Delta$.*

*Proof.* Take any $\alpha \in \Gamma$ and consider the action of $\overline{v}^\alpha$ on $X$ for some $v \in V$. For $x \in X$, we have

$$\begin{aligned} x^{\overline{v}^\alpha} &= (v^{-1} x^{\alpha^{-1}} v)^\alpha \\ &= (v^{-1})^\alpha x v^\alpha \\ &= x^{\overline{v^\alpha}}, \end{aligned}$$

and since $\alpha$ fixes $V$, $\overline{v^\alpha} \in \overline{V}$, and thus $\alpha|_U \in N_{\mathrm{Aut}(U)}(\overline{V}|_U)$. So define $\phi : \alpha \mapsto \alpha|_U$ and we have $\alpha^\phi \in \Delta$. Further, $\phi$ is monomorphism by Lemma 4.14 and by Theorem 4.15 each $\delta \in \Delta$ can be extended to a unique automorphism of $X$ that fixes $V$. So there exists a unique $\alpha \in \mathrm{Aut}(X)$ such that $\alpha|_U = \delta$, $U^\alpha = U$ and $V^\alpha = V$. Hence $\alpha \in \Gamma$, and $\phi$ is an isomorphism. □

### Conjugation search

If we now relax the condition of $U \operatorname{char} X$, then we can still construct $\mathrm{Aut}(X)$ using the same approach, with a few alterations. Firstly, we define $\Delta'$ as follows:

$$\Delta' = \left\{\delta \in \mathrm{Aut}(U) : W^\delta = W,\ N_U(V)^\delta = N_U(V),\ \delta|_W \in N_{\mathrm{Aut}(W)}(\overline{V}|_W)\right\},$$

and let $\Delta \leq \Delta'$ be the subgroup of $\Delta'$ whose elements extend to automorphisms of $X$ that fix $V$ (compare with (4.2)). Recall that $\Gamma = \{\alpha \in \mathrm{Aut}(X) : V^\alpha = V, U^\alpha = U\}$.

**Proposition 4.17.** *There exists an injective homomorphism $\phi : \Gamma \to \Delta'$, and* $\operatorname{Im}\phi = \Delta$.

*Proof.* Define $\phi : \alpha \mapsto \alpha|_U$ for $\alpha \in \Gamma$. We begin by checking that $\alpha^\phi \in \Delta$. Firstly $W^\alpha = W$ for all $\alpha \in \Gamma$ as $W \operatorname{char} X$ and hence $\alpha|_U$ fixes $W$. Also, $V = V^\alpha = (n^{-1}Vn)^\alpha = (n^{-1})^\alpha V n^\alpha$ for any $n \in N_U(V)$ so $\alpha$, and therefore $\alpha|_U$, fixes $N_U(V)$. Finally for $w \in W$, $w^{\alpha^{-1}\overline{v}\alpha} = w^{\overline{v^\alpha}}$ and $v^\alpha \in V$, hence the last condition holds. Now by Lemma 4.14, $\phi$ is injective, and since each $\alpha \in \Gamma$ fixes $U$ and $V$, $\alpha|_U \in \Delta$. Thus $\operatorname{Im}\phi \leq \Delta$. So suppose that $\delta \in \Delta$, and hence $\delta$ extends to an automorphism of $X$ fixing $V$. Therefore, there exists $\alpha \in \Gamma$ such that $\alpha|_U = \delta$, and $\delta \in \operatorname{Im}\phi$. So $\operatorname{Im}\phi = \Delta$. □

We now use Lemmas 4.11 and 4.12 and attempt to extend each $\delta \in \Delta'$ to an automorphism of $X$. Therefore in practice we determine $\Delta$ by performing a search for automorphisms which extend correctly.

However if $X = WN_X(V)$ and therefore $X = WN_U(V)V$, we can prove that $\Delta = \Delta'$ and no searching is required. Observe that in these cases we write the product of two elements $x_1 = w_1n_1v_1$ and $x_2 = w_2n_2v_2$ in $X$ as

$$(w_1n_1v_1)(w_2n_2v_2) = w_1 w_2^{v_1^{-1}n_1^{-1}} n_1 n_2 v_1^{n_2} v_2, \tag{4.3}$$

where $w_1, w_2 \in W$, $n_1, n_2 \in N_U(V)$ and $v_1, v_2 \in V$.

**Theorem 4.18.** *Assume Hypothesis 4.9 and suppose that $X = WN_U(V)V$. Define $\Delta' = \{\delta \in \operatorname{Aut}(U) : W^\delta = W,\ N_U(V)^\delta = N_U(V),\ \delta|_W \in N_{\operatorname{Aut}(W)}(\overline{V}|_W)\}$. Then each $\delta \in \Delta'$ can be extended to a unique automorphism of $X$ fixing $V$.*

*Proof.* We follow a similar argument to the proof of Theorem 4.15. Uniqueness follows from Lemma 4.12. By Lemma 4.11, $\delta$ defines an automorphism $\nu \in \operatorname{Aut}(V)$ such that $\overline{v}^\delta|_W = \overline{v^\nu}|_W$ and thus $\delta|_W \overline{v^\nu}|_W = \overline{v}|_W \delta|_W$ for $v \in V$. So for any $w_2 \in W$,

$v_1 \in V$ we have

$$(w_2^\delta)^{(v_1^{-1})^\nu} = (w_2^{v_1^{-1}})^\delta$$

and hence for $n_2 \in N_U(V)$ we obtain

$$\begin{aligned}
(w_2^\delta)^{(v_1^{-1})^\nu} &= (w_2^{v_1^{-1}})^\delta && \Longrightarrow \\
v_1^\nu w_2^\delta (v_1^\nu)^{-1} &= (w_2^{v_1^{-1}})^\delta && \Longrightarrow \\
v_1^\nu w_2^\delta &= (w_2^{v_1^{-1}})^\delta v_1^\nu && \Longrightarrow \\
v_1^\nu w_2^\delta n_2^\delta &= (w_2^{v_1^{-1}})^\delta n_2^\delta (v_1^\nu)^{n_2^\delta}.
\end{aligned}$$

Now we claim that

$$(v^\nu)^{n^\delta} = (v^n)^\nu \tag{4.4}$$

holds for all $n \in N_U(V)$, $v \in V$. It is sufficient to show that the action of the RHS and LHS on $W$ are the same since $C_V(W) = 1$. Note that $\overline{v^n} = \overline{v}^{\overline{n}}$ in our notation.

$$\begin{aligned}
\overline{(v^\nu)^{n^\delta}} &= \overline{v^\nu}^{\overline{n^\delta}} \\
&= \overline{n^\delta}^{-1} \delta^{-1} \overline{v} \delta \overline{n^\delta} \\
&= \overline{n^{-1\delta}} \delta^{-1} \overline{v} \delta \overline{n^\delta} \\
&= \delta^{-1} \overline{n^{-1}} \overline{vn} \delta \\
&= \overline{v}^{\overline{n}\delta}
\end{aligned}$$

$$\begin{aligned}
\overline{(v^n)^\nu} &= \overline{v^n}^\delta \\
&= \delta^{-1} \overline{v}^{\overline{n}} \delta \\
&= \overline{v}^{\overline{n}\delta} \\
&= \overline{(v^\nu)^{n^\delta}},
\end{aligned}$$

which proves the claim.

Hence, continuing our earlier calculation, we have

$$\begin{aligned} v_1^\nu w_2^\delta n_2^\delta &= (w_2^{v_1^{-1}})^\delta n_2^\delta (v_1^\nu)^{n_2^\delta} \\ &= (w_2^{v_1^{-1}})^\delta n_2^\delta (v_1^{n_2})^\nu. && \text{(by (4.4))} \end{aligned}$$

Furthermore, for $w_1 \in W$, $n_1 \in N_U(V)$ and $v_2 \in V$ we deduce

$$\begin{aligned} n_1^\delta v_1^\nu (w_2 n_2)^\delta &= n_1^\delta (w_2^{v_1^{-1}})^\delta n_2^\delta (v_1^{n_2})^\nu \\ &= (n_1 w_2^{v_1^{-1}} n_2)^\delta (v_1^{n_2})^\nu \\ &= (w_2^{v_1^{-1} n_1^{-1}} n_1 n_2)^\delta (v_1^{n_2})^\nu \end{aligned}$$

and hence

$$((w_1 n_1)^\delta v_1^\nu)((w_2 n_2)^\delta v_2^\nu) = (w_1 w_2^{v_1^{-1} n_1^{-1}} n_1 n_2)^\delta (v_1^{n_2} v_2)^\nu.$$

Using (4.3), $uv \mapsto u^\delta v^\nu$ defines an endomorphism, and hence an automorphism of $X$. □

## 4.2 Isomorphism testing

We now outline results for isomorphism testing, which mirror those for automorphism computation given in the previous section. Again, these results are written in very general terms and will be referred to later in the thesis.

Let $X_1$ and $X_2$ be two groups, and suppose that we want to determine if $X_1 \cong X_2$. Our general strategy is to assume that some basic structural compatibility has already been determined between $X_1$ and $X_2$ and we then use this to attempt to construct an isomorphism.

### 4.2.1 Direct product

Given two groups which are direct products of characteristic subgroups, we show that an isomorphism between the two can be constructed from isomorphisms between the corresponding factors of the direct products.

**Theorem 4.19.** *Let $X_1, X_2$ be groups with characteristic subgroups $U_i, V_i \leq X_i$ with $X_i = U_iV_i$ and $U_i \cap V_i = 1$ for $i = 1, 2$. Then $X_i \cong U_i \times V_i$ for $i = 1, 2$. Further suppose that $|U_1| = |U_2|$ and $|V_1| = |V_2|$. Then $X_1 \cong X_2$ if and only if there exist isomorphisms $\mu : U_1 \to U_2$ and $\nu : V_1 \to V_2$, and moreover $\Phi : X_1 \to X_2$ is an isomorphism defined by $\Phi(uv) = u^\mu v^\nu$ where $u \in U$ and $v \in V$.*

*Proof.* Let $\phi_i : X_i \to U_i \times V_i$ for $i = 1, 2$, be defined as $\phi_i : x_i \mapsto (u_i, v_i)$ where $x_i = u_iv_i$ for $x_i \in X_i$, $u_i \in U_i$ and $v_i \in V_i$. Suppose we have isomorphisms $\mu$ and $\nu$, then define $\Phi_\mu : U_1 \times V_1 \to U_2 \times V_1$ and $\Phi_\nu : U_1 \times V_1 \to U_1 \times V_2$ by $\Phi_\mu : (u, v) \mapsto (u^\mu, v)$ and $\Phi_\nu : (u, v) \mapsto (u, v^\nu)$ for $u \in U_1$, $v \in V_1$. Clearly both $\Phi_\mu, \Phi_\nu$ are isomorphisms, as is $\Phi_\mu\Phi_\nu : (u, v) \mapsto (u^\mu, v^\nu)$ and so $\phi_1 \circ \Phi_\mu\Phi_\nu \circ \phi_2^{-1} : X_1 \to X_2$ is an isomorphism.

Now suppose that $X_1 \cong X_2$ and $\Phi : X_1 \to X_2$ is an isomorphism. By Lemma 6.1, $U_1 \cong U_2$ and $V_1 \cong V_2$, and hence there exists isomorphisms $\mu : U_1 \to U_2$ and $\nu : V_1 \to V_2$. Since $U_2, V_2 \operatorname{char} X_2$, $U_1^\Phi = U_2$ and $V_1^\Phi = V_2$, thus $\mu = \Phi|_{U_1}$ and $\nu = \Phi|_{V_1}$. □

### 4.2.2 Subdirect product

In this section we assume that we have two groups $X_1$ and $X_2$ which can be written as subdirect products, and show that isomorphisms between the subdirect factors can be used to construct an isomorphism $X_1 \to X_2$.

**Theorem 4.20.** *Let $X_1$, $X_2$ be groups with subdirect products defined by $\psi_i : X_i \to X_i/U_i \times X_i/V_i$ where $U_i, V_i \operatorname{char} X_i$ and $V_i \cap U_i = 1$ for $i = 1, 2$. Then there exists an isomorphism $\Psi : X_1 \to X_2$ such that $U_1^\Psi = U_2$ and $V_1^\Psi = V_2$ if and only if* *there exist isomorphisms* $\Psi_U : X_1/U_1 \to X_2/U_2$ *and* $\Psi_V : X_1/V_1 \to X_2/V_2$ *such that* $X_1^{\psi_1(\Psi_U,\Psi_V)} = X_2^{\psi_2}$.

*Proof.* Assume that there exists an isomorphism $\Psi : X_1 \to X_2$, then we can define isomorphisms $\Psi_U : X_1/U_1 \to X_2/U_2$, $\Psi_V : X_1/V_1 \to X_2/V_2$ by $\Psi_U : xU_1 \mapsto x^{\Psi}U_2$ and $\Psi_V : xV_1 \mapsto x^{\Psi}V_2$ for $x \in X_1$.

It now remains to check that $X_1^{\psi_1(\Psi_U,\Psi_V)} = X_2^{\psi_2}$. Take $x \in X_1$, then we have

$$\begin{aligned} x^{\psi_1(\Psi_U,\Psi_V)} &= (xU_1, xV_1)^{(\Psi_U,\Psi_V)} \\ &= ((xU_1)^{\Psi_U}, (xV_1)^{\Psi_V}) \\ &= (x^{\Psi}U_2, x^{\Psi}V_2) \\ &= (x^{\Psi})^{\psi_2}, \end{aligned}$$

and since $\Psi$ is an isomorphism, we are done.

Now we prove the converse. We claim that the mapping $\Psi : X_1 \to X_2$ defined by $\Psi = \psi_1(\Psi_U, \Psi_V)\psi_2^{-1}$ is an isomorphism. First, given $x \in X_1$, $x^{\psi_1(\Psi_U,\Psi_V)} = (xU_1, xV_1)^{(\Psi_U,\Psi_V)} = (x_2U_2, x_2V_2)$ for some $x_2 \in X_2$ since $X_1^{\psi_1(\Psi_U,\Psi_V)} = X_2^{\psi_2}$. Thus as $\psi_2$ is a monomorphism, $\Psi$ is a well defined mapping, and since $\psi_1, \psi_2, \Psi_U$ and $\Psi_V$ are homomorphisms, $\Psi$ is a homomorphism.

Now suppose that $x^{\Psi} = 1_{X_2}$ for some $x \in X_1$. Then as $\psi_1$, $\psi_2$ are monomorphisms and $\Psi_U$, $\Psi_V$ are isomorphisms, we observe that $(1_{X_2})^{\psi_2} = 1_{X_2/U_2 \times X_2/V_2} = {1_{X_1/U_1 \times X_1/V_1}}^{(\Psi_U,\Psi_V)}$, and $1^{\psi_1} = 1_{X_1/U_1 \times X_1/V_1}$. Hence $x = 1_{X_1}$ and $\Psi$ is a monomorphism. Finally, $\Psi$ is surjective since $X_1^{\psi_1(\Psi_U,\Psi_V)} = X_2^{\psi_2}$, and hence $\Psi$ is an isomorphism, which proves the claim. Thus we have constructed an isomorphism $\Psi : X_1 \to X_2$. □

**Theorem 4.21.** *Let* $X_1, X_2$ *be groups with characteristic subgroups* $U_i, V_i \leq X_i$ *with* $U_i \cap V_i = 1$ *for* $i = 1, 2$. *Define* $\phi_{i,U} : X_i \to X_i/U_i$ *and* $\phi_{i,V} : X_i \to X_i/V_i$ *to be the natural maps. Then there exists an isomorphism* $\Psi : X_1 \to X_2$ *such that* $U_1^{\Psi} = U_2$

*and $V_1^\Psi = V_2$ if and only if there exist isomorphisms $\Psi_U : X_1/U_1 \to X_2/U_2$ and $\Psi_V : X_1/V_1 \to X_2/V_2$ such that*

$$\left((x_1U_1V_1/U_1)^{\Psi_U}\right)^{\phi_{2,U}^{-1}} = \left((x_1U_1V_1/V_1)^{\Psi_V}\right)^{\phi_{2,V}^{-1}} \tag{4.5}$$

*for every $x_1 \in X_1$.*

*Proof.* Suppose that $\Psi : X_1 \to X_2$ is an isomorphism such that $U_1^\Psi = U_2$ and $V_1^\Psi = V_2$. Then by Theorem 4.20, there exist isomorphisms $\Psi_U : X_1/U_1 \to X_2/U_2$ and $\Psi_V : X_1/V_1 \to X_2/V_2$ defined by $\Psi_U : xU_1 \mapsto x^\Psi U_2$ and $\Psi_V : xV_1 \mapsto x^\Psi V_2$ respectively, such that $X_1^{\psi_1(\Psi_U,\Psi_V)} = X_2^{\psi_2}$ where $\psi_i : X_i \to X_i/U_i \times X_i/V_i$ for $i = 1, 2$. In particular, if $x \in X_1$ and $(xU_1)^{\Psi_U} = yU_2$ for some $y \in X_2$, then $(xV_1)^{\Psi_V} = yV_2$. Observe that $(V_1U_1)^{\Psi_U} = V_1^\Psi U_2 = V_2U_2$ since $V_1^\Psi = V_2$ and similarly $(U_1V_1)^{\Psi_V} = U_1^\Psi V_2 = U_2V_2$ due to $U_1^\Psi = U_2$. Now, given $x \in X_1$ suppose that $(xU_1V_1)^{\Psi_U} = yU_2V_2$ with $y \in X_2$. Then $(xU_1V_1)^{\Psi_V} = yU_2V_2$, and so $\Psi_U$ and $\Psi_V$ satisfy (4.5).

Conversely, suppose that $\Psi_U : X_1/U_1 \to X_2/U_2$ and $\Psi_V : X_1/V_1 \to X_2/V_2$ are isomorphisms such that (4.5) holds. Let $x \in X_1$ and suppose that $(xU_1)^{\Psi_U} = aU_2$ and $(xV_1)^{\Psi_V} = bV_2$ where $a, b \in X_2$. Since $V_1^{\Psi_U} = V_2$ and $U_1^{\Psi_V} = U_2$, we get $(xU_1V_1)^{\Psi_U} = aU_2V_2$ and $(xU_1V_1)^{\Psi_V} = bU_2V_2$. By (4.5), we must have $aU_2V_2 = bU_2V_2$ and hence $a \in bU_2V_2$. Therefore there exist $u_2 \in U_2$ and $v_2 \in V_2$ such that $a = bu_2v_2$, and $(xU_1, xV_2)^{(\Psi_U,\Psi_V)} = (bv_2U_2, bV_2) = (bv_2U_2, bv_2V_2)$. Therefore $X_1^{\psi_1(\Psi_U,\Psi_V)} = X_2^{\psi_2}$, and by Theorem 4.20 $X_1 \cong X_2$. □

### 4.2.3 Conjugation action

In this section we outline results for isomorphism testing which are analogous to those for constructing automorphism groups found in Section 5.3.

**Definition 4.22.** Let $X_1$ and $X_2$ be groups and suppose that $U_i, V_i \leq X_i$ for $i = 1, 2$.

Then isomorphisms $\mu : U_1 \to U_2$ and $\nu : V_1 \to V_2$ are said to be *associated* if there exists an isomorphism $\Phi : X_1 \to X_2$ such that $\Phi|_{U_1} = \mu$ and $\Phi|_{V_1} = \nu$, and we write $\mu \sim_{X_1,X_2} \nu$. Further, we say that $\mu$ *extends*, or *can be extended* to an isomorphism $X_1 \to X_2$.

In particular, we note the following consequence of this definition.

**Lemma 4.23.** *Let $X_1$ and $X_2$ be groups with subgroups $U_i, V_i \leq X_i$ such that $X_i = U_iV_i$ for $i = 1, 2$. Suppose that $\mu : U_1 \to U_2$ and $\nu : V_1 \to V_2$ are isomorphisms. Then $\mu \sim \nu$ if and only if $uv \mapsto u^\mu v^\nu$ is an isomorphism $X_1 \to X_2$ for $u \in U_1, v \in V_1$.*

*Proof.* Every element $x_i \in X_i$ can be expressed as a product $x_i = u_iv_i$ for $u_i \in U_i$, $v_i \in V_i$, where $i = 1, 2$. Hence by definition, there exists an isomorphism $\Phi : X_1 \to X_2$ with $\Phi|_{U_1} = \mu$ and $\Phi|_{V_2} = \nu$ if and only if $\Phi : x_1 \mapsto u_1^\Phi v_1^\Phi = u_1^\mu v_1^\nu$ for $u_1 \in U_1, u_2 \in U_2$, is an isomorphism. □

We now introduce a hypothesis which will be assumed throughout this section.

**Hypothesis 4.24.** Let $X_1$ and $X_2$ be finite groups with subgroups $U_i, V_i, W_i \leq X_i$ such that $X_i = U_iV_i$, $U_i \cap V_i = 1$, $W_i \leq U_i$, $W_i \operatorname{char} X_i$ and $C_{V_i}(W_i) = 1$ for $i = 1, 2$.

Using Definition 4.22 we can derive conditions for an isomorphism $\mu : U_1 \to U_2$ to be associated with an isomorphism $\nu : V_1 \to V_2$, and further, how to construct such a $\nu$ given $\mu$.

**Lemma 4.25.** *Let $\mu : U_1 \to U_2$ be an isomorphism such that $W_1^\mu = W_2$ and $(\overline{v_1}|_{W_1})^{\mu|_{W_1}} \in \overline{V_2}|_{W_2}$ for every $v_1 \in V_1$. Then the mapping $\nu : V_1 \to V_2$, given by $\nu : v_1 \mapsto v_2$ where $(\overline{v_1}|_{W_1})^{\mu|_{W_1}} = \overline{v_2}|_{W_2}$ for some $v_2 \in V_2$, is a monomorphism.*

*Proof.* Assume that all maps are restricted to $W_1$ or $W_2$ where appropriate. We have $\overline{v_1}^\mu = \overline{v_2}$ where $v_1 \in V_1$, $v_2 \in V_2$ and this $v_2$ is unique since $C_{V_2}(W_2) = 1$ (recall that by Lemma 4.10, the mapping $v_1 \mapsto \overline{v_1}|_{W_1}$ is a monomorphism). Hence $\nu$ is well defined. It is straightforward to check that the mapping is a homomorphism. For $v_1, v_2 \in V_1$ we have:

$$\overline{(v_1v_2)^\nu} \;= \mu^{-1}\overline{v_1v_2}\mu = \mu^{-1}\overline{v_1}\,\overline{v_2}\mu = \mu^{-1}\overline{v_1}\mu\,\mu^{-1}\overline{v_2}\mu \;= \overline{v_1^\nu}\,\overline{v_2^\nu} = \overline{v_1^\nu v_2^\nu}.$$

Further suppose that $v_1^\nu = 1_{V_2}$ for some $v_1 \in V_1$. Then $\overline{v_1}^\mu = \mathrm{Id}_{W_2}$ and thus $((w_2^{\mu^{-1}})^{\overline{v_1}})^\mu = w_2$ for every $w_2 \in W_2$. Hence $(w_2^{\mu^{-1}})^{\overline{v_1}} = w_2^{\mu^{-1}}$ for all $w_2 \in W_2$ and since $W_1^\mu = W_2$, $w_1^{\overline{v_1}} = w_1$ for all $w_1 \in W_1$, so $v_1 = 1$ as $C_{V_1}(W_1) = 1$. Therefore $\nu$ is a monomorphism. □

**Lemma 4.26.** *Let $\mu : U_1 \to U_2$ be an isomorphism such that $W_1^\mu = W_2$ and let $\nu : V_1 \to V_2$ be an isomorphism. If $\mu \sim_{X_1,X_2} \nu$ then $(\overline{v_1}|_{W_1})^{\mu|_{W_1}} = \overline{v_1^\nu}|_{W_2}$ for every $v_1 \in V_1$.*

*Proof.* Recall that $\mu \sim_{X_1,X_2} \nu$ implies that there exists an isomorphism $\Phi : X_1 \to X_2$ with $\Phi|_{U_1} = \mu$ and $\Phi|_{V_1} = \nu$. So, for all $w_1 \in W_1$ and $v_1 \in V_1$, $(w_1^{v_1})^\Phi = (v_1^{-1})^\Phi k^\Phi v_1^\Phi$. Therefore $(w_1^{\overline{v_1}})^\mu = (v_1^{-1})^\nu w_1^\mu v_1^\nu$, and hence $w_1^{\overline{v_1}} = w_1^{\mu\overline{v_1^\nu}\mu^{-1}}$. Let $w_2 = w_1^\mu \in W_2$, then $w_2^{\mu^{-1}\overline{v_1}\mu} = w_2^{\overline{v_1}^\mu} = w_2^{\overline{v_1^\nu}}$. □

### Semidirect product

In this section assume Hypothesis 4.24 and further suppose that $U_i = W_i$ for $i = 1, 2$. Thus, as in the equivalent case for automorphism groups, we have $X_i = U_i \rtimes V_i$ and every $x_i \in X_i$ can be expressed uniquely as $x_i = u_iv_i$ for some $u_i \in U_i$ and $v_i \in V_i$ for $i = 1, 2$. Furthermore we have

$$u_1v_1u_2v_2 = u_1u_2^{v_1^{-1}}v_1v_2, \tag{4.6}$$

for $u_1, u_2 \in U_1$ and $v_1, v_2 \in V_1$ (and the same holds for $u_1, u_2 \in U_2$ and $v_1, v_2 \in V_2$).

**Theorem 4.27.** *Assume Hypothesis 4.24 with $U_i = W_i$ for $i = 1, 2$. Let $\mu : U_1 \to U_2$ be an isomorphism such that $(\overline{V_1}|_{U_1})^\mu = \overline{V_2}|_{U_2}$. Then $\mu$ can be extended to an* *isomorphism $X_1 \to X_2$.*

*Proof.* Define $\nu : V_1 \to V_2$ by $v_1 \mapsto v_2$ for $v_1 \in V_1$ and $v_2 \in V_2$ where $\overline{v_1}^{\mu} = \overline{v_2}$. By Lemma 4.25, $\nu$ is a monomorphism, and hence an isomorphism (since $(\overline{V_1}|_{U_1})^{\mu} = \overline{V_2}|_{U_2}$). We claim that $\mu$ and $\nu$ combine to form an isomorphism; i.e. that $uv \mapsto u^{\mu}v^{\nu}$ is an isomorphism $X_1 \to X_2$ (Lemma 4.23). By the definition of $\nu$, for $u_2' \in U_2$ and $v_1 \in V_1$ we have

$$(u_2')^{\mu^{-1}\overline{v_1^{-1}}\mu} = (u_2')^{\overline{v_1^{-1\nu}}},$$

and if we now define $u_2 \in U_1$ such that $u_2^{\mu} = u_2'$ then we obtain

$$u_2^{\overline{v_1^{-1}}\mu} = u_2^{\mu\overline{v_1^{-1\nu}}}.$$

Hence $u_2^{\overline{v_1^{-1}}\mu}v_1^{\nu} = v_1^{\nu}u_2^{\mu}$ and for $u_1 \in U_1$, $v_2 \in V_1$ we have

$$u_1^{\mu}(u_2^{\overline{v_1^{-1}}})^{\mu}v_1^{\nu}v_2^{\nu} = (u_1^{\mu}v_1^{\nu})(u_2^{\mu}v_2^{\nu}).$$

Therefore using (4.6) in Section 5.3.1, $uv \mapsto u^{\mu}v^{\nu}$ is an isomorphism $X_1 \to X_2$. □

### General method

Again if we assume Hypothesis 4.24 but now relax the condition of $U_i \operatorname{char} X_i$ for $i = 1, 2$ given in the previous section, we can still use a similar approach to the semidirect case, with several refinements.

As in the automorphism group case, if $X_i = W_i N_{U_i}(V_i)V_i$ for $i = 1, 2$, then we can give specific criteria necessary for an isomorphism $\mu : U_1 \to U_2$ to extend to an isomorphism $\Phi : X_1 \to X_2$, as demonstrated in the following theorem.

**Theorem 4.28.** *Assume Hypothesis 4.24 and suppose that $X_i = W_i N_{U_i}(V_i)V_i$ for $i = 1, 2$. Let $\mu : U_1 \to U_2$ be an isomorphism such that $W_1^{\mu} = W_2$, $N_{U_1}(V_1)^{\mu} = N_{U_2}(V_2)$ and $(\overline{V_1}|_{W_1})^{\mu|_{W_1}} = \overline{V_2}|_{W_2}$. Then $\mu$ extends to an isomorphism $X_1 \to X_2$.*

*Proof.* Using Lemma 4.25 and Lemma 4.26, $\mu$ defines an isomorphism $\nu : V_1 \to V_2$ such that $(\overline{v}|_{W_1})^{\mu|_{W_1}} = \overline{v^\nu}|_{W_2}$, and hence

$$\overline{v}|_{W_1}\mu|_{W_1} = \mu|_{W_1}\overline{v^\nu}|_{W_2} \tag{4.7}$$

for $v \in V_1$.

Now we claim that

$$(v^\nu)^{n^\mu} = (v^n)^\nu \tag{4.8}$$

holds for all $n \in N_{U_1}(V_1)$, $v \in V_1$. It is sufficient to show that the action of the RHS and LHS on $W_2$ are the same since $C_{V_2}(W_2) = 1$. As in the proof of Theorem 4.18, note that $\overline{v^n} = \overline{v}^{\overline{n}}$, and to further simplify notation we write $\overline{v}|_{W_1}$ as $\overline{v}$.

$$\begin{aligned}
\overline{(v^\nu)^{n^\mu}} &= \overline{v^\nu}^{\overline{n^\mu}} \\
&= \overline{n^\mu}^{-1}\mu^{-1}\overline{v}\mu\overline{n^\mu} \\
&= \overline{n^{-1\mu}}\mu^{-1}\overline{v}\mu\overline{n^\mu} \\
&= \mu^{-1}\overline{n^{-1}}\overline{vn}\mu \\
&= \overline{v}^{\overline{n}\mu}
\end{aligned}$$

$$\begin{aligned}
\overline{(v^n)^\nu} &= \overline{v^n}^\mu \\
&= \mu^{-1}\overline{v}^{\overline{n}}\mu \\
&= \overline{v}^{\overline{n}\mu} \\
&= \overline{(v^\nu)^{n^\mu}},
\end{aligned}$$

which proves the claim.

We note that, for any $w_2 \in W_1$, $v_1 \in V_1$ we have

$$(w_2^\mu)^{(v_1^{-1})^\nu} = (w_2^{v_1^{-1}})^\mu$$

by (4.7) and hence we obtain

$$\begin{aligned}
(w_2^{\mu})^{(v_1^{-1})^{\nu}} &= (w_2^{v_1^{-1}})^{\mu} && \Longrightarrow \\
v_1^{\nu} w_2^{\mu} (v_1^{\nu})^{-1} &= (w_2^{v_1^{-1}})^{\mu} && \Longrightarrow \\
v_1^{\nu} w_2^{\mu} &= (w_2^{v_1^{-1}})^{\mu} v_1^{\nu} && \Longrightarrow \\
v_1^{\nu} w_2^{\mu} n_2^{\mu} &= (w_2^{v_1^{-1}})^{\mu} n_2^{\mu} (v_1^{\nu})^{n_2^{\mu}} \\
&= (w_2^{v_1^{-1}})^{\mu} n_2^{\mu} (v_1^{n_2})^{\nu} && \text{(by (4.8))}
\end{aligned}$$

for $n_2 \in N_{U_1}(V_1)$. Furthermore, for $w_1 \in W_1$, $n_1 \in N_{U_1}(V_1)$ and $v_2 \in V_1$ we deduce

$$\begin{aligned}
n_1^{\mu} v_1^{\nu} (w_2 n_2)^{\mu} &= n_1^{\mu} (w_2^{v_1^{-1}})^{\mu} n_2^{\mu} (v_1^{n_2})^{\nu} \\
&= (n_1 w_2^{v_1^{-1}} n_2)^{\mu} (v_1^{n_2})^{\nu} \\
&= (w_2^{v_1^{-1} n_1^{-1}} n_1 n_2)^{\mu} (v_1^{n_2})^{\nu}
\end{aligned}$$

and hence

$$((w_1 n_1)^{\mu} v_1^{\nu})((w_2 n_2)^{\mu} v_2^{\nu}) = (w_1 w_2^{v_1^{-1} n_1^{-1}} n_1 n_2)^{\mu} (v_1^{n_2} v_2)^{\nu}.$$

Using (4.3) in Section 5.3.2, $uv \mapsto u^{\mu} v^{\nu}$ defines an isomorphism $X_1 \to X_2$. □

# Chapter 5

# Computing automorphism groups of soluble groups

In this chapter we derive a procedure to construct the automorphism group of a general soluble group $G$, and further, an algorithm which constructs a polycyclic presentation for $\mathrm{Aut}(G)$ when soluble. Most of the results refer directly to those given in Section 4.1.

We will often avoid stating explicit complexity evaluations of the algorithms in this chapter, as most computations rely on the automorphism group for $p$-group algorithm, which is known to be at least exponential. The goal is to find algorithms which perform well in practice, and to this end we include timing results for many examples in Chapter 8.

We begin by defining some notation which will be assumed as standard for the rest of the chapter.

**Notation (†)**

Let $G$ be a finite soluble group with $|G| = p^n q$ where $p \in \pi(G)$ and $q, n \in \mathbb{N}$ such that $\mathrm{hcf}(p, q) = 1$. Then by the results of Sylow and Hall detailed in Chapter 2, there exist subgroups $K, S, T \leq G$ such that $K = O_p(G)$, $S \in \mathrm{Syl}_p(G)$ and $T \in \mathrm{Hall}_q(G)$. We assume that $p$ has been chosen such that $K \neq 1$.

This construction gives rise to a product structure for $G$ which will be used for the rest of the chapter, thus:

**Lemma 5.1.** *$G = ST$.*

*Proof.* We have $S \cap T = 1$ and $|S||T| = |G|$. □

**Lemma 5.2.** *Let $g \in G$. Then there exists some $s_g \in S$ such that $g^{-1}Tg = s_g^{-1}Ts_g$. Similarly there exists some $t_g \in T$ such that $g^{-1}Sg = t_g^{-1}St_g$.*

*Proof.* Since $G = ST$ (by Lemma 5.1) we can find $s \in S$ and $t \in T$ such that $g^{-1} = st$. So if we take $s_g = s^{-1}$, then $s_g g^{-1} = s_g(st) = t \in N_G(T)$, and hence $g^{-1}Tg = s_g^{-1}Ts_g$. We repeat the argument exchanging $S$ and $T$ giving the equivalent result for conjugates of $S$. □

We now outline several results which allow us to split the algorithm into different cases.

**Definition 5.3.** Given a group $G$ we define the Fitting subgroup to be the largest normal nilpotent subgroup of $G$, denoted $F(G)$.

**Proposition 5.4.** *Let $G$ be a soluble group. Then $C_G(F(G)) \leq F(G)$.*

*Proof.* See [KS04, 6.1.4]. □

**Proposition 5.5.** *Suppose that $C_T(K) \lhd G$. Then $C_T(K) = O_{p'}(G)$ and hence $C_T(K) \operatorname{char} G$.*

*Proof.* We have $C_T(K) \leq O_{p'}(G)$ since $C_T(K)$ is normal in $G$, and $O_{p'}(G) \leq T$ (recall that $O_{p'}(G) = \bigcap_{g \in G} g^{-1}Tg$). Then for all $t \in O_{p'}(G)$ and $k \in K$

$$k^{-1}t^{-1}kt = (k^{-1}t^{-1}k)t = k^{-1}(t^{-1}kt) \in O_{p'}(G) \cap K = 1,$$

which gives $k^t = k$ and so $t \in C_T(K)$. Therefore $O_{p'}(G) \leq C_T(K)$. □

**Proposition 5.6.** *Suppose that* $C_T(K) \neq 1$. *Then* $O_{p'}(G) \neq 1$.

*Proof.* Firstly, $C_T(K) \neq 1$ and so $K \lneq KC_G(K) \lhd G$. Define $N/K$ to be a minimal normal subgroup of $G/K$ contained in $KC_G(K)/K$. Since $G$ is soluble, this is an elementary abelian $r$-group for some prime $r \neq p$ (recall $K = O_p(G)$). By definition $K \leq N \leq KC_G(K)$, so $N = KC_N(K)$ and hence $N = KL$ for $L \in \mathrm{Syl}_r(C_N(K))$. By Proposition 2.3, $N = K \times L$ and $L \operatorname{char} N \lhd G$. Therefore $L$ is normal in $G$ and $1 \neq L \leq O_{p'}(G)$. □

We are now in a position to give a full outline of the automorphism group algorithm, which proceeds by splitting into the following four cases:

1. **Direct Product**

   $S, T \lhd G$.

2. **Subdirect Product**

   $L = O_{p'}(G) \neq 1$. Here we construct a subdirect product $\psi : G \to G/K \times G/L$ and recurse.

3. **Semidirect Product**

   $S \lhd G$ and $T \ntriangleleft G$, giving $G \cong S \rtimes T$ (assume that $C_T(S) = 1$, otherwise in case 2).

4. **General Method (Conjugation Search)** $S \ntriangleleft G$.

The next few sections will address each case in detail. As we will sometimes need to recurse back to the main algorithm, we define a prototype [for the naming and parameters here], and then give a full description later in the chapter. The algorithm AutomorphismGroupSolubleGroup takes a parameter $G$ which is a finite soluble group, and constructs the automorphism group of $G$.

We begin by recalling the idea of extending automorphisms given in Definition 4.7. In particular, we recall the result of Lemma 4.8: an automorphism $\sigma \in \operatorname{Aut}(S)$ extends to an automorphism of $G$ fixing $T$ if and only if there exists an automorphism $\tau \in \operatorname{Aut}(T)$ such that $st \mapsto s^\sigma t^\tau$ is an automorphism of $G$.

## 5.1 Direct product

Assume that both $S$ and $T$ are normal (and therefore characteristic) subgroups of $G$. Then $G$ is a direct product, $G \cong S \times T$, and we construct the automorphism group of $G$ using Theorem 4.3. All automorphisms of $S$ can be extended to automorphisms of $G$ fixing $T$ (as we would expect since $T \operatorname{char} G$). A presentation for $G$ can be constructed from presentations of $S$ and $T$: for more details see [Joh97, 4.3, Proposition 4]. So, given polycyclic presentations $\operatorname{Pc}\langle s_1, s_2, \ldots, s_n | R_S\rangle$ and $\operatorname{Pc}\langle t_1, t_2, \ldots, t_m | R_T\rangle$ for $S$ and $T$ respectively, we can construct a polycyclic presentation

$$\operatorname{Pc}\langle s_1, s_2, \ldots, s_n, t_1, t_2, \ldots, t_m | R_S, R_T, [S, T]\rangle$$

for $G$. To define an automorphism $\alpha : G \to G$, we only need to specify the action of $\alpha$ on this generating set of $G$. The full algorithm AutomorphismGroupDirectProduct is given in Algorithm 5.1.

**Algorithm 5.1:** AutomorphismGroupDirectProduct

```
Input: $(G, S, T)$ : A finite soluble group $G$ with subgroups $S \in \mathrm{Syl}_p(G)$ and
        $T \in \mathrm{Hall}_{p'}(G)$ such that $S, T \lhd G$
Output: $A$ : The automorphism group of $G$
$A_S$ := AutomorphismGroupPGroup($S$);
$A_T$ := AutomorphismGroupSolubleGroup($T$);
$A_G$ := [ ];
for $\alpha_S \in$ Generators($A_S$) do
    Append(~ $A_G$, [$s \mapsto s^{\alpha_S}$ : $s \in$ PCGenerators(S)] $\cup$ [$t \mapsto t$ : $t \in$
    PCGenerators(T)]);
for $\alpha_T \in$ Generators($A_T$) do
    Append(~ $A_G$, [$s \mapsto s$ : $s \in$ PCGenerators(S)] $\cup$ [$t \mapsto t^{\alpha_T}$ : $t \in$
    PCGenerators(T)]));
return $\langle \alpha : \alpha \in A_G \rangle$;
```

## 5.2 Subdirect product

In this section we define $L = O_{p'}(G)$ and assume that $L \neq 1$. Since both $K$ and $L$ are characteristic subgroups of $G$ and $L \cap K = 1$, we construct a subdirect product $\psi : G \to G/K \times G/L$ given by $\psi : g \mapsto (gK, gL)$ and apply Theorems 4.5 and 4.6 to compute $\mathrm{Aut}(G)$. Determining $\mathrm{Aut}(G)$ is now reduced to two sub-problems: computing $\mathrm{Aut}(G/K)$ and $\mathrm{Aut}(G/L)$, and FixSubgroup calculations. Note that if $L = T$ then $G/L$ is a $p$-group and so again we are relying on the $p$-group automorphism algorithm.

We now give an outline of the formal procedure to compute $\mathrm{Aut}(G)$ in this case, AutomorphismGroupSubdirectProduct (see Algorithm 5.2).

**Algorithm 5.2:** AutomorphismGroupSubdirectProduct

```
Input: $(G, K, L)$ - $G$ soluble group, $K = O_p(G)$, $L = O_{p'}(G)$
Output: $A$ : The automorphism group of $G$
Let $\phi_K : G \to G/K$ and $\phi_L : G \to G/L$ be the natural maps;
/* Construct $\mathrm{Aut}(G/K)$ and $\mathrm{Aut}(G/L)$ */
$A_L :=$ AutomorphismGroupSolubleGroup($G^{\phi_L}$);
$A_K :=$ AutomorphismGroupSolubleGroup($G^{\phi_K}$);
$A_{L,K} :=$ FixSubgroup($A_L, K^{\phi_L}$);
$A_{K,L} :=$ FixSubgroup($A_K, L^{\phi_K}$);
$R_L, \varphi_L :=$ Representation($A_{L,K}$);
$R_K, \varphi_K :=$ Representation($A_{K,L}$);
Let $\phi_{K,LK} : G/K \to G/LK$ and $\phi_{L,LK} : G/L \to G/LK$ be the natural maps;
$A_{LK} :=$ AutomorphismGroup($(G^{\phi_K})^{\phi_{K,LK}}$);
$R_{LK}, \varphi_{LK} :=$ Representation($A_{LK}$);
Construct maps $\varphi_{L,LK} : R_L \to R_{LK}$ and $\varphi_{K,LK} : R_K \to R_{LK}$;
$A'_{gens} := [(k, 1_{R_L}) : k \in$ Generators($\mathrm{Ker}\,\varphi_{L,LK}$)$]$ cat
    $[(1_{R_K}, k) : k \in$ Generators($\mathrm{Ker}\,\varphi_{K,LK}$)$]$ cat
    $\left[\left((r_{KL})^{\varphi^{-1}_{K,LK}}, (r_{KL})^{\varphi^{-1}_{L,LK}}\right) : r_{LK} \in$ Generators($\mathrm{Im}\,\varphi_{K,LK} \cap \mathrm{Im}\,\varphi_{L,LK}$)$\right]$;
$A' := \langle\, (k^{\varphi_K^{-1}}, l^{\varphi_L^{-1}}) : (k, l) \in A'_{gens} \,\rangle$;
$A_G := [\,]$;
Define $\psi : G \to G/K \times G/L$ to be the natural mapping;
for $(\Psi_K, \Psi_L) \in$ Generators($A'$) do
    Append($\sim A_G, [g \mapsto g^{\psi(\Psi_K, \Psi_L)\psi^{-1}} : g \in$ PCGenerators($G$)$]$);
return $\langle \alpha : \alpha \in A_G \rangle$;
```

## 5.3 Conjugation action

In this section we handle the two remaining cases for the soluble group automorphism group algorithm: where $T$ is not normal in $G$. Note that we assume that $C_T(K) = 1$ otherwise we use the subdirect case (Proposition 5.6).

**Theorem 5.7.** *Define* $\Gamma = \{\alpha \in \mathrm{Aut}(G) : T^\alpha = T, S^\alpha = S\}$ *and let* $I = \mathrm{Inn}(G)$. *Then* $\mathrm{Aut}(G) = \Gamma I$*, and thus* $\mathrm{Aut}(G)$ *is determined by restriction to the action on* $S$ *and* $T$*.*

*Proof.* Given any automorphism $\alpha \in \mathrm{Aut}(G)$ we know that $S^\alpha = g^{-1}Sg$ for some $g \in G$ (by Lemma 2.17). Using Lemma 5.2 we can find $s \in S$ such that $T^{\alpha\overline{g^{-1}}} = s^{-1}Ts$, so $\alpha\overline{g^{-1}s^{-1}} \in \Gamma$. □

The strategy to compute the automorphism group in these cases is based on the results outlined in Section 4.1.3.

### 5.3.1 Semidirect product

Assume that $S \lhd G$ and $C_T(K) = 1$. Then $S$ is a characteristic subgroup of $G$ and by Theorem 5.7 we can use the results from the subsection “Semidirect Product” in Section 4.1.3. Thus to construct $\mathrm{Aut}(G)$ we need to compute

$$\Sigma = \{\sigma \in \mathrm{Aut}(S) : \sigma \in N_{\mathrm{Aut}(S)}(\overline{T}|_S)\}, \tag{5.1}$$

(as defined in (4.2)), and define a mapping $\phi : \Gamma \to \Sigma$ as $\phi : \alpha \mapsto \alpha|_S$. By Proposition 4.16, $\phi$ is an isomorphism and we have $\Sigma \cong \Gamma$. So we construct $\Gamma$ by extending the generators of $\Sigma$ to automorphisms of $G$ (note that given any $\sigma \in \Sigma$, we can construct an automorphism $T \to T$ using Lemma 4.11).

The formal procedure to compute $\mathrm{Aut}(G)$ for the semidirect product case is given in AutomorphismGroupSemidirectProduct (see Algorithm 5.3).

**Algorithm 5.3:** AutomorphismGroupSemidirectProduct

```
Input: $(G, S, T)$ : A soluble group $G$ with $S = O_p(G)$, $T \in \mathrm{Hall}_{p'}(G)$ and
       $C_T(S) = 1$
Output: $A$ : The automorphism group of $G$
$A_S :=$ AutomorphismGroupPGroup($S$);
$\phi, R :=$ Representation($A_S$);                    /* See Section 3.1 */
$T :=$ HallSubgroup($G, p'$);
$\Sigma :=$ Normaliser($R, \langle\, (\bar{t}|_S)^{\phi} : t \in$ PCGenerators($T$) $\rangle$);
$A_G :=$ [ ];
/* Construct $\Gamma$ by extending generators of $\Sigma$ */
for $\sigma \in$ Generators($\Sigma$) do
    $\alpha_S := [s \mapsto s^{\sigma^{\phi^{-1}}} : s \in$ PCGenerators($S$)];
    /* Find an element $t \in T$ such that $S^{\bar{t}} = S^{\sigma}$ */
    $\alpha_T := [t \mapsto$ FindConjugatingElement($T, S, \bar{t}^{\sigma}$) $: t \in$
    PCGenerators($T$)];
    Append($\sim A_G, \alpha_S \cup \alpha_T$);
/* Construct $\mathrm{Inn}(G)$ */
for $g \in$ Generators($G$) do
    Append($\sim A_G$, InnerAutomorphism($G, g$));
return $\langle \alpha : \alpha \in A_G \rangle$;
```

### 5.3.2 Conjugation search

Now we consider the case where $S \neq K$ and $O_{p'}(G) = 1$. Again using Theorem 5.7 and the results of Section 4.1.3, we want to find some subgroup of $\mathrm{Aut}(S)$ which can be extended to the automorphism group of $G$. Let $\Sigma' \leq \mathrm{Aut}(S)$ be defined as follows:

$$\Sigma' = \{\sigma \in \mathrm{Aut}(S) : K^\sigma = K,\, N_S(T)^\sigma = N_S(T),\, \sigma|_K \in N_{\mathrm{Aut}(K)}(\,\overline{T}|_K\,)\}.$$

Define $\Sigma \leq \Sigma'$ to be the subgroup of $\Sigma'$ whose elements extend to automorphisms of $G$ (again compare with (4.2) and (5.1)).

We now use Lemmas 4.11 and 4.12 from Section 4.1.3 and attempt to extend each $\sigma \in \Sigma'$ to an automorphism of $G$. In practice, this involves a search for automorphisms which extend correctly. We begin by testing the generators of $\Sigma'$, holding those which do extend in a subgroup $\Sigma_0 \leq \Sigma'$. We then construct double cosets $\{\Sigma_0 \alpha \Sigma_0 : \alpha \in \Sigma'\}$, and test each double coset representative. Those that extend are adjoined to $\Sigma_0$, and we recompute the double coset representatives. This continues until all representatives have been tested.

However, if $G = KN_G(T)$ and therefore $G = KN_S(T)T$, then $\Sigma = \Sigma'$ (by Theorem 4.18) and no searching is required. There are several examples where this construction occurs, so we devote the next few results to demonstrating this.

**Proposition 5.8.** *Let $G$ be a finite soluble group such that $O_p(G) = 1$ for some $p \in \pi(G)$, and suppose that $G$ has a cyclic Sylow $q$-subgroup for all primes $q \in \pi(G) \setminus \{p\}$. Then $G$ has a normal Hall $p'$-subgroup.*

*Proof.* Consider $F(G)$, the Fitting group of $G$. Since $F(G)$ is a nilpotent normal subgroup of $G$, it is the direct product of its Sylow subgroups, which in turn are all normal in $F(G)$. Thus $F(G)$ is a direct product of $O_q(F(G))$ (actually $O_q(F(G)) = O_q(G)$) for all primes $q \in \pi(F(G)) \subseteq \pi(G)$, and since each $O_q(F(G))$ is characteristic in $F(G)$ and $F(G) \lhd G$ we have $O_q(F(G)) \lhd G$. Therefore $F(G)$ is a $p'$-subgroup of $G$ (recall that $O_p(G) = 1$). As each $O_q(F(G))$ is cyclic, $F(G)$ is cyclic and hence $\mathrm{Aut}(F(G))$ is abelian. Furthermore, by Proposition 5.4 we have that $C_G(F(G)) = F(G)$, and using Proposition 2.6 we obtain

$$I \cong \frac{N_G(F(G))}{C_G(F(G))} = \frac{G}{F(G)}$$

for some subgroup $I \leq \mathrm{Aut}(F(G))$. Hence $G/F(G)$ is abelian. Thus any $H \in \mathrm{Hall}_{p'}(G/F(G))$ is normal in $G/F(G)$ and therefore $HF(G) \lhd G$ and $HF(G) \in \mathrm{Hall}_{p'}(G)$. □

**Theorem 5.9.** *Let $G$ be a finite soluble group and let $p \in \pi(G)$ be such that $G$ has a cyclic Sylow $q$-subgroup for each $q \in \pi(G) \setminus \{p\}$. Further, let $S \in \mathrm{Syl}_p(G)$, $T \in \mathrm{Hall}_{p'}(G)$ and $K = O_p(G)$. Then $G = KN_S(T)T$.*

*Proof.* Apply Proposition 5.8 to $G/K$, noting that $TK/K \in \mathrm{Hall}_{p'}(G/K)$. We have $TK \lhd G$ and by the Frattini argument for Hall subgroups (Corollary 2.13), $G = N_G(T)KT = KN_S(T)T$ since $G = ST$ (Lemma 5.1). □

**Corollary 5.10.** *Let $G$ be a soluble group such that $|G| = p^n \Pi_{i \in I} p_i$ where $p, p_i$ for $i \in I$ are distinct primes, and suppose that $S \in \mathrm{Syl}_p(G)$, $K = O_p(G)$ and $T \in \mathrm{Hall}_{p'}(G)$. Then $G = KN_S(T)T$.*

*Proof.* Let $P_i \in \mathrm{Syl}_{p_i}(G)$, then $|P_i| = p_i$ and therefore $P_i$ is cyclic for each $i \in I$. Hence the result follows from Theorem 5.9. □

We now give an outline of the formal procedure in AutomorphismGroupConjugationSearch (see Algorithm 5.4). We note that our implementation doesn't check if $G = KN_S(T)T$; in this case the search detailed in Lines 19 to 25 does not run since $\langle D_{R_K} \rangle = N$.

**Algorithm 5.4:** AutomorphismGroupConjugationSearch

```
Input: $(G, S, K, T)$ : A soluble group $G$ with $S \in \mathrm{Syl}_p(G)$, $K = O_p(G)$,
       $T \in \mathrm{Hall}_{p'}(G)$ and $C_T(K) = 1$
Output: $A$ : The automorphism group of $G$

$A_S :=$ AutomorphismGroupPGroup($S$);
$\phi_S, R_S :=$ Representation($A_S$);                    /* See Section 3.1 */
$A_{S,K} :=$ FixSubgroup($A_S, K$);
$N_S(T) :=$ Normaliser($S, T$);
$A_{S,K,N} :=$ FixSubgroup($A_{S,K}, N_S(T)$);
$A_K :=$ AutomorphismGroupPGroup($K$);
$\phi_K, R_K :=$ Representation($A_K$);                    /* See Section 3.1 */
$N :=$ Normaliser($R_K, \langle\, (\bar{t}|_K)^{\phi_K} : t \in$ PCGenerators($T$) $\rangle$);
/* Subgroup of $R_K$ which has been extended                              */
$A_G := [\,]$; $D_{R_K} := [\,]$;
/* Loop through the generators before performing full search */
for $n \in$ Generators($N$) do
    $\alpha_G := [s \mapsto s^{n^{\phi^{-1}}} : s \in$ PCGenerators($S$)$] \cup [t \mapsto$ FindConjugatingElement($T, S, \bar{t}^{\,n}$) $: t \in$ PCGenerators($T$)$]$;
    if IsHomomorphism($G, G, \alpha_G$) then
        Append($\sim A_G, \alpha_G$); Append($\sim D_{R_K}, n$);
/* Construct a list of double coset representatives                */
$R :=$ DoubleCosetRepresentatives($N, \langle D_{R_K}\rangle, \langle D_{R_K}\rangle$);
$i := 1$;          /* Start at 1 since the 0 entry is the identity */
while $i <$ Length($R$) do
    $n := R[i]$;
    $\alpha_G := [s \mapsto s^{n^{\phi^{-1}}} : s \in$ PCGenerators($S$)$] \cup [t \mapsto$ FindConjugatingElement($T, S, \bar{t}^{\,n}$) $: t \in$ PCGenerators($T$)$]$;
    if IsHomomorphism($G, G, \alpha_G$) then
        Append($\sim A_G, \alpha_G$); Append($\sim D_{R_K}, n$); $i := 0$;
        $R :=$ DoubleCosetRepresentatives($N, \langle D_{R_K}\rangle, \langle D_{R_K}\rangle$);
    $i := i + 1$;
/* Construct Inn($G$)                                                     */
for $g \in$ Generators($G$) do
    Append($\sim A_G$, InnerAutomorphism($G, g$));
return $\langle \alpha : \alpha \in A_G \rangle$;
```

## 5.4 Full algorithm and summary

For completeness, we now include the full procedure, AutomorphismGroupSolubleGroup, given in Algorithm 5.5.

**Algorithm 5.5:** AutomorphismGroupSolubleGroup

```
Input: $G$ : A finite soluble group
Output: $A$ : The automorphism group of $G$
Determine a sensible value of $p$, so $O_p(G) \neq 1$;  /* See Section 5.4.1 */
$K :=$ pCore$(G, p)$;
$L :=$ Core$(G, p')$;
if $|K||L| = |G|$ then
    return AutomorphismGroupDirectProduct$(G, K, L)$;
if $L \neq 1$ then
    return AutomorphismGroupSubdirectProduct$(G, K, L)$;
$S :=$ SylowSubgroup$(G, p)$;
$T :=$ HallSubgroup$(G, p')$;
if $|K| = |S|$ then
    return AutomorphismGroupSemidirectProduct$(G, S, T)$;
return AutomorphismGroupConjugateSearch$(G, S, K, T)$;
```

In summary of the results of the last few sections, we make several remarks which will be referred to later. Continuing the use of Notation (†), given a soluble group $G$ and an appropriate Sylow $p$-subgroup S, we can compute $\mathrm{Aut}(G)$ by constructing subgroups $A, B \leq \mathrm{Aut}(G)$ such that $\mathrm{Aut}(G) = AB$, where $A$ is computed by extending automorphisms from a specific subgroup $\Sigma \leq \mathrm{Aut}(S)$, and $B$ is either 1 or $\mathrm{Inn}(G)$. In order to construct a simplified view of the whole algorithm for later use, we define $A$ and $B$ for each of the sub-cases of the algorithm:

- **Direct Product** Every automorphism $\sigma \in \mathrm{Aut}(S)$ extends to an automorphism of $G$. Thus $\Sigma = \mathrm{Aut}(S)$ and each $\sigma \in \Sigma$ gives rise to the set of automorphisms $\mathcal{T}_\sigma = \{st \mapsto s^\sigma t^\tau : \tau \in \mathrm{Aut}(T)\} \subset \mathrm{Aut}(G)$. Setting $A = \{\mathcal{T}_\sigma : \sigma \in \mathrm{Aut}(S)\}$ gives $\mathrm{Aut}(G) = AB$ with $B = 1$.

- **Subdirect Product** $L = O_{p'}(G) > 1$; i.e. $T \lhd G$ or $C_T(K) \neq 1$. Each $\alpha \in \mathrm{Aut}(G)$ is induced by some $\overline{\alpha} \in A' \leq \mathrm{Aut}(G/K) \times \mathrm{Aut}(G/L)$. Thus we have a similar situation to the direct product case. Each $\overline{\alpha}$ fixes the image of $\psi : G \to G/K \times G/T$ where $\psi : g \mapsto (gK, gT)$ for $g \in G$. Let $\kappa_{\alpha_T} = \{(\alpha_K, \alpha_T) : (\alpha_K, \alpha_T) \in A'\}$ and $A = \{\psi \kappa_{\alpha_T} \psi^{-1} : \alpha_T \in A_T\}$. Thus $\mathrm{Aut}(G) = AB$ with $B = 1$.

- **Semidirect Product** $C_T(K) = 1$. Each automorphism $\sigma \in \Sigma = N_{\mathrm{Aut}(S)}(\overline{T}|_S)$ can be extended to an automorphism of $G$ fixing $T$. These induced automorphisms of $G$ give rise to the subgroup $A \leq \mathrm{Aut}(G)$. Then $\mathrm{Aut}(G) = AB$ where $B = \mathrm{Inn}(G)$.

- **Conjugate Search** $C_T(K) = 1$. We construct $\Sigma' = \{\sigma \in \mathrm{Aut}(S) : K^\sigma = K,\ N_S(T)^\sigma = N_S(T),\ \sigma|_K \in N_{\mathrm{Aut}(K)}(\overline{T}|_K)\}$, and perform a search to find $\Sigma \leq \Sigma'$: the automorphisms of $S$ which extend to automorphisms of $G$ fixing $T$. Again, these induced automorphisms of $G$ generate the subgroup $A$ and we have $\mathrm{Aut}(G) = AB$ where $B = \mathrm{Inn}(G)$.

This information will be particularly useful in the next section and in Chapter 7, where given an automorphism $\sigma \in \Sigma$ we will need to determine the induced automorphism(s) of $G$.

### 5.4.1 Selection of appropriate $p$ values

As part of the algorithm, we also need to determine a sensible value of $p$ from the given finite soluble group $G$. In terms of performance, it is generally preferred that $p$ is taken to be the prime which creates the largest Sylow $p$-subgroup of $G$ with non-trivial $p$-Core; i.e.

$$p = \arg\max_{p\in\pi(G)} \left\{|P| : P \in \mathrm{Syl}_p(G), O_p(G) \neq 1\right\}.$$

## 5.5 Determining solubility and PC presentations

In this section we tackle the problem of constructing polycyclic presentations for soluble automorphism groups of soluble groups. Given an automorphism group $\mathrm{Aut}(G)$ for some soluble group $G$, the task is to construct a polycyclic presentation $\mathrm{Aut}(G)_{PC}$ and an isomorphism $\varphi_G : \mathrm{Aut}(G) \to \mathrm{Aut}(G)_{PC}$. We will use this notation throughout.

In general, this is not a straightforward calculation. Starting with the generating set of an automorphism group, the current strategy is to construct a permutation representation and test for solubility. A PC presentation is then computed if the group is soluble. This approach relies on being able to quickly construct a practical (low-degree) permutation representation, greatly limiting effectiveness for large examples.

In the method described here, we attempt to determine the solubility of the resulting automorphism group by testing the solubility of groups used in intermediary steps during its construction. This does not hinder the calculation, as we already use PC presentations over other group types whenever possible. If it is found to be soluble, we the can construct a PC presentation for the whole automorphism group by extending the PC presentation of an intermediary group.

### 5.5.1 Direct and subdirect products

Suppose that $\mathrm{Aut}(G)$ has been constructed using either the direct or subdirect product methods outlined in the previous section. In both cases we rely on recursion to construct the PC presentation: i.e. if polycyclic presentations for the two component parts have been constructed, then we glue them together to form a polycyclic presentation for the whole group. Note the following well known result.

**Theorem 5.11.** *Let $X$ be a group and $U, V \leq X$ such that $X \cong U \times V$. Then $X$ is soluble if and only if $U$ and $V$ are soluble.*

*Proof.* See [Rot95, Thm 5.15 and Cor 5.18]. □

### Direct product

For the direct product construction we proceed as follows. Given polycyclic presentations for $\mathrm{Aut}(S)$ and $\mathrm{Aut}(T)$, say $\mathrm{Aut}(S)_{PC}$ and $\mathrm{Aut}(T)_{PC}$ with corresponding mappings $\varphi_S : \mathrm{Aut}(S) \to \mathrm{Aut}(S)_{PC}$ and $\varphi_T : \mathrm{Aut}(T) \to \mathrm{Aut}(T)_{PC}$, we compute the direct product $\mathrm{Aut}(S)_{PC} \times \mathrm{Aut}(T)_{PC}$ from the diagram below.

$$\begin{tikzcd}
\mathrm{Aut}(G) \arrow[r, "\Phi"] \arrow[d, "\varphi_G"] & \mathrm{Aut}(S) \times \mathrm{Aut}(T) \arrow[d, "\varphi_S \times \varphi_T"] \\
\mathrm{Aut}(G)_{PC} \arrow[r, "\Psi"] & \mathrm{Aut}(S)_{PC} \times \mathrm{Aut}(T)_{PC}
\end{tikzcd} \tag{5.2}$$

Thus we have $\mathrm{Aut}(G)_{PC}$ and $\varphi_G : \mathrm{Aut}(G) \to \mathrm{Aut}(G)_{PC}$. Given an automorphism $\alpha \in \mathrm{Aut}(G)$ we construct $\alpha_S = \alpha|_S : S \to S$ and $\alpha_T = \alpha|_T : T \to T$. Then $\alpha \mapsto^{\Phi} (\alpha_S, \alpha_T) \mapsto (\alpha_S^{\varphi_S}, \alpha_T^{\varphi_T})$. Similarly given an element $\alpha_{PC} \in \mathrm{Aut}(G)_{PC}$, compute $\alpha_{PC}^{\Psi(\varphi_S,\varphi_T)\Phi^{-1}}$ to obtain the corresponding automorphism of $G$. The formal listing of this procedure, PCGroupDirectProduct, is given in Algorithm 5.6.

### Subdirect product

In our calculation of $\mathrm{Aut}(G)$ for subdirect cases, we construct a subgroup $A' \leq \mathrm{Aut}(G/K) \times \mathrm{Aut}(G/L)$ where $L = O_{p'}(G)$, and we recall that in doing so we need to have obtained manageable representations of both $\mathrm{Aut}(G/K)$ and $\mathrm{Aut}(G/L)$. Thus, if $\mathrm{Aut}(G/K)$ and $\mathrm{Aut}(G/L)$ are soluble and have PC representations, we can construct a direct product of their PC representations, and then derive the PC

```
Algorithm 5.6: PCGroupDirectProduct

Input: $(A, \varphi_S, \varphi_T)$ : $A = \mathrm{Aut}(G)$ for a soluble group $G$, with $\mathrm{Aut}(S)$,
       $\mathrm{Aut}(T)$ soluble, and $\varphi_S$, $\varphi_T$ the corresponding PC representation
       mappings
Output: $(A_{PC}, \varphi_G)$ : $A_{PC} = \mathrm{Aut}(G)_{PC}$ a polycyclic representation of
        $\mathrm{Aut}(G)$, and a mapping $\varphi_G : \mathrm{Aut}(G) \to \mathrm{Aut}(G)_{PC}$
$A_{PC}, \varphi :=$ DirectProduct($\mathrm{Aut}(S)_{PC}, \mathrm{Aut}(T)_{PC}$);
$\varphi_G := \varphi^{-1} \circ (\varphi_S \times \varphi_T)$;
return $\mathrm{Aut}(G)_{PC}, \varphi_G$
```

representation for $A'$. The process is conceptually equivalent to the direct product case, interchanging $S$ with $G/K$ and $T$ with $G/L$, thus we omit the pseudocode.

In situations where $\mathrm{Aut}(G/L)$ and $\mathrm{Aut}(G/K)$ are not both soluble, then we can construct a direct product of their representations and test if the image of $\mathrm{Aut}(G)$ in $\mathrm{Aut}(G/L) \times \mathrm{Aut}(G/K)$ is soluble. If this is soluble, then we can construct a PC presentation using existing machinery.

### 5.5.2 Conjugation action

Recall that in Section 5.3 we determine the automorphism group of a soluble group $G$ by finding a subgroup $\Sigma \leq \mathrm{Aut}(S)$, where each $\sigma \in \Sigma$ extends to an automorphism of $G$ fixing $S$ and $T$. By extending these automorphisms of $S$, we construct $\Gamma = \{\alpha \in \mathrm{Aut}(G) : S^\alpha = S, T^\alpha = T\}$ and then $\mathrm{Aut}(G) = \Gamma I$, where $I = \mathrm{Inn}(G)$, by Theorem 5.7. Since $G$ is soluble, $I$ is soluble, and given this construction of $\mathrm{Aut}(G)$ we now prove that it is possible to determine its solubility from the solubility of $\Sigma$. Note that we handle the two cases $K = S$ and $K \neq S$ simultaneously, as the calculation is the same for both.

**Theorem 5.12.** *Assume the hypothesis given in Notation (†). Define* $\Gamma = \{\alpha \in \mathrm{Aut}(G) : S^\alpha = S, T^\alpha = T\}$, *and let* $\Sigma = \{\sigma \in \mathrm{Aut}(S) : \sigma = \alpha|_S$ *for some* $\alpha \in \Gamma\}$.

*Suppose that* $O_{p'}(G) = 1$*. Then* $\mathrm{Aut}(G)$ *is soluble if* $\Sigma$ *is soluble.*

*Proof.* Firstly we note that the condition $O_{p'}(G) = 1$ guarantees that $C_T(K) = 1$ (by Proposition 5.6). So, using the structure of $\mathrm{Aut}(G)$ as described in Theorem 5.7, we have $\mathrm{Aut}(G) = \Gamma I$ where $I = \mathrm{Inn}(G)$. From Proposition 4.17 we have $\Gamma \cong \Sigma$, and therefore $\Gamma$ is soluble if and only if $\Sigma$ is soluble. Noting that $\mathrm{Out}(G) = \mathrm{Aut}(G)/\mathrm{Inn}(G)$, we have

$$\mathrm{Out}(G) = \frac{\Gamma I}{I} \cong \frac{\Gamma}{\Gamma \cap I}$$

by the second isomorphism theorem, and so now assume that $\Sigma$ is soluble. Clearly $I$ is soluble since $I \cong G/Z(G)$, and so $\Gamma \cap I$ is soluble. Hence $\Gamma/(\Gamma \cap I)$ and therefore $\mathrm{Out}(G)$ is soluble. □

Briefly moving away from the context of Notation (†), we note that we have proved the following result for general soluble groups.

**Corollary 5.13.** *Let* $G$ *be a soluble group and let* $S \in \mathrm{Syl}_p(G)$ *for some prime* $p \in \pi(G)$ *such that* $O_{p'}(G) = 1$*. Then* $\mathrm{Aut}(G)$ *is soluble if* $\mathrm{Aut}(S)$ *is soluble.*

*Proof.* If $\mathrm{Aut}(S)$ is soluble then $\Sigma$ is soluble and the result follows from Theorem 5.12. □

So given a polycyclic presentation for $\Sigma$, or equivalently $\Gamma$, we want to be able to construct a polycyclic representation for $\mathrm{Aut}(G) = \Gamma I$. Define $\Gamma' = \Gamma/(\Gamma \cap I)$ and suppose that $\Sigma$, and hence $\Gamma'$, is soluble. Further, let

$$\Gamma' = X_1 > X_2 > \cdots > X_{n+1} = 1$$

be a polycyclic series for $\Gamma'$ and define $X = \mathrm{Pc}\langle x_1, x_2, \ldots, x_n \mid R_X\rangle$ to be the corresponding polycyclic presentation. As $G$ is soluble, we also have a polycyclic series

$$I = Y_1 > Y_2 > \cdots > Y_{m+1} = 1$$

and a corresponding polycyclic presentation $Y = \mathrm{Pc}\langle y_1, y_2, \ldots, y_m \mid R_Y\rangle$, for $I \cong G/Z(G)$.

We can now construct a polycyclic series

$$\mathrm{Aut}(G) = X_1Y_1 > X_2Y_1 > \cdots > X_nY_1 > X_{n+1}Y_1 = Y_1 > Y_2 > \cdots > Y_{m+1} = 1$$

and a corresponding polycyclic presentation

$$Z = \mathrm{Pc}\langle x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m \mid {R_X}', R_Y, R_{X,Y}\rangle,$$

which is the presentation of the extension of $I$ by $\Gamma I/I$. Here ${R_X}'$ contains relations corresponding to the series $X_1Y_1 > X_2Y_1 > \cdots > X_nY_1 > X_{n+1}Y_1$, taken from $R_X$ and adapted by multiplying by appropriate elements of $Y_1$. $R_{X,Y}$ contains the "cross" relations between generators of $X$ and $Y$, so relations of the form ${x_i}^{-1}y_jx_i$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. Since each $x_i$ fixes both $S$ and $T$ we have ${x_i}^{-1}y_jx_i \in Y$, thus normal forms for these relations are easily computed.

**Theorem 5.14.** *Let $\Gamma'$ have a polycyclic presentation $X = \mathrm{Pc}\langle x_1, x_2, \ldots, x_n \mid R_X\rangle$ and let $I$ have a polycyclic presentation $Y = \mathrm{Pc}\langle y_1, y_2, \ldots, y_m \mid R_Y\rangle$. Then*

$$\mathrm{Pc}\langle x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m \mid {R_X}', R_Y, R_{X,Y}\rangle$$

*is a polycyclic presentation for* $\mathrm{Aut}(G)$.

*Proof.* Follows from [Joh97, Chapter 10: Presentations of Group Extensions], applied to the extension of $I$ by $\Gamma I/I$. □

Constructing the presentations is now straightforward. Let us assume that $\Sigma_{PC}$ is a polycyclic presentation for $\Sigma$ and $\varphi_\Sigma : \Sigma \to \Sigma_{PC}$ is the corresponding isomorphism. From Proposition 4.17 we have an isomorphism $\phi : \Gamma \to \Sigma$, and so set $\Gamma_{PC} = (\Sigma_{PC})^{\phi^{-1}}$ with $\varphi_\Gamma : \Gamma \to \Gamma_{PC}$ defined to be $\varphi_\Gamma = \phi^{-1}\varphi_\Sigma$. Therefore we can compute

$\Gamma_{PC}$ and $\varphi_\Gamma$ for $\Gamma$ from $\Sigma_{PC}$ and $\varphi_\Sigma$. Now we have

$$\Gamma \xrightarrow{\varphi_\Gamma} \Gamma_{PC} \xrightarrow{\Phi} \frac{\Gamma_{PC}}{(\Gamma \cap I)^{\varphi_\Gamma}} = Q_{PC} \tag{5.3}$$

where $\Phi$ is defined to be the natural mapping. Defining $\Omega : G \to \mathrm{Inn}(G)$ as $\Omega : g \mapsto \omega_g$ where $\omega_g : G \to G$ such that $\omega_g : x \mapsto x^g$, we have

$$G \xrightarrow{\Omega} I \xrightarrow{\varphi_I} I_{PC}. \tag{5.4}$$

which is the natural polycyclic presentation of $I$.

Let $q_1, \ldots, q_n$ and $g_1, \ldots, g_m$ be generators for $Q_{PC}$ and $I_{PC}$ respectively, and let $\overline{q_1}, \ldots, \overline{q_n}$ and $\overline{g_1}, \ldots, \overline{g_m}$ be the corresponding automorphisms in $\mathrm{Aut}(G)$. Define $\mathrm{Aut}(G)_{PC}$ to be the PC representation of $\mathrm{Aut}(G)$, and construct the presentation relations of $\mathrm{Aut}(G)_{PC}$ as follows:

1. Set $\widetilde{q_1}, \ldots, \widetilde{q_n}, \widetilde{g_1}, \ldots, \widetilde{g_m}$ to be the generators of $\mathrm{Aut}(G)_{PC}$, where $\widetilde{q_i}$ corresponds to the generator $q_i$ of $Q_{PC}$ for $1 \leq i \leq n$, and $\widetilde{g_j}$ corresponds to the generator $g_j$ of $I_{PC}$ for $1 \leq j \leq m$. Note that for any element $g \in G$, we write $\widetilde{g}$ to represent the element of $\mathrm{Aut}(G)_{PC}$ corresponding to the map $\overline{g} \in I$ which has PC representative $g^{\Omega\varphi_I} \in I_{PC}$.

2. Construct the relations of $\mathrm{Aut}(G)_{PC}$ derived from relations of $Q_{PC}$, by evaluating the relations of $Q_{PC}$ in $\mathrm{Aut}(G)$. These will be the same up to multiplication of $\overline{g}$ for some $\overline{g} \in I$.

   So for each relation $R(q_1, \ldots, q_n) = 1_{Q_{PC}}$ in $Q_{PC}$:

   (a) Compute $x := \textsc{FindConjugatingElement}(G, G, R(\overline{q_1}, \ldots, \overline{q_n}))$.

   (b) So $R(\overline{q_1}, \ldots, \overline{q_n})\overline{x^{-1}} = 1_{\mathrm{Aut}(G)}$ is the corresponding relation in $\mathrm{Aut}(G)$. Add relation $R(\widetilde{q_1}, \ldots, \widetilde{q_n}) \cdot \widetilde{x^{-1}} = 1_{\mathrm{Aut}(G)_{PC}}$ to $\mathrm{Aut}(G)_{PC}$.

3. Construct relations of $\mathrm{Aut}(G)_{PC}$ derived from relations of $I_{PC}$.

For each relation $R(g_1, \ldots, g_m) = 1_{I_{PC}}$ in $I_{PC}$, add relation $R(\widetilde{g_1}, \ldots, \widetilde{g_m}) = 1_{\mathrm{Aut}(G)_{PC}}$ to $\mathrm{Aut}(G)_{PC}$.

4. Construct conjugation relations between generators from $Q_{PC}$ and generators from $I_{PC}$. Noting that $\overline{g_j}^{\overline{q_i}} = (\overline{q_i})^{-1}\overline{g_j q_i}$, we can write the action of this map on an element $x \in G$ as $x \mapsto [(\overline{q_i})^{-1}\overline{g_j}(\overline{q_i})](x) = (\overline{g_j^{\overline{q_i}}})(x)$.

   For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$:

   (a) Compute $x :=$ FindConjugatingElement$(G, G, \overline{g_j^{\overline{q_i}}})$.

   (b) Add relation $\widetilde{g_j}^{\widetilde{q_i}} = \widetilde{x}$ to $\mathrm{Aut}(G)_{PC}$.

The pseudocode for this procedure, ConstructPCRelations, is given in Algorithm 5.7.

Now we need to construct an isomorphism $\varphi_G : \mathrm{Aut}(G) \to \mathrm{Aut}(G)_{PC}$; i.e. for any automorphism $\alpha \in \mathrm{Aut}(G)$ construct a product of the PC generators $\widetilde{q_1}, \ldots, \widetilde{q_n}, \widetilde{g_1}, \ldots, \widetilde{g_m}$ of $\mathrm{Aut}(G)_{PC}$ which represents $\alpha$, and vice versa.

Given a PC element in $\mathrm{Aut}(G)_{PC}$ is it easy to construct the corresponding map: simply evaluate the word using the PC automorphism generators of $\mathrm{Aut}(G)$ determined earlier. However, constructing a PC representation of an arbitrary element $\alpha \in \mathrm{Aut}(G)$ is not straightforward, as determining a word in the PC automorphism generators is not immediately possible. In this situation, we can build the corresponding PC element using the underlying structure of $\mathrm{Aut}(G)$, as described in the following sections.

### Semidirect product

Using Notation (†) and recalling the results from Section 5.3, suppose that we are given an arbitrary automorphism $\alpha \in \mathrm{Aut}(G)$ where $S \lhd G$ and $T \ntriangleleft G$. Then $\alpha$ acts on $T$ by conjugation and so we can find an element $s \in S$ such that $T^\alpha = T^s$. Following our notation, the PC representation of $s$ in $\widetilde{g_1}, \ldots, \widetilde{g_m}$ is given by $\widetilde{s}$. We

**Algorithm 5.7:** ConstructPCRelations

```
Input: $(I, Q_{PC}) : I = \mathrm{Inn}(G)$ and $Q_{PC}$ is a polycyclic representation of
        $\Gamma_{PC}/(\Gamma \cap I)^{\varphi_\Gamma}$
Output: $R$ : A sequence of relations of a PC presentation
        $\mathrm{Aut}(G)_{PC} = \mathrm{Pc}\langle \widetilde{q_1}, \ldots, \widetilde{q_n}, \widetilde{g_1}, \ldots, \widetilde{g_m} \mid R \rangle$ for $\mathrm{Aut}(G)$

/* Sequence for relations of $\mathrm{Aut}(G)_{PC}$ */
$R := [\,]$;
/* Iterate through the relations of $Q_{PC}$ */
for $R(X_1, \ldots, X_n) \in$ Relations($Q_{PC}$) do
    $x :=$ FindConjugatingElement($G, G, R(\overline{q_1}, \ldots, \overline{q_n})$);
    Append($\sim R, R(\widetilde{q_1}, \ldots, \widetilde{q_n}) \cdot \widetilde{x^{-1}}$);
/* Iterate through the relations of $I_{PC}$ */
for $R(X_1, \ldots, X_m) \in$ Relations($I_{PC}$) do
    Append($\sim R, R(\widetilde{g_1}, \ldots, \widetilde{g_m})$);
/* Construct the cross relations between $Q_{PC}$ and $I$ */
for $i \in [1 \ldots n]$ do
    for $j \in [1 \ldots m]$ do
        $x :=$ FindConjugatingElement($G, G, \overline{g_j^{\overline{q_i}}}$);
        Append($\sim R, \widetilde{g_j}^{\widetilde{q_i}} = \widetilde{x}$);
return $R$;
```

now construct $\alpha_T = \alpha\overline{s^{-1}}$ which fixes $T$, and thus $\alpha_T \in \Gamma$. Therefore we can compute $(\alpha_T)^{\varphi_\Gamma \Phi}$ to obtain the remaining product in $\widetilde{q}_1, \ldots, \widetilde{q}_n$, and $\varphi_G(\alpha) = (\alpha_T)^{\varphi_\Gamma \Phi}\widetilde{s}$. The full procedure, PCGroupSemidirectProductMap, is outlined in Algorithm 5.8.

**Algorithm 5.8:** PCGroupSemidirectProductMap

```
Input: $(A^*, \alpha)$ : $A^*$ contains $\mathrm{Aut}(G)$ and PC representation information,
       $\alpha \in \mathrm{Aut}(G)$
Output: The element of $\mathrm{Aut}(G)_{PC}$ which represents the action of $\alpha$ on $G$
/* Find an element $s \in S$ such that $T^\alpha = T^{\overline{s}}$ */
$s$ := FindConjugatingElement($S, T, \alpha$);
/* Construct the automorphism fixing $S$ and $T$ */
$\alpha_\Gamma := \alpha \circ \overline{s^{-1}}$;
/* Compute the corresponding element in $Q_{PC}$ */
$q_\Gamma := (\alpha_\Gamma)^{\varphi_\Gamma \Phi}$;
$g$ := FindConjugatingElement($G, G, \alpha \circ (\alpha_\Gamma)^{-1}$);
return $\widetilde{q}_\Gamma \cdot \widetilde{g}$;
```

### Conjugate search

Adapting the ideas from the semidirect product case, we start by finding an element $s \in S$ such that $T^\alpha = T^s$, and then construct $\alpha_T$ as before. So $\alpha_T = \alpha\overline{s^{-1}}$. Further, we can now find an element $t \in T$ such that $S^{\alpha_T} = S^t$, and we can compute $\alpha_{S,T} = \alpha_T\overline{t^{-1}}$, an automorphism fixing both $S$ and $T$. Hence $\alpha_{S,T} \in \Gamma$ and $\varphi_G(\alpha) = (\alpha_{S,T})^{\varphi_\Gamma \Phi}\widetilde{st}$. The full procedure, PCGroupConjugateSearchMap is given in Algorithm 5.9.

**Algorithm 5.9:** PCGroupConjugationSearchMap

```
Input: $(A^*, \alpha)$ : $A^*$ contains $\mathrm{Aut}(G)$ and PC representation information,
       $\alpha \in \mathrm{Aut}(G)$
Output: The element of $\mathrm{Aut}(G)_{PC}$ which represents the action of $\alpha$ on $G$
/* Find an element $s \in S$ such that $T^\alpha = T^{\overline{s}}$ */
$s :=$ FindConjugatingElement$(S, T, \alpha)$;
/* Construct the automorphism fixing $T$ */
$\alpha_T := \alpha \circ \overline{s^{-1}}$;
/* Find an element of $T$ such that $S^{\alpha_T} = S^{\overline{t}}$ */
$t :=$ FindConjugatingElement$(T, S, \alpha_T)$;
/* Construct the automorphism fixing $S$ and $T$ */
$\alpha_\Gamma := \alpha_T \circ \overline{t^{-1}}$;
/* Compute the corresponding element in $Q_{PC}$ */
$q_\Gamma := \alpha_\Gamma^{\varphi_\Gamma \Phi}$;
$g :=$ FindConjugatingElement$(G, G, \alpha \circ (\alpha_\Gamma)^{-1})$;
return $\widetilde{q_\Gamma} \cdot \widetilde{g}$;
```

# Chapter 6

# Isomorphism testing for soluble groups

In this chapter we adapt the methods outlined in Section 4.2 to perform isomorphism testing between finite soluble groups. Many features of the resulting procedure mirror those of the automorphism group algorithm given in Chapter 5.

We begin by defining some notation for this purpose, modifying the notation given as Notation (†) in the previous chapter. We add an index subscript $i \in 1, 2$ to distinguish between the two finite soluble groups $G_1$, $G_2$ and their subgroup components required for the algorithms. Note that if $|G_1| \neq |G_2|$ then it is trivial to deduce that $G_1 \ncong G_2$, therefore we assume that the orders of the two groups are equal.

**Notation (‡)**

Let $G_1$ and $G_2$ be finite soluble groups where $|G_1| = |G_2| = p^n q$ for some prime $p \in \mathbb{N}$ and $q \in \mathbb{N}$ with $\mathrm{hcf}(p, q) = 1$. Define $S_i \in \mathrm{Syl}_p(G_i)$, $T_i \in \mathrm{Hall}_q(G_i)$ and $K_i = O_p(G_i)$ and assume that $p$ has been chosen such that $K_i \neq 1$ for $i = 1, 2$.

We now recall the notion of extending isomorphisms between subgroups to cover their parent groups (from Section 4.2.3), and in particular the result of Lemma 4.23: an isomorphism $\sigma : S_1 \to S_2$ extends to an isomorphism $G_1 \to G_2$ if there exists an isomorphism $\tau : T_1 \to T_2$ such that $st \mapsto s^\sigma t^\tau$ is an isomorphism $G_1 \to G_2$.

Following the algorithms in Chapter 5, we will be attempting to construct an isomorphism $G_1 \to G_2$ using isomorphisms $S_1 \to S_2$ and $T_1 \to T_2$. We begin by outlining three lemmas which provide starting points for all of the proceeding results.

**Lemma 6.1.** *Let $X_1, X_2$ be finite soluble groups and let $\Phi : X_1 \to X_2$ be an isomorphism. For a given set of primes $\pi \subseteq \pi(X_1)$, take $H_1 \in \mathrm{Hall}_\pi(X_1)$ and $H_2 \in \mathrm{Hall}_\pi(X_2)$. Then there exists $x \in X_2$ such that $H_1^{\Phi\overline{x}} = H_2$, and in particular, $H_1 \cong H_2$.*

*Proof.* Note that $|(H_1)^\Phi| = |H_1|$ for any $H_1 \in \mathrm{Hall}_\pi(X_1)$, and hence $H_1^\Phi \in \mathrm{Hall}_\pi(X_2)$. Now take $H_2 \in \mathrm{Hall}_\pi(X_2)$ and since all Hall $\pi$-subgroups are conjugate there exists $x \in X_2$ such that $x^{-1}H_1^\Phi x = H_2$. Hence $\Phi\overline{x}|_{H_1} : H_1 \to H_2$ is an isomorphism. □

**Lemma 6.2.** *Let $X_1, X_2$ be finite soluble groups and let $\Phi : X_1 \to X_2$ be an isomorphism. Then for a given set of primes $\pi \subseteq \pi(X_1)$, $|\mathrm{Hall}_\pi(X_1)| = |\mathrm{Hall}_\pi(X_2)|$.*

*Proof.* Let $k_i = |\mathrm{Hall}_\pi(X_i)|$ and $H_i \in \mathrm{Hall}_\pi(X_i)$ for $i = 1, 2$. Then $k_i = |X_i|/|N_{X_i}(H_i)|$ for $i = 1, 2$, and since $\Phi$ is an isomorphism, $N_{X_1}(H_1)^\Phi = N_{X_2}(H_2)$. □

**Lemma 6.3.** *Let $X_1, X_2$ be finite groups, let $\Phi : X_1 \to X_2$ be an isomorphism and for $i = 1, 2$ define $W_i = O_p(X_i)$ for a given prime $p \in \pi(X_1)$. Then $W_1^\Phi = W_2$.*

*Proof.* By Lemma 2.15, $W_1$ is the largest normal $p$-subgroup in $X_1$, and as $|W_1^\Phi| = |W_1|$, $W_1^\Phi$ is the largest normal $p$-subgroup of $X_2$ and this is unique. Thus $W_1^\Phi = W_2$. □

Thus, now we need to prove that given isomorphisms $\sigma : S_1 \to S_2$ and $\tau : T_1 \to T_2$, we can construct an isomorphism $\Phi : G_1 \to G_2$. We assume that $S_1 \cong S_2$,

$T_1 \cong T_2$ and $K_1 \cong K_2$ otherwise $G_1$ and $G_2$ are not isomorphic by Lemmas 6.1 and 6.3. Similarly assume that $S_1 \lhd G_1 \iff S_2 \lhd G_2$, $T_1 \lhd G_1 \iff T_2 \lhd G_2$ otherwise $G_1$ and $G_2$ are not isomorphic since $|\operatorname{Hall}_\pi(G_1)| = |\operatorname{Hall}_\pi(G_2)|$ for all subsets $\pi \subseteq \pi(G)$ (Lemma 6.2).

Now, as in the automorphism group algorithm from Chapter 5, we can split into the following four sub-cases:

1. **Direct Product**

   $S_i, T_i \lhd G_i$ for $i = 1, 2$.

2. **Subdirect Product**

   $L_i = O_{p'}(G_i) \neq 1$ for $i = 1, 2$. Here we construct subdirect products $\psi_i : G_i \to G_i/K_i \times G_i/L_i$ for $i = 1, 2$ and recurse.

3. **Semidirect Product**

   $S_i \lhd G_i$ and $T_i \ntriangleleft G_i$, giving $G_i \cong S_i \rtimes T_i$ for $i = 1, 2$.

4. **General Method (Conjugation Search)** $S_i \ntriangleleft G_i$ for $i = 1, 2$.

## 6.1 Direct product

In this section we assume that $S_i, T_i \lhd G_i$ and hence $S_i, T_i \operatorname{char} G_i$ for $i = 1, 2$. Using Proposition 2.3 and Proposition 2.4 we have $G_1 \cong S_1 \times T_1$ and $G_2 \cong S_2 \times T_2$ and Theorem 4.19 implies that $G_1 \cong G_2$ if and only if $S_1 \cong S_2$ and $T_1 \cong T_2$. Thus, where we have isomorphisms $\sigma : S_1 \to S_2$ and $\tau : T_1 \to T_2$ we construct an isomorphism $\Phi : G_1 \to G_2$ defined by $\Phi : st \to s^\sigma t^\tau$.

The algorithm IsIsomorphicDirectProduct is given in Algorithm 6.1.

**Algorithm 6.1:** IsIsomorphicDirectProduct

```
Input: $(G_i, S_i, T_i)$ : $G_i$ are finite soluble groups, $S_i \in \mathrm{Syl}_p(G_i)$ and
       $T_i \in \mathrm{Hall}_{p'}(G_i)$ for $i = 1, 2$
Output: $(b, \Phi)$ : Boolean value $b$: true if $G_1 \cong G_2$, false otherwise; map
        $\Phi : G_1 \to G_2$ if groups are isomorphic, null otherwise
$b_S, \phi_S :=$ IsIsomorphicPGroup($S_1, S_2$);
if $b_S \neq$ true then
    return false, null;
$b_T, \phi_T :=$ IsIsomorphicSolubleGroup($T_1, T_2$);
if $b_T \neq$ true then
    return false, null;
Define $\phi_G : G \to G$ by $\phi_G : st \to s^{\phi_S} t^{\phi_T}$;
return true, $\phi_G$;
```

## 6.2 Subdirect product

Now suppose that we have $L_i = O_{p'}(G) > 1$ for $i = 1, 2$. Then, as in Section 5.2, we can define subdirect products $\psi_i : G_i \to G_i/K_i \times G_i/L_i$ for $i = 1, 2$. We then use the results of Section 4.2.2, and so by Theorems 4.20 and 4.21 we attempt to construct isomorphisms $\Psi_K : G_1/K_1 \to G_2/K_2$ and $\Psi_L : G_1/L_1 \to G_2/L_2$ such that $G_1^{\psi_1(\Psi_K, \Psi_L)} = G_2^{\psi_2}$.

The full procedure, IsIsomorphicSubdirectProduct, is given in Algorithm 6.2.

## 6.3 Conjugation action

Now we consider the cases where $O_{p'}(G_i) = 1$ for $i = 1, 2$, and recall that this implies that $C_{T_i}(K_i) = 1$ for $i = 1, 2$. Therefore $G_1$, $G_1$ and their associated subgroups defined in (‡) satisfy Hypothesis 4.24. We now tackle the remaining two cases.

**Algorithm 6.2:** IsIsomorphicSubdirectProduct

```
Input: $(G_i, K_i, L_i)$ : $G_i$ are finite soluble groups, $K_i = O_p(G_i)$ and
       $L_i = O_{p'}(G_i)$ for $i = 1, 2$
Output: $(b, \Phi)$ : Boolean value $b$: true if $G_1 \cong G_2$, false otherwise; map
        $\Phi : G_1 \to G_2$ if groups are isomorphic, null otherwise

Define $\varphi_{K_i} : G_i \to G_i/K_i$ and $\varphi_{L_i} : G_i \to G_i/L_i$ for $i = 1, 2$;
$b_K, \Psi_K :=$ IsIsomorphicSolubleGroup($G_1/K_1, G_2/K_2$);
if $b_K \neq$ true then return false, null;
$b_L, \Psi_L :=$ IsIsomorphicSolubleGroup($G_1/L_1, G_2/L_2$);
if $b_L \neq$ true then return false, null;
$A_K :=$ AutomorphismGroup($G_2/K_2$);
$A_L :=$ AutomorphismGroup($G_2/L_2$);
/* Verify that $\Psi_K : L_1/K_1 \mapsto L_2/K_2$                                  */
$b_K, \alpha_K :=$ FindMappingAutomorphism($((L_1^{\varphi_{K_1}})^{\Psi_K}, L_2^{\varphi_{K_2}}, A_K)$);
if $b_K \neq$ true then return false, null;
$\Psi_K := \Psi_K \circ \alpha_K$;
/* Verify that $\Psi_L : K_1/L_1 \mapsto K_2/L_2$                                  */
$b_L, \alpha_L :=$ FindMappingAutomorphism($((K_1^{\varphi_{L_1}})^{\Psi_L}, K_2^{\varphi_{L_2}}, A_L)$);
if $b_L \neq$ true then return false, null;
$\Psi_L := \Psi_L \circ \alpha_L$;
Let $\phi_{i,K,LK} : G_i/K_i \to G_i/L_iK_i$ and $\phi_{i,L,LK} : G_i/L_i \to G_i/L_iK_i$ be the
natural maps for $i = 1, 2$;
/* Construct induced isomorphisms $\Psi_{K,LK} : G_1/L_1K_1 \to G_2/L_2K_2$ and
   $\Psi_{L,LK} : G_1/L_1K_1 \to G_2/L_2K_2$ from $\Psi_K$ and $\Psi_L$ respectively   */
$\Psi_{K,LK} := \phi_{1,K,LK}^{-1} \circ \Psi_K \circ \phi_{2,K,LK}$;
$\Psi_{L,LK} := \phi_{1,L,LK}^{-1} \circ \Psi_L \circ \phi_{2,L,LK}$;
/* Check that $\Psi_{K,LK}$ and $\Psi_{L,LK}$ can be made consistent on
   $G_1/L_1K_1$                                                                     */
$\alpha := [x \mapsto x^{\Psi_{K,LK}^{-1}\Psi_{L,LK}} : x \in$ Generators$(G_2/K_2L_2)]$;
if $\alpha \notin \mathrm{Aut}(G_2/K_2L_2)$ then return false, null;
Let $\alpha_K$ be the lift of $\alpha$ to $A_K$;
Define $\psi_i : G_i \to G_i/K_i \times G_i/L_i$ to be the subdirect product maps for
$i = 1, 2$;
return true, $\psi_1 \circ (\Psi_K \circ \alpha, \Psi_L) \circ \psi_2^{-1}$;
```

### 6.3.1 Semidirect product

In this section we assume that $S_i \lhd G_i$ (and hence $S_i \operatorname{char} G_i$) for $i = 1, 2$, and that there exists an isomorphism $\sigma : S_1 \to S_2$, otherwise $G_1$ and $G_2$ are trivially non-isomorphic. If $\sigma$ satisfies $(\overline{T_1}|_{S_1})^\sigma = \overline{T_2}|_{S_2}$ (up to composition with an element of $\mathrm{Aut}(S_2)$), then $\sigma$ extends to an isomorphism $\Phi : G_1 \to G_2$ by Theorem 4.27, and we use Lemma 4.25 and Lemma 4.26 to construct $\Phi$. If this relation does not hold, then $G_1 \ncong G_2$.

The algorithm, IsIsomorphicSolubleGroupSemidirectProduct, is given in Algorithm 6.3. The following result guarantees that this is sufficient to ensure that $G_1 \cong G_2$.

**Theorem 6.4.** *Assume the hypothesis given in Notation (‡) with $S_i = K_i$ for $i = 1, 2$ and suppose that $G_1 \cong G_2$. Then there exists an isomorphism $\sigma : S_1 \to S_2$ such that $(\overline{T_1}|_{S_1})^\sigma = \overline{T_2}|_{S_2}$.*

*Proof.* Let $\Phi : G_1 \to G_2$ be an isomorphism. By Lemma 6.1 and Lemma 5.2 there exists $s_2 \in S_2$ such that $\Phi\overline{s_2}$ is an isomorphism and $\Phi\overline{s_2} : T_1 \to T_2$. In particular, we can now define $\sigma = \Phi\overline{s_2}|_{S_1} : S_1 \to S_2$ and an isomorphism $\tau = \Phi\overline{s_2}|_{T_1} : T_1 \to T_2$. Clearly $K_1^\sigma = K_2$, and furthermore $\sigma$ and $\tau$ combine to form an isomorphism $G_1 \to G_2$. So by Lemma 4.26 we have that $(\overline{t_1}|_{S_1})^\sigma = \overline{t_1^\tau}|_{S_2}$ for all $t_1 \in T_1$. Thus $(\overline{T_1}|_{S_1})^\sigma = \overline{T_2}|_{S_2}$. □

### 6.3.2 Conjugate search

We now assume that $S_i \ntriangleleft G_i$ and $O_{p'}(G_i)$ for $i = 1, 2$.

**Theorem 6.5.** *Assume the hypothesis given in Notation (‡) and suppose that $G_1 \cong G_2$. Then there exists an isomorphism $\sigma : S_1 \to S_2$ such that $K_1^\sigma = K_2$, $N_{S_1}(T_1)^\sigma = N_{S_2}(T_2)$ and $(\overline{T_1}|_{K_1})^{\sigma|_{K_1}} = \overline{T_2}|_{K_2}$.*

**Algorithm 6.3:** IsIsomorphicSolubleGroupSemidirectProduct

```
Input: $(G_i, S_i, T_i, \sigma)$ : Soluble groups $G_i$ with $S_i \in O_p(G_i)$, $T_i \in \mathrm{Hall}_{p'}(G_i)$
       such that $C_{T_i}(S_i) = 1$ for $i = 1, 2$ and $\sigma : S_1 \to S_2$ is an isomorphism
Output: $(b, \Phi)$ : Boolean $b$ which is either true in which case $\Phi$ is an
        isomorphism $G_1 \to G_2$, or false and $\Phi$ is null

$A_{S_1}$ := AutomorphismGroupPGroup($S_1$);
$\phi_1, R_1$ := Representation($A_{S_1}$);                /* See Section 3.1.  */
$A_{S_2}$ := AutomorphismGroupPGroup($S_2$);
$\phi_2, R_2$ := Representation($A_{S_2}$);                /* See Section 3.1.  */
Construct $A_\sigma : A_{S_1} \to A_{S_2}$;
$b, x$ := FindConjugatingElement($R_2, (\overline{T_1}|_{S_1})^{A_\sigma}, \overline{T_2}|_{S_2}$);
if $b \neq$ true then return false, null;
$\Phi := [s_1 \mapsto s_1^\sigma : s_1 \in$ PCGenerators($S_1$)$] \cup [t_1 \mapsto$
        FindConjugatingElement($T_2, S_2, (\overline{t_1}|_{S_1})^{A_\sigma x}$)$]$;
return true, $\Phi$;
```

*Proof.* Let $\Phi : G_1 \to G_2$ be an isomorphism. By Lemma 6.1 there exists $g \in G_2$ such that $(\Phi\overline{g})|_{S_1} : S_1 \to S_2$ is an isomorphism and further, we can find some $s \in S_2$ (by Lemma 5.2) such that $T_1^{\Phi\overline{g}\,\overline{s}} = T_2$. So relabel $\Phi = \Phi\overline{gs} : G_1 \to G_2$ and define isomorphisms $\sigma = \Phi|_{S_1} : S_1 \to S_2$ and $\tau = \Phi|_{T_1} : T_1 \to T_2$.

Clearly $K_1^\Phi = K_2$ and $T_1^\Phi = (n^{-1}T_1n)^\Phi = (n^{-1})^\Phi T_1^\Phi n^\Phi = (n^{-1})^\Phi T_2 n^\Phi = T_2$ for all $n \in N_{S_1}(T_1)$. Hence $K_1^\sigma = K_2$, $N_{S_1}(T_1)^\sigma = N_{S_2}(T_2)$ and as $\sigma$ and $\tau$ combine to form an isomorphism $G_1 \to G_2$, we have $(\overline{t_1}|_{K_1})^{\sigma|_{K_1}} = \overline{t_1^\tau}|_{K_2}$ for all $t_1 \in T_1$ by Lemma 4.26. Thus $(\overline{T_1}|_{K_1})^{\sigma|_{K_1}} = \overline{T_2}|_{K_2}$.

□

So to construct an isomorphism $G_1 \to G_2$, we begin by assuming that we have an isomorphism $\sigma : S_1 \to S_2$. We check each of the conditions given in the statement of Theorem 6.5; i.e. $K_1^\sigma = K_2$, $N_{S_1}(T_1)^\sigma = N_{S_2}(T_2)$ and $(\overline{T_1}|_{K_1})^{\sigma|_{K_1}} = \overline{T_2}|_{K_2}$ (up to composition with some element of $\mathrm{Aut}(S_2)$). If any of these conditions do not hold, then $G_1$ and $G_2$ are not isomorphic. We then use Lemma 4.25 and Lemma 4.26

to attempt to construct an isomorphism $\tau : T_1 \to T_2$ such that $st \mapsto s^\sigma t^\tau$, for $s \in S_1, t \in T_1$, is an isomorphism $G_1 \to G_2$. Again, if this fails, then $G_1$ and $G_2$ are not isomorphic.

We note that if $G_i = K_i N_{S_i}(T_i) T_i$ for $i = 1, 2$ then an isomorphism $\sigma : S_1 \to S_2$, which satisfies the conditions given in the hypothesis of Theorem 6.5, will extend to an isomorphism $G_1 \to G_2$ (by Theorem 4.28).

The algorithm, IsIsomorphicConjugateSearch, is given in Algorithm 6.4.

**Algorithm 6.4:** IsIsomorphicConjugateSearch

```
Input: $(G_i, S_i, K_i, T_i, \sigma)$ : Soluble groups $G_i$ with $S_i \in O_p(G_i)$,
       $K_i = O_p(G_i)$, $T_i \in \mathrm{Hall}_{p'}(G_i)$ such that $C_{T_i}(K_i) = 1$ for $i = 1, 2$ and
       $\sigma : S_1 \to S_2$ is an isomorphism
Output: $(b, \Phi)$ : Boolean $b$ which is either true in which case $\Phi$ is an
        isomorphism $G_1 \to G_2$, or false and $\Phi$ is null

$A_{S_2}$ := AutomorphismGroupPGroup($S_2$);
$b, m$ := FindMappingAutomorphism($K_1^\sigma, K_2, A_{S_2}$);
if $b \neq$ true then return false, null;
$\sigma := \sigma \circ m$;
$b, m$ := FindMappingAutomorphism($N_{S_1}(T_1)^\sigma, N_{S_2}(T_2), A_{S_2}$);
if $b \neq$ true then return false, null;
$\sigma := \sigma \circ m$;
$A_{K_1}$ := AutomorphismGroupPGroup($K_1$);
$\phi_{K_1}, R_{K_1}$ := Representation($A_{K_1}$);            /* See Section 3.1.  */
$A_{K_2}$ := AutomorphismGroupPGroup($K_2$);
$\phi_{K_2}, R_{K_2}$ := Representation($A_{K_2}$);            /* See Section 3.1.  */
Construct $A_\sigma : A_{K_1} \to A_{K_2}$;
$b, x$ := FindConjugatingElement($A_{K_2}, (\overline{T_1}|_{K_1})^{A_\sigma}, \overline{T_2}|_{K_2}$);
if $b \neq$ true then return false, null;
$\Phi$ := [$s_1 \mapsto s_1^\sigma : s_1 \in$ PCGenerators($S_1$)] $\cup$ [$t_1 \mapsto$
        FindConjugatingElement($T_2, S_2, (\overline{t_1}|_{K_1})^{A_\sigma x}$)];
if IsHomomorphism($G_1, G_2, \Phi$) $\neq$ true then
    return false, null;
return true, $\Phi$;
```

## 6.4 Full algorithm

Again, as in Section 5.4, we provide the full algorithm, IsIsomorphicSolubleGroups (see Algorithm 6.5), which tests two finite soluble groups $G_1, G_2$ for isomorphism.

**Algorithm 6.5:** IsIsomorphicSolubleGroups

```
Input: $(G_1, G_2)$ : Soluble groups $G_1$ and $G_2$
Output: $(b, \Phi)$ : Boolean $b$ which is either true in which case $\Phi$ is an
        isomorphism $G_1 \to G_2$, or false and $\Phi$ is null
if $|G_1| \neq |G_2|$ then return false, null;
Determine a sensible value of $p$ for $G_1$, in particular $O_p(G_1) \neq 1$;
$K_i :=$ pCore$(G_i, p)$ for $i = 1, 2$;
if $|K_1| \neq |K_2|$ then return false, null;
$L_i :=$ Core$(G_i, p')$ for $i = 1, 2$;
if $|L_1| \neq |L_2|$ then return false, null;
if $|K_1||L_1| = |G_1|$ then
    return IsIsomorphicGroupDirectProduct$(G_i, K_i, L_i)$;
if $L \neq 1$ then
    return IsIsomorphicSubdirectProduct$(G_i, K_i, L_i)$;
$S_i :=$ SylowSubgroup$(G_i, p)$ for $i = 1, 2$;
$T_i :=$ HallSubgroup$(G_i, p')$ for $i = 1, 2$;
$b_S, \sigma :=$ IsIsomorphicPGroups$(S_1, S_2)$;
if $b_S \neq$ true then return false, null;
if $|K_1| = |S_1|$ then
    return IsIsomorphicSemidirectProduct$(G_i, S_i, T_i, \sigma)$;
return IsIsomorphicConjugateSearch$(G_i, S_i, K_i, T_i, \sigma)$;
```

# Chapter 7

# Extending methods to non-soluble examples

The soluble group structure defined at the beginning of Chapter 5 (in Notation (†)) lends itself well to the result of Theorem 5.7, which forms the basis of our algorithms for constructing automorphism groups and performing isomorphism testing. In this chapter, we adapt the same ideas to develop similar algorithms for some non-soluble finite groups.

The results that follow are speculative in the sense that they do not form a complete strategy which can be applied to any finite group. Our aim is to introduce alternative approaches to current methods which fit nicely with the rest of the material in this thesis, and therefore the procedures described here will only be suitable for finite groups which match some specific criteria. The general strategy is to construct the automorphism group of a finite group $G$ using the automorphism group of the soluble radical $R = O_\infty(G)$ (recall Definition 2.2). In particular, we rely on the soluble group automorphism algorithm detailed in Chapter 5 to compute $\mathrm{Aut}(R)$. Further calculations with $\mathrm{Aut}(R)$ can then be simplified by using our knowledge of the output structure of our algorithm. For this reason, we will assume, and in some cases explicitly refer back to, details of the algorithm which are given in the summary found in Section 5.4.

We split our study of non-soluble groups into three main categories of finite groups which either:

- can be easily determined to be a direct or subdirect product of $R$ and some other characteristic subgroup,
- have a complement $H$ of $R$ in $G$, such that $G = RH$ and $R \cap H = 1$, or
- have a "crossover prime", i.e. there exists some $p \in \pi(R) \cap \pi(G/R)$.

## 7.1 Identifying direct and subdirect products

In this section we explore several methods for determining if a given finite group $G$ is a direct or subdirect product of $R = O_\infty(G)$ and some other characteristic subgroup $C \leq G$.

**Definition 7.1.** Let $G$ be a finite group. Then the *soluble residual* of $G$, denoted $G^\infty$, is the last term in the derived series for $G$.

We note that $G^\infty \operatorname{char} G$ for every finite group $G$, since $[G, G] \operatorname{char} G$.

**Lemma 7.2.** *Let $G$ be a finite group. Then*

- *$G/G^\infty$ is soluble, and*
- *if $N$ is a normal subgroup of $G$ such that $G/N$ is soluble, $G^\infty \leq N$.*

*Proof.* Observe that any finite group $G$ has a corresponding derived series

$$G = G^{(1)} \geq G^{(2)} \geq \dots G^{(k-1)} \geq G^{(k)} \geq G^{(k+1)} \dots$$

If $G^{(n)} = 1$ for some $n$, then $G$ is soluble (and $G^\infty = G^{(n)} = 1$). So let us assume that $G$ is not soluble, and hence $[G^{(m)}, G^{(m)}] = G^{(m)}$ for some $m \in \mathbb{N}$. Noting that

$[G, G] \operatorname{char} G$, now we have a normal series with abelian factors:

$$\frac{G^{(1)}}{G^{(m)}} \geq \frac{G^{(2)}}{G^{(m)}} \geq \dots \frac{G^{(m-1)}}{G^{(m)}} \geq \frac{G^{(m)}}{G^{(m)}} = 1,$$

and hence $G/G^{(m)}$ is soluble.

Suppose now that $N$ is a normal subgroup of $G$ such that $G/N$ is soluble. Then since $G$ is not soluble, $N$ is not soluble. Note that $[G/N, G/N] = [G, G]N/N$ and so by induction we have $(G/N)^{(i)} = G^{(i)}N/N$ for each $i$. In particular, for some $m \in \mathbb{N}$ we have $(G/N)^{(m)} = 1$, and hence $G^{(m)} \leq N$. Therefore $G^{\infty} \leq N$. □

So we have shown that the soluble residual $G^{\infty}$ of a finite group $G$ is the smallest normal subgroup of $G$ which produces a soluble quotient. In other words, $G/G^{\infty}$ is the largest soluble quotient group of $G$.

**Lemma 7.3.** *Let $G$ be a finite group and define $R = O_{\infty}(G)$. Suppose that $G/R$ is perfect (in particular, simple). Then $G$ is a direct product $R \times C$ where $C \leq G$ if and only if $G/R \cong G^{\infty}$.*

*Proof.* Suppose that $G/R$ is perfect, and we have $G \cong R \times C$ for some $C \leq G$. Then $G/C \cong R$, and hence $G^{\infty} \leq C$. Since $C$ is perfect, $C = C^{\infty} \leq G^{\infty}$, and hence $C = G^{\infty}$. Conversely, if $G/R \cong G^{\infty}$ then we have $R, G^{\infty} \operatorname{char} G$ and $R \cap G^{\infty} = 1$. Hence $G \cong R \times G^{\infty}$. □

Therefore we can determine if a group $G$ is a direct product of its soluble radical and $G^{\infty}$ by testing if $G/R \cong G^{\infty}$. In this situation we compute $\operatorname{Aut}(G)$ as $\operatorname{Aut}(R) \times \operatorname{Aut}(G^{\infty})$ using the result of Theorem 4.3.

So now suppose that $G$ is a finite group and $G/R$ is not perfect. If $G^{\infty} \neq 1$, then we can construct a subdirect product as follows.

**Lemma 7.4.** *Let $G$ be a finite group, define $R = O_{\infty}(G)$ and assume that $R \cap G^{\infty} = 1$. Then $\psi : G \to G/R \times G/G^{\infty}$ where $\psi : g \mapsto (gR, gG^{\infty})$ is a subdirect product.*

*Proof.* We note that $R, G^{\infty} \operatorname{char} G$ and $R \cap G^{\infty} = 1$. Thus by Lemma 4.4, we can construct a subdirect product $\psi : G \to G/R \times G/G^{\infty}$. □

Hence following the hypothesis of Lemma 7.4, we can compute $\operatorname{Aut}(G)$ by constructing $\operatorname{Aut}(G/R)$ and $\operatorname{Aut}(G/G^{\infty})$ which is soluble, and then applying Theorem 4.6.

### Isomorphism testing

As in previous chapters, methods for computing automorphism groups naturally give rise to algorithms for testing two groups for isomorphism.

**Lemma 7.5.** *Let $G_1$, $G_2$ be finite groups. Define $R_i = O_{\infty}(G_i)$ and suppose that $G_i/R_i \cong G_i^{\infty}$ for $i = 1, 2$. Then $G_1 \cong G_2$ if and only if $R_1 \cong R_2$ and $G_1^{\infty} \cong G_2^{\infty}$.*

*Proof.* Clearly we have $G_i \cong R_i \times G_i^{\infty}$ with $R_i, G_i^{\infty} \operatorname{char} G_i$ for $i = 1, 2$, and so the result follows from Theorem 4.19. □

**Lemma 7.6.** *Let $G_1$, $G_2$ be finite groups. Define $R_i = O_{\infty}(G_i)$ and suppose that $G_i^{\infty} \neq 1$ and $R_i \cap G_i^{\infty} = 1$ for $i = 1, 2$. Let $\psi_i : G_i \to G_i/R_i \times G_i/G_i^{\infty}$ be the natural maps for $i = 1, 2$. Then $G_1 \cong G_2$ if and only if there exist isomorphisms $\Psi_R : G_1/R_1 \to G_2/R_2$ and $\Psi_{G^{\infty}} : G_1/G_1^{\infty} \to G_2/G_2^{\infty}$ such that $G_1^{\psi_1 (\Psi_R, \Psi_{G^{\infty}}) \psi_2^{-1}} = G_2$.*

*Proof.* By Lemma 7.4, we have subdirect products for each $G_i$, and we can now apply Theorem 4.20. □

## 7.2 Soluble radical complement

We now state a well know result of Schur and Zassenhaus.

**Theorem 7.7** (Schur-Zassenhaus). *Let $G$ be a group and suppose that $K \lhd G$ such that $(|K|, |G/K|) = 1$. Then $K$ has a complement in $G$. If in addition $K$ or $G/K$ is soluble, then all such complements are conjugate in $G$.*

*Proof.* See [KS04, Theorem 6.2.1] □

We note that the well-known theorem of Feit and Thompson, which states that every finite group of odd order is soluble, guarantees that at least one of $K$ or $G/K$ is soluble in the statement of Theorem 7.7.

Now suppose that $G$ is a finite group and $R = O_\infty(G)$, a soluble characteristic subgroup of $G$, with $\pi(R) \cap \pi(G/R) = \emptyset$. Then by Theorem 7.7 there exists a subgroup $H \leq G$ which is a complement of $R$ in $G$, i.e. where $H \cap R = 1$ and $G = RH$. Further, all such complements are conjugate in $G$, and so given $\alpha \in \mathrm{Aut}(G)$, $H^\alpha = H^g$ for some $g \in G$. Thus automorphisms of $G$ act on the set of all complements of $R$ by conjugation, and $\mathrm{Aut}(G)$ is determined by restriction to the action on $R$ and $H$ (recall Definition 4.2).

**Lemma 7.8.** *Let $G$ be a finite group and suppose that $R = O_\infty(G)$ such that $\pi(R) \cap \pi(G/R) = \emptyset$. Then there exists a complement $H$ to $R$ in $G$, and* $\mathrm{Aut}(G)$ *is determined by restriction to the action on $R$ and $H$.*

In this construction, we have a semidirect product $G = R \rtimes H$. Assuming $C_H(R) = 1$ and that we have an algorithm to compute $H$ from $G$ and $R$, we can use the results of Section 4.1.3 to compute $\mathrm{Aut}(G)$. Define $\Gamma = \{\alpha \in \mathrm{Aut}(G) : R^\alpha = R, H^\alpha = H\}$, then $\mathrm{Aut}(G) = \Gamma I$ where $I = \mathrm{Inn}(G)$. Use the automorphism group algorithm from Chapter 5 to construct $\mathrm{Aut}(R)$. To find automorphisms of $R$ fixing $H$ we find the normaliser

$$\Delta = N_{\mathrm{Aut}(R)}(\overline{H}|_R) \tag{7.1}$$

and can then extend each of the automorphisms $\delta \in \Delta$ to automorphisms of $G$ fixing $H$ using Lemma 4.11. We have now constructed $\Gamma$, and after adding inner automorphisms, $\mathrm{Aut}(G)$.

We note that if $C_H(R) > 1$, then $C_H(R)^\infty \neq 1$, and we then construct a subdirect product $\psi : G \to G/R \times G/C_H(R)^\infty$, and use Theorems 4.5 and 4.6 to compute $\mathrm{Aut}(G)$.

### Computing $\Delta$

To determine $\Delta$ (7.1) in practice, we will need to construct a manageable representation of $\mathrm{Aut}(R)$. As discussed in earlier chapters, computing representations of large automorphism groups is often impractical, particularly if the underlying group, in this case $R$, is large. Assuming that we don't already have a PC representation of $\mathrm{Aut}(R)$ (see Section 5.5), we can avoid still constructing a permutation representation of the full automorphism group. Since we know the structure of $\mathrm{Aut}(R)$ for each particular sub-case of the algorithm (recall the summary given in Section 5.4), we can use this information to construct $\Delta$. Define $S, T$ and $K$ as in Notation (†) from Chapter 5, taking $G$ in this context to be $R$. We can now give a brief outline of how to compute $\Delta$ from $\mathrm{Aut}(R)$ using either the the direct or semidirect product sub-cases.

If $R = S \times T$, then we have $S, T \operatorname{char} R$, and hence $S, T \operatorname{char} G$. Thus we can write

$$N_{\mathrm{Aut}(G)}\left(\overline{H}|_R\right) = N_{\mathrm{Aut}(S)}\left(\overline{H}|_S\right) \times N_{\mathrm{Aut}(T)}\left(\overline{H}|_T\right), \tag{7.2}$$

and we have reduced the problem of finding a representation of $\mathrm{Aut}(R)$ into computing one for $\mathrm{Aut}(S)$ and $\mathrm{Aut}(T)$.

Alternatively, assume that $R$ is a semidirect product $R \cong S \rtimes T$ with $C_T(S) = 1$. If $S \leq N_R(H)$, then we can reduce the problem to $N_{\mathrm{Aut}(S)}(\overline{H}|_S)$ and now we only need a representation of $\mathrm{Aut}(S)$, which has already been computed in the construction of $\mathrm{Aut}(R)$. We note that $C_H(R) = 1$ implies that $C_H(S) = 1$ (Suppose that $C_H(S) \neq 1$ and let $M$ be a minimal normal subgroup of $G$ contained in $C_H(S)$. Then $[M, R] \leq M \cap R = 1$ and $C_H(R) \neq 1$).

### Isomorphism testing

Again, we note that it is straightforward to adapt these ideas for performing isomorphism testing between two groups with isomorphic soluble radicals and radical complements.

**Lemma 7.9.** *Let $G_1$ and $G_2$ be finite groups and define $R_i = O_\infty(G_i)$ for $i = 1, 2$. Suppose that $\pi(R_i) \cap \pi(G_i/R_i) = \emptyset$ and therefore there exist complements $H_i \leq G_i$ of $R_i$ for $i = 1, 2$. Further assume that $C_{H_i}(R_i) = 1$ for $i = 1, 2$. If $G_1 \cong G_2$ then there exists an isomorphism $\rho : R_1 \to R_2$ such that $(\overline{H_1}|_{R_1})^\rho = \overline{H_2}|_{R_2}$.*

*Proof.* This result is an analogy of Theorem 6.4. Assume that $\Phi : G_1 \to G_2$ is an isomorphism. Following a similar argument to Lemma 6.3, it is clear that $\Phi : R_1 \mapsto R_2$, and so $H_1^\Phi$ is a complement of $R_2$ in $G_2$. Since all complements of $R_2$ in $G_2$ are conjugate, there exists $g \in G_2$ such that $\Phi\overline{g}|_{H_1} : H_1 \to H_2$. Now define isomorphisms $\rho = \Phi\overline{g}|_{R_1} : R_1 \to R_2$ and $\iota : \Phi\overline{g}|_{H_1} : H_1 \to H_2$, and we have $\rho \sim_{G_1,G_2} \iota$ (recall Definition 4.22). Hence, by Lemma 4.26, $(\overline{h_1}|_{R_1})^\rho = \overline{h_1^\iota}|_{R_2}$ for all $h_1 \in H_1$, and the result follows. □

So to test if $G_1$ and $G_2$ are isomorphic, begin by assuming that $\rho : R_1 \to R_2$ is an isomorphism. Verify that $(\overline{H_1}|_{R_1})^\rho = \overline{H_2}|_{R_2}$ (up to multiplying $\rho$ by an element of $\mathrm{Aut}(R_2)$). If this relation holds then $G_1 \cong G_2$ by Theorem 4.27 and use Lemma 4.25 and Lemma 4.26 to extend $\rho$ to an isomorphism $G_1 \to G_2$. If not, then $G_1 \ncong G_2$.

We note that in cases where $C_{H_i}(R_i) \neq 1$ for $i = 1, 2$ we can construct subdirect products as in the automorphism case, and then use Theorems 4.20 and 4.21 to test for isomorphism.

## 7.3 Extend from soluble radical

In this section we explore methods which involve computing the automorphism group of a finite group $G$ using the automorphism group of its soluble radical $R = O_\infty(G)$, but we assume that $\mathrm{hcf}(|R|, |G/R|) \neq 1$ and $G/R$ is almost simple.

The almost simple groups used in this section are assumed to have a specific structure which has been pre-computed and stored in some accessible database of groups. In particular, for each almost simple group $A$, we will store a list of pairs of Sylow subgroups $(P, Q)$ where $P \in \mathrm{Syl}_p(G)$ and $Q \in \mathrm{Syl}_q(G)$ for $p \neq q \in \pi(A)$ such that $A = \langle P, Q \rangle$, and $\mathrm{Aut}(A)$ is determined by restriction to the action on $P$ and $Q$ (recall Definition 4.2). A sample dataset is given in Table 7.1 below. In practice, each small almost simple group that has been examined has been found to have this structure.

| Group | Order | Primes | Sylow Subgroups |
|---|---|---|---|
| $A_5$ | 60 | $\{2,3\}$ | $P_2 = \langle\, (1,2)(3,4), (1,3)(2,4) \,\rangle$, $P_3 = \langle\, (1,4,5) \,\rangle$ |
| | | $\{2,5\}$ | $P_2 = \langle\, (1,2)(3,4), (1,3)(2,4) \,\rangle$<br>$P_5 = \langle\, (1,5,3,4,2) \,\rangle$ |
| $S_5$ | 120 | $\{2,3\}$ | $P_2 = \langle\, (1,3)(2,4), (1,2) \,\rangle$, $P_3 = \langle\, (1,5,2) \,\rangle$ |
| | | $\{2,5\}$ | $P_2 = \langle\, (1,3)(2,4), (1,2) \,\rangle$, $P_5 = \langle\, (1,4,5,3,2) \,\rangle$ |
| $L_2(7)$ | 168 | $\{2,3\}$ | $P_2 = \langle\, (1,2)(3,8)(4,7)(5,6),\ (1,4)(2,3)(5,8)(6,7), (1,5)(2,6)(3,7)(4,8) \rangle$<br>$P_3 = \langle\, (1,8,7)(3,4,5) \,\rangle$ |
| | | $\{2,7\}$ | $P_2 = \langle\, (1,2)(3,8)(4,7)(5,6),\ (1,4)(2,3)(5,8)(6,7), (1,5)(2,6)(3,7)(4,8) \rangle$<br>$P_7 = \langle\, (1,8,4,2,6,5,7) \,\rangle$ |
| | | $\{3,7\}$ | $P_3 = \langle\, (1,8,7)(3,4,5) \,\rangle$<br>$P_7 = \langle\, (1,8,4,2,6,5,7) \,\rangle$ |
| $A_6$ | 360 | $\{2,3\}$ | $P_2 = \langle\, (1,2)(3,4), (3,5)(4,6) \,\rangle$<br>$P_3 = \langle\, (1,3,4), (2,5,6) \,\rangle$ |
| | | $\{2,5\}$ | $P_2 = \langle\, (1,2)(3,4), (3,5)(4,6) \,\rangle$<br>$P_5 = \langle\, (1,6,3,4,5) \,\rangle$ |

Table 7.1: Catalogue of some small almost simple groups with appropriate $(p, q)$ values and Sylow $p, q$-subgroups.

We now describe the general setup in the following hypothesis which will assumed, unless otherwise stated, for the remainder of this section.

**Hypothesis 7.10.** Let $G$ be a finite group with $R = O_\infty(G)$, define $\phi : G \to G/R$ to be the natural mapping, and suppose that $G^\phi$ is almost simple. Let $P \in \mathrm{Syl}_p(G)$, $L = O_p(G)$, $Q \in \mathrm{Syl}_q(G)$ where $p \in \pi(R) \cap \pi(G^\phi)$, $q \in \pi(G^\phi)$ such that $G^\phi = \langle P^\phi, Q^\phi \rangle$ and assume that $\mathrm{Aut}(G^\phi)$ is determined by restriction to the action on $P^\phi$

and $Q^{\phi}$. In addition, we require that $R < N_G(P)$ and $C_Q(L) = 1$.

We note that $P \cap R \in \mathrm{Syl}_p(R)$, and since $R$ normalises $P$ we have $P \cap R = O_p(R)$. Further, since $O_p(G) \leq R$, we must have $O_p(R) = O_p(G)$, and hence $P \cap R = L$.

**Lemma 7.11.** *Assume Hypothesis 7.10. Then* $\mathrm{Aut}(G)$ *is determined by restriction to the action on $P$ and $Q$.*

*Proof.* Take $\alpha \in \mathrm{Aut}(G)$. Then there exists an element $g \in G$ such that $\alpha_P = \alpha\overline{g^{-1}}$ fixes $P$ (Lemma 2.17). Note that every automorphism $\gamma \in \mathrm{Aut}(G)$ induces some automorphism $\widetilde{\gamma} \in \mathrm{Aut}(G^{\phi})$, where $\widetilde{\gamma} = \phi^{-1}\gamma\phi$. In particular, $\widetilde{\alpha_P}$ fixes $P^{\phi}$, and since $G^{\phi}$ is determined by restriction to the action on $P^{\phi}$ and $Q^{\phi}$, there exists some $x \in G$ such that $\overline{x^{\phi}}$ fixes $P^{\phi}$ and $\widetilde{\alpha_P}\overline{(x^{\phi})^{-1}}$ fixes $Q^{\phi}$. Thus $\alpha_P\overline{x^{-1}} : Q \mapsto r^{-1}Qr$ for some $r \in R$. Then $\alpha_P\overline{x^{-1}r^{-1}}$ fixes both $P$ and $Q$. □

We begin by defining some notation which will provide a useful shorthand in the calculations that follow.

**Definition 7.12.** Let $G$ be and group and suppose that $C \leq G$ is a characteristic subgroup of $G$. Define $\mathrm{Aut}(C)|^{\mathrm{Aut}(G)} = \{\alpha|_C : \alpha \in \mathrm{Aut}(G)\}$, i.e. the subgroup of $\mathrm{Aut}(C)$ which is induced by elements of $\mathrm{Aut}(G)$.

**Lemma 7.13.** *Let $G$ be a finite group with $C \,\mathrm{char}\, G$ and let $\phi : G \to G/C$ be the natural mapping. Suppose that $G^{\phi} = \langle A^{\phi}, B^{\phi} \rangle$ for some subgroups $A, B \leq G$. Then $G = \langle A, B, C \rangle$.*

We will now outline a procedure to construct the automorphism group for a finite group $G$ which satisfies Hypothesis 7.10, and one further requirement: $\mathrm{Aut}(R)$ can be computed from $\mathrm{Aut}(L)$ using the methods of Chapter 5.

Since $\mathrm{Aut}(G)$ is determined by the action restricted to $P$ and $Q$, it is sufficient to construct $\Gamma = \{\alpha \in \mathrm{Aut}(G) : P^{\alpha} = P, Q^{\alpha} = Q\}$; and by Lemma 7.13 above, to construct automorphisms of $G$ it is sufficient to define the action of automorphism generators on $R$, $P$ and $Q$. Thus we proceed as follows:

1. Compute $\mathrm{Aut}(P)$.

2. Compute the mapping $\Phi : \mathrm{Aut}(P)_L \to \mathrm{Aut}(L)$, where $\Phi : \alpha \mapsto \alpha|_L$.

3. Find the subgroup of $\mathrm{Aut}(P)_L$ whose elements extend to automorphisms of $\langle P, Q\rangle$. Conceptually, we want to find the set of pairs

$$A_{\langle P,Q\rangle} = \{(\alpha_P, \alpha_Q) \in \mathrm{Aut}(P)_L \times \mathrm{Aut}(Q) : \alpha_P \sim_\Gamma \alpha_Q\}\,.$$

4. Using $\mathrm{Aut}(L)$, compute $\mathrm{Aut}(R)$, and maintain the correspondence between automorphisms of $L$ and the automorphisms of $R$ which they extend to. Thus we have:

$$A_{\langle P,R\rangle} = \{(\alpha_P, \alpha_R) \in \mathrm{Aut}(P)_L \times \mathrm{Aut}(R) : \alpha_P \sim_\Gamma \alpha_R\}\,.$$

5. Construct the tuples

$$A_{\langle R,P,Q\rangle} = \{(\alpha_P, \alpha_Q, \alpha_R) \in \mathrm{Aut}(P)_L \times \mathrm{Aut}(Q) \times \mathrm{Aut}(R) : \alpha_P \sim_\Gamma \alpha_Q, \alpha_P \sim_\Gamma \alpha_R\}\,,$$

and perform a search to find those tuples which define automorphisms of $G$.

We note that our assumptions at this point give the following important consequences for $\mathrm{Aut}(G)$, and in particular $\Gamma$: any $\alpha \in \Gamma$ fixes $R$ and $L$, since they are both characteristic subgroups of $G$, and by definition, $\alpha$ fixes $P$ and $Q$. Thus any $\alpha \in \Gamma$ has the property that $\alpha|_{\langle P,Q\rangle} \in \mathrm{Aut}(\langle P, Q\rangle)$. Clearly, $\mathrm{Aut}(\langle P, Q\rangle)$ is determined by restriction of the action on $P$ and $Q$, so define

$$\Gamma_{\langle P,Q\rangle} := \{\alpha \in \mathrm{Aut}(\langle P, Q\rangle) : P^\alpha = P, Q^\alpha = Q\}\,.$$

Further, we now claim that $L = O_p(\langle P, Q\rangle)$. Observe that since $G^\phi = \langle P^\phi, Q^\phi\rangle$ and $G^\phi$ is almost simple, we have $O_p(\langle P^\phi, Q^\phi\rangle) = 1$. Hence $O_p(\langle P, Q\rangle) \leq R$ and the result follows since $L = P \cap R = O_p(R)$.

Therefore we begin by computing $\mathrm{Aut}(P)$ and constructing the mapping

$$\Phi : \mathrm{Aut}(P)_L \to \mathrm{Aut}(L),$$

which is defined to be restriction to $L$. Using the results of Section 4.1.3, there exists a monomorphism $\phi : \Gamma_{\langle P,Q\rangle} \to \Delta'_{\langle P,Q\rangle}$, where $\phi : \alpha \mapsto \alpha|_L$ and

$$\Delta'_{\langle P,Q\rangle} = \left\{\alpha \in \mathrm{Aut}(P)_L : \alpha|_L \in N_{\mathrm{Aut}(L)}(\overline{Q}|_L), N_P(Q)^\alpha = N_P(Q)\right\},$$

and so for each $\alpha_P \in \Delta'_{\langle P,Q\rangle}$ we can attempt to construct $\alpha_Q \in \mathrm{Aut}(Q)$ using Lemma 4.12, such that $\alpha_P \sim_{\Gamma_{\langle P,Q\rangle}} \alpha_Q$. Hence we have constructed $\Gamma_{\langle P,Q\rangle}$, and thus $A_{\langle P,Q\rangle}$.

Now, since $R \,\mathrm{char}\, G$, any automorphism $\alpha \in \mathrm{Aut}(R)$ induces some $\alpha|_R \in \mathrm{Aut}(R)$. In addition, if we define $\Gamma_P = \{\alpha \in \mathrm{Aut}(G) : P^\alpha = P\}$, then a first step in constructing $A_{\langle R,P\rangle}$ is to find $\mathrm{Aut}(R)|^{\Gamma_P}$. Any automorphism of $P$ which extends to an automorphism of $G$, must fix $L$. Since we can compute $\mathrm{Aut}(R)$ from $\mathrm{Aut}(L)$, $\mathrm{Im}\,\Phi$ will extend to $\mathrm{Aut}(R)|^{\Gamma_P}$, i.e. automorphisms of $R$ which are induced by automorphisms of $G$ which fix $P$. In particular, there exists a subgroup

$$\Delta_{\langle R,P\rangle} \leq \mathrm{Im}\,\Phi$$

such that if $\delta \in \Delta_{\langle R,P\rangle}$ then there exists a set

$$A_R(\delta) = \left\{\rho \in \mathrm{Aut}(R)|^{\Gamma_P} : \rho|_L = \delta\right\},$$

and given such $\delta$, we can determine $A_R(\delta) \subseteq \mathrm{Aut}(R)$ (as described in the summary in Section 5.4). Now we have

$$A_{\langle R,P\rangle} = \left\{(\alpha_P, \alpha_R) : \alpha_P \in \Phi^{-1}(\Delta_{\langle R,P\rangle}), \alpha_R \in A_R(\alpha_P^{\Phi})\right\}.$$

Following step 5 is just a question of performing a search to match the automorphisms of $P$. Finally, another search is done on $A_{\langle R,P,Q\rangle}$ to determine which tuples construct automorphisms of $G$. Thus we have constructed $\Gamma$, and hence $\mathrm{Aut}(G)$.

# Chapter 8

# Benchmarks

In this chapter we present a collection of benchmarks for our implementation of AutomorphismGroupSolubleGroup (see Algorithm 5.5) in the MAGMA language. To give an indication of the relative performance of our algorithm, we have also included timings of the other algorithms currently available in MAGMA [BCP97].

We use $A_P$ to refer to the MAGMA language implementation of the general finite group automorphism algorithm (used here just for permutation groups) described in [CH03], $A_{Sol}$ refers to the MAGMA 'C' implementation of the soluble group automorphism group algorithm [Smi94], and $A_N$ is the MAGMA language implementation of the algorithm described in this thesis. Thus, when referring to $A_N$ applied to a finite soluble group $G$, the Sylow $p$-group $P \in \mathrm{Syl}_p(G)$ will be group whose automorphism group has been computed as part of the computation of $\mathrm{Aut}(G)$. If more than one such automorphism group has been constructed, we give the largest. Given an algorithm $A$, we use $t(A)$ to denote the time, in seconds, taken for it to complete the calculation. Finally we define our notation for groups: $G_i$ refers to the groups that are found in the `solgps` package of MAGMA, $(S_3)^n = \times_{i=1}^{n} S_3$ where $S_3 = \mathrm{Sym}(3)$, $D_m$ is the dihedral group of order $2m$, $T_{m,n}$ denotes the $n$th transitive group of degree $m$ (`TransitiveGroup(m, n)` in MAGMA) and $P_{m,n}$ denotes the $n$th primitive group of degree $m$(`PrimitiveGroup(m, n)` in MAGMA).

## 8.1 Some Large Examples

In Table 8.1 we give a collection of miscellaneous finite soluble groups together with timings for the construction of their respective automorphism groups using $A_P$, $A_{Sol}$ and $A_N$.

We have deliberately included timings for $(S_3)^n$ for increasing values of $n$ as they were previously highlighted in [CH03] as being soluble groups which are particularly hard examples for both the soluble group and permutation group automorphism algorithms. We also include timings for $D_6^5$ and $D_12^5$, which are particularly tricky for $A_P$ and $A_{Sol}$, but finish in reasonable time on $A_N$.

It should be noted that there are two large examples in the Magma `solgps` package for which the $A_{PC}$ algorithm vastly outperforms $A_N$, and these are $G_1$ and $B(2, 6)$. In the case of $G_1$ the $p$-group $P$ chosen is a 13-group and has a very large Frattini quotient ($|P/\Phi(P)| = 13^{14}$). Hence the search through $\mathrm{Aut}(P/\Phi(P)) \cong \mathrm{GL}(14, 13)$ for automorphisms which lift to the next layer is very hard, and the process stalls here. The Burnside group $|B(2, 6)| = 2^{28}3^{25}$ can be tackled with $p = 2$ or 3, but both hit stumbling blocks. For $p = 3$ the automorphism group computation of the 3-group is quick, but not soluble, and constructing a permutation representation stalls ($|\mathrm{Aut}(P)| = 2^{10}3^{110}5 \cdot 11^2 13$). With $p = 2$, the calculation reduces to finding a normaliser in $\mathrm{GL}(28, 2)$, which is equally hard!

When an algorithm has been set running with a time limit of $n$ seconds, and did not finish, we use $n+$. There are quite a number of examples with $n = 3600$, so we use - instead of 3600+.

## 8.2 Small Groups

The small groups database of order up to 2000 [BEO02] provides a huge range of convenient test examples. We have chosen some (relatively) large orders from the database and timed our algorithm on 10,000 randomly selected examples for each order (except orders 1458 and 1701 which have 1798 and 309 groups respectively, so we run through each in turn). Our timings are given in Table 8.2. Completing a similar benchmark using the soluble group automorphism group algorithm ($A_{Sol}$) in MAGMA was attempted but could not be completed, due to the huge number of groups which take longer than 5 minutes to finish. In a brief test using the same groups timed on our algorithm, we found that: 1112 out of the first 1293 groups of order 1280; 1031 out of the first 1679 groups of order 1920; and 1138 out of the first 1151 groups of order 1536, each took longer than 5 minutes to complete.

## 8.3 Transitive Groups

We also include a table of timings for some groups from the transitive group database (see Table 8.3). Having performed runs over different degrees, we have found several hundred large groups which are hard examples for the current algorithms. We include a varied selection here, with timings for both $A_P$ and $A_N$. In particular, there are over are over 150 such groups of degree 24, most of which have order $2^{12} \cdot 3$ or $2^{11} \cdot 3$, with timings for $A_P$ ranging from 5 minutes to 30 hours. These large examples demonstrate the advantages of using $A_N$ over $A_P$ for groups with large Sylow subgroups that have relatively easy $p$-group automorphism calculations.

It should be noted that our profiling also picked up several groups where both the $A_P$ and $A_N$ perform poorly. Examples from the database of degree 30 tend to produce $p$-groups with large Frattini quotients, and so the first lifting step of the $p$-group automorphism algorithm is very hard.

We make further comments about some individual examples:

- We note that the timing of $T_{28,1583}$ was split into 49s for computing the $p$-group automorphism group, and then 120.0s computing a permutation group for the $p$-Core automorphism group.

- We have deliberately included timings for $T_{20,846}$ even though $A_N$ doesn't show much speed improvement over $A_P$. In fact, $\mathrm{Aut}(T_{20,846})$ is soluble and $A_N$ automatically constructs a PC presentation for it. Computing a similar presentation from the output of $A_P$ requires the construction of a permutation representation, which adds a further 40s to the calculation in this case.

| $G$ | $\|G\|$ | $p$ | $\|\mathrm{Aut}(P)\|$ | $\|\mathrm{Aut}(G)\|$ | $t(A_P)$ | $t(A_{Sol})$ | $t(A_N)$ |
|---|---|---|---|---|---|---|---|
| $(S_3)^5$ | $2^5 3^5$ | 3 | $2^{10}3^{10}5 \cdot 11^2 13$ | $2^8 3^6 5$ | 11.54 | 7651.32 | 0.03 |
| $(S_3)^6$ | $2^6 3^6$ | 3 | $2^{13}3^{15}5 \cdot 7 \cdot 11^2 13^2$ | $2^{10}3^8 5$ | 3315.96 | - | 0.04 |
| $(S_3)^7$ | $2^7 3^7$ | 3 | $2^{14}3^{21}5 \cdot 7 \cdot 11^2 13^2 1093$ | $2^{11}3^9 5 \cdot 7$ | - | - | 0.07 |
| $(S_3)^8$ | $2^8 3^8$ | 3 | $2^{19}3^{28}5^2 7 \cdot 11^2 13^2 41 \cdot 1093$ | $2^{15}3^{10}5 \cdot 7$ | - | - | 0.10 |
| $(S_3)^9$ | $2^9 3^9$ | 3 | $2^{20}3^{36}5^2 7 \cdot 11^2 13^3 41 \cdot 757 \cdot 1093$ | $2^{16}3^{13}5 \cdot 7$ | - | - | 0.16 |
| $(S_3)^{10}$ | $2^{10}3^{10}$ | 3 | $2^{23}3^{45}5^2 7 \cdot 11^4 13^3 41 \cdot 61 \cdot 757 \cdot 1093$ | $2^{18}3^{14}5^2 7$ | - | - | 0.22 |
| $(S_3)^{20}$ | $2^{20}3^{20}$ | 3 | $2^{48}3^{190}5^6 7^3 11^8 13^6 \ldots 797161$ | $2^{38}3^{28}5^4 7^2 11 \cdot 13 \cdot 17 \cdot 19$ | - | - | 3.45 |
| $(D_6)^5$ | $2^{10}3^5$ | 2 | $2^{10}3^{10}5 \cdot 11^2 13$ | $2^{43}3^8 5^2 7 \cdot 31$ | 1697.86 | - | 1.94 |
| $(D_{12})^5$ | $2^{15}3^5$ | 2 | $2^{10}3^{10}5 \cdot 11^2 13$ | $2^{63}3^6 5$ | 1800+ | - | 21.420 |
| $P_{625,84}$ | $2^6 5^4$ | 5 | $2^{11}3^2 5^6 13 \cdot 31$ | $2^{10}3^2 5^5$ | 28.51 | 1800+ | 0.44 |
| $P_{625,375}$ | $2^9 5^4$ | 5 | $2^{11}3^2 5^6 13 \cdot 31$ | $2^{11}3 \cdot 5^4$ | 247.48 | 1800+ | 0.36 |
| $G_7$ | $2^{11} \cdot 11 \cdot 23 \cdot 89$ | 2 | $2^{55}3^6 5^2 7^3 11 \cdot 17 \cdot 23 \cdot 31^2 73 \cdot 89 \cdot 127$ | $2^{11}11 \cdot 23 \cdot 89$ | N/A | 0.50 | 0.12 |
| $G_8$ | $3^{10}2^4 5 \cdot 11^2 61$ | 3 | $2^{23}3^{45}5^2 7 \cdot 11^4 13^3 41 \cdot 61 \cdot 757 \cdot 1093$ | $2^4 3^{10}5 \cdot 11^2 61$ | N/A | 65.98 | 10.99 |
| $G_9$ | $3^{10}2^5$ | 3 | $2^5 3^{10}$ (largest) | $2^9 3^{10}$ | N/A | 786.50 | 0.56 |
| $G_{10}$ | $2^{18}7^4$ | 2 | $2^{37}3 \cdot 7^3$ (largest) | $2^{20}3^2 7^6$ | N/A | 18000+ | 18.480 |
| $G_{11}$ | $2^{13}13 \cdot 8191$ | 2 | $2^{78}3^8 5^3 7^4 11 \cdot 13 \cdot 17 \cdot 23 \cdot 31^2 73 \cdot 89 \cdot 127 \cdot 8191$ | $2^{13}13 \cdot 8191$ | N/A | 7.99 | 0.72 |

Table 8.1: Miscellaneous collection of finite soluble groups with timings for $A_P$, $A_{Sol}$ and $A_N$

| $\|G\|$ | No. Tested | $\Sigma t(A_N)$ | Avg $t(A_N)$ | Med $t(A_N)$ | Max $t(A_N)$ | Over 1s |
|---|---|---|---|---|---|---|
| 384 | 10000 | 1016.97 | 0.1017 | 0.1 | 0.48 | 0 |
| 768 | 10000 | 1752.39 | 0.1752 | 0.16 | 2.37 | 3 |
| 1458 | 1798 | 173.13 | 0.0962 | 0.09 | 0.60 | 0 |
| 1536 | 10000 | 4422.41 | 0.4422 | 0.35 | 3.95 | 495 |
| 1701 | 309 | 19.89 | 0.064 | 0.06 | 0.19 | 0 |
| 1792 | 10000 | 1816.03 | 0.182 | 0.16 | 2.54 | 6 |
| 1920 | 10000 | 1562.82 | 0.157 | 0.15 | 0.97 | 0 |

Table 8.2: Timings for random collections of small groups

| $G$ | $\|G\|$ | $p$ | $\|\operatorname{Aut}(P)\|$ | $\|\operatorname{Aut}(G)\|$ | $t(A_P)$ | $t(A_N)$ |
|---|---|---|---|---|---|---|
| $T_{20,893}$ | $2^9 5^4$ | 5 | $2^{11}3^2 5^6 13 \cdot 31$ | $2^{11}3 \cdot 5^4$ | 206.54 | 0.38 |
| $T_{20,846}$ | $2^{15}5$ | 2 | $2^{58}3 \cdot 5$ | $2^{25}5$ | 51.02 | 21.89 |
| $T_{24,10398}$ | $2^{12}3$ | 2 | $2^{31}3$ | $2^{23}3$ | 28269.24 | 8.57 |
| $T_{24,12814}$ | $2^{13}3$ | 2 | $2^{40}3$ | $2^{22}3$ | 9722.93 | 12.31 |
| $T_{24,18679}$ | $2^{16}3$ | 2 | $2^{33}3^4$ | $2^{23}3^3$ | 28016.08 | 6.87 |
| $T_{24,21854}$ | $2^{19}3$ | 2 | $2^{52}3$ | $2^{30}3$ | 301.28 | 8.49 |
| $\operatorname{Aut}(T_{24,15000})$ | $2^{19}3$ | 2 | $2^{48}$ | $2^{39}3$ | 43200+ | 9.85 |
| $T_{24,22800}$ | $2^{20}3$ | 2 | $2^{40}3$ | $2^{28}3$ | 305.52 | 13.32 |
| $T_{24,23435}$ | $2^{21}3$ | 2 | $2^{46}3$ | $2^{30}3$ | 2121.17 | 13.26 |
| $T_{24,22267}$ | $2^8 3^8$ | 3 | $2^{19}3^{28}5^2 7 \cdot 11^2 13^2 41 \cdot 1093$ | $2^{14}3^9$ | 65108.53 | 0.08 |
| $T_{24,23531}$ | $2^{10}3^8$ | 3 | $2^{19}3^{28}5^2 7 \cdot 11^2 13^2 41 \cdot 1093$ | $2^{15}3^9$ | 75460.35 | 11.74 |
| $T_{24,23532}$ | $2^{10}3^8$ | 3 | $2^{19}3^{28}5^2 7 \cdot 11^2 13^2 41 \cdot 1093$ | $2^{14}3^9 7$ | 31081.23 | 6.25 |
| $T_{24,23995}$ | $2^{11}3^8$ | 3 | $2^{19}3^{28}5^2 7 \cdot 11^2 13^2 41 \cdot 1093$ | $2^{13}3^8$ | 302.59 | 1.02 |
| $T_{24,24304}$ | $2^{12}3^8$ | 3 | $2^{19}3^{28}5^2 7 \cdot 11^2 13^2 41 \cdot 1093$ | $2^{16}3^9$ | 12500.03 | 17.9 |
| $T_{27,894}$ | $2 \cdot 3^8$ | 3 | $2^5 3^{13}$ | $2^5 3^{14}$ | 83556.90 | 1.19 |
| $T_{28,1583}$ | $2^{22}7$ | 2 | $2^{68}3$ | $2^{29}3 \cdot 7$ | 15136.84 | 187.51 |
| $T_{30,4436}$ | $2^{12}3 \cdot 5^6$ | 5 | $2^{16}3^4 5^{15} 7 \cdot 11 \cdot 13 \cdot 31^2 71$ | $2^{14}3^2 5^6$ | 1114.23 | 3.25 |

Table 8.3: Timings for some transitive groups

# Bibliography

[BCP97] W. Bosma, J. J. Cannon, and C. Playoust. The Magma algebra system I. the user language. *Journal of Symbolic Computation*, 24:235–265, 1997.

[BEO02] H. U. Besche, B. Eick, and E. A. O'Brien. A millennium project: Constructing small groups. *Intern. J. Alg. and Comput*, 12:623–644, 2002.

[BM83] G. Butler and J. McKay. The transitive groups of degree up to 11. *Communications in Algebra*, 11:863–911, 1983.

[But93] G. Butler. The transitive groups of degree fouteen and fifteen. *Journal of Symbolic Computation*, 16:413–422, 1993.

[CH03] J. J. Cannon and D. F. Holt. Automorphism group computation and isomorphism testing in finite groups. *Journal of Symbolic Computation*, 35:241–267, 2003.

[CH08] J. J. Cannon and D. F. Holt. The transitive permutation groups of degree 32. *Experiment. Math.*, 17:307–317, 2008.

[Cou11] H. Coutts. *Topics in Computational Group Theory: Primitive permutation groups and matrix group normalisers.* PhD thesis, St. Andrews, 2011.

[CQRD11] H. Coutts, M. Quick, and C. M. Roney-Dougal. The primitive permutation groups of degree less than 4096. *Communications in Algebra*, 39(10):3526–3546, 2011.

[ELGO02] B. Eick, C. R. Leedham-Green, and E. A. O'Brien. Constructing automorphism groups of p-groups. *Communications in Algebra*, 30(5):2271–2295, 2002.

[EO09] B. Eick and E. A. O'Brien. GAP package AutPGrp: Computing the automorphism group of a p-group, 2009.

[FN67] V. Felsch and J. Neubüser. On a programme for the determination the automorphism group of a finite group. *Computation Problems in Abstract Algebra*, pages 59–60, 1967.

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

[HEO05] D. F. Holt, B. Eick, and E. A. O'Brien. *Handbook of Computational Group Theory*. Chapman Hall/CRC, 2005.

[HLGO11] D. F. Holt, C. R. Leedham-Green, and E. A. O'Brien. A new model for computation with matrix groups. *preprint*, 2011.

[Hul05] A. Hulpke. Constructing transitive permutation groups. *Journal of Symbolic Computation*, 39(1):1–30, 2005.

[Joh97] D. Johnson. *Presentations of Groups*, volume 15 of *London Mathematical Society Student Texts*. Cambridge University Press, second edition, 1997.

[KS04] H. Kurzweil and B. Stellmacher. *The Theory of Finite Groups: An Introduction*. Springer, 2004.

[O'B92] E. A. O'Brien. Computing automorphisms groups of p-groups. *Computational Algebra Number and Number Theory, Sydney*, 1992.

[Rob76] H. Robertz. *Eini Methode zur Berechnung der Automorphismen-gruppe einer endlichen Gruppe*. PhD thesis, Aachen, 1976.

[Rot95] J. J. Rotman. *An Introduction to the Theory of Groups*, volume 148 of *Graduate Texts in Mathematics*. Springer, 4th edition, 1995.

[Roy87] G. F. Royle. The transitive groups of degree twelve. *Journal of Symbolic Computation*, 4:255–268, 1987.

[Seg83] D. Segal. *Polycyclic Groups*. Cambridge University Press, 1983.

[Sim70] C. C. Sims. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 169–183. Pergamon, Oxford, 1970.

[Sim71] C. C. Sims. Computation with permutation groups. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, SYMSAC '71, pages 23–28, New York, NY, USA, 1971. ACM.

[Smi94] M. J. Smith. *Computing Automorphisms of Finite Soluble Groups*. PhD thesis, Australian National University, 1994.