

# COMPLEXITY UPPER BOUNDS USING PERMUTATION GROUP THEORY

Thesis submitted in  
partial fulfillment of the degree of  
Doctor of Philosophy (Ph.D)

by

**Piyush P Kurur**

Theoretical Computer Science Group,  
The Institute of Mathematical Sciences,  
Taramani, Chennai 600 113.

University of Madras  
Chennai 600 005

January, 2006

# Declaration

I declare that the thesis entitled *Complexity Upper Bounds using Permutation Group theory* submitted for the degree of Doctor of Philosophy is the record of the work carried out by me during *January 2003* to *January 2006* under the guidance of *V. Arvind* and has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this University or any other University or other Institution of Higher learning.

Piyush P Kurur  
Theoretical Computer Science Group,  
Institute of Mathematical Sciences,  
Taramani, Chennai 600 113.

January 13, 2006

# Certificate

I certify that the thesis entitled *Complexity Upper Bounds using Permutation Group theory* submitted for the degree of Doctor of Philosophy by *Piyush P Kurur* is the record of research carried out by him during *January 2003* to *January 2006* under my guidance and supervision, and that this work has not formed the basis for the award of any degree, diploma, associateship, fellowship or other titles in this University or any other University or Institution of higher learning.

V. Arvind  
Thesis Supervisor  
Professor, Theoretical Computer Science  
Institute of Mathematical Sciences

January 13, 2006

# Acknowledgements

During my stay at IMSc, I was fortunate to be in the company of some wonderful people who have directly or indirectly contributed to this thesis. Firstly I thank Arvind for supervising my work. It was he who introduced me to many of the fascinating topics in Computer Science. Working with him was a great experience in itself. In him I found a great teacher, a friend and a research collaborator. His contribution to this thesis is far more than what I could express in this limited space.

I had the privilege to learn Computer Science and Mathematics through some excellent lectures at IMSc. I thank the group at IMSc, especially Meena Mahajan, R. Ramanujam, Kamal Lodaya and Venkatesh Raman, for this wonderful research environment.

My visit to Germany as part of the DST-DAAD personnel program was a great learning experience for which I am indebted. I had the good fortune to enjoy the hospitality of Johannes Köbler while visiting Humboldt Universität, Berlin under this program. During this period I had a great time visiting Universität Paderborn thanks to Joachim von zur Gathen and Technische Universität Darmstadt thanks to Johannes Buchmann. My visit to Japan in December 2003 was made memorable due to the time I spent at the Tokyo Institute of Technology thanks to Osamu Watanabe.

Many individuals and organisations helped me by funding my research visits. I would like to thank the conference committees of CCC 2002, CCC 2005 (especially Eric Allender and Lance Fortnow) and ANTS VI for their funds that helped me attend these conferences. I thank the National Board of Higher Mathematics (NBHM) and the Indian Association for Research in Computer Sciences (IARCS) for funding some of my research visits.

Finally I would like to thank my friends for making the life in IMSc memorable and entertaining.

# Contents

<b>Acknowledgements</b>	<b>iii</b>
<b>Notation</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Overview of this thesis . . . . .	2
<b>2 Complexity theory</b>	<b>7</b>
2.1 Counting complexity classes . . . . .	9
<b>3 Group Theory</b>	<b>14</b>
3.1 Permutation Groups . . . . .	17
3.2 Strong generator set . . . . .	18
3.3 Transitivity, Blocks and Primitivity . . . . .	18
3.4 Structure Tree and Structure Forest . . . . .	21
<b>4 The Graph Isomorphism problem</b>	<b>24</b>
4.1 Group theoretic formulation of Graph Isomorphism problem .	25
4.2 Problems related to Graph Isomorphism . . . . .	26
4.3 Computing the lex-least element of a Coset . . . . .	28
4.4 The FINDGROUP problem . . . . .	29
4.5 The complexity of Graph Isomorphism . . . . .	32
4.6 Discussion . . . . .	33
<b>5 Bounded colour multiplicity Graph Isomorphism problem</b>	<b>34</b>
5.1 The Pointwise stabiliser problem . . . . .	35
5.2 Characteristic subgroups and Socles . . . . .	38
5.3 Residues and Residual Series . . . . .	40
5.4 Strong generator set revisited . . . . .	43
5.4.1 Computing the strong generator set . . . . .	45

5.5	The target reduction procedure . . . . .	54
5.5.1	Computing the critical orbits: abelian case . . . . .	58
5.5.2	Computing the critical orbits: nonabelian case . . . . .	62
5.6	Complexity of $\text{BCGI}_b$ . . . . .	71
5.7	Discussion . . . . .	71
<b>6</b>	<b>Computational Galois theory</b>	<b>73</b>
6.1	Galois theory . . . . .	74
6.2	Finite Fields . . . . .	76
6.3	Algebraic numbers and number fields . . . . .	76
6.3.1	Ring of Algebraic Integers . . . . .	77
6.4	Basic algorithms . . . . .	78
6.4.1	Encoding algebraic entities . . . . .	79
6.4.2	Factoring polynomials and related problems . . . . .	81
6.4.3	Algorithms for Galois group computation . . . . .	82
6.5	Some useful bounds . . . . .	85
6.6	Discussion . . . . .	87
<b>7</b>	<b>Testing nilpotence of Galois group</b>	<b>89</b>
7.1	Computing the fields $\mathbb{Q}_\Delta$ . . . . .	91
7.2	Nilpotence testing for Galois groups . . . . .	95
7.2.1	The nilpotence test . . . . .	99
7.3	$\Gamma_d$ -testing for Galois groups . . . . .	102
7.4	Discussion . . . . .	105
<b>8</b>	<b>Chebotarev density theorem and Order finding</b>	<b>106</b>
8.1	Chebotarev density theorem . . . . .	107
8.2	Computing the order of the Galois group . . . . .	110
8.3	Computing the order of Galois groups in $\Gamma_d$ . . . . .	112
8.4	Discussion . . . . .	115
<b>9</b>	<b>Computing Galois groups</b>	<b>118</b>
9.1	Computing abelian Galois groups . . . . .	119
9.2	Computing simple Galois groups . . . . .	123
9.3	Discussion . . . . .	124
	<b>Bibliography</b>	<b>125</b>
	<b>Index</b>	<b>132</b>

# Notation

$C_G(A)$	the centraliser of $A$ in $G$ , page 14.
$\text{Diag}(G_1 \times G_2)$	diagonal subgroup of $G_1 \times G_2$ , page 16.
$\text{NCL}_G(A)$	the normal closure of $A$ in $G$ , page 14.
$\text{Res}_T(G)$	for simple group $T$ the $T$ -residue of $G$ , page 42.
$\text{Soc}(G)$	Socle of $G$ , page 39.
$G \ltimes H$	semidirect product of $G$ and $H$ , page 15.
$G \times H$	direct product of $G$ and $H$ , page 15.
$\text{Sift}(g)$	The sift of $g$ , page 43.
$H \hookrightarrow G$	$H$ embeds into $G$ , page 14.
$H \leq G, G \geq H$	$H$ is a subgroup $G$ .
$H < G, G > H$	$H \leq G$ and $H \neq G$ .
$[G : H]$	the index of $H$ in $G$ for $H \leq G$ , page 14.
$H \trianglelefteq G, G \trianglerighteq H$	$H$ is a normal subgroup of $G$ .
$H \triangleleft G, G \triangleright H$	$H \trianglelefteq G$ and $H \neq G$ .
$[\Sigma : \Delta]$	index of the block $\Delta$ in $\Sigma$ , page 19.
$\alpha^g$	image of $\alpha$ under a permutation $g$ .
$\mathcal{B}(\Sigma/\Delta)$	conjugate blocks of $\Delta$ contained in $\Sigma$ , page 19.
$G(\Delta)$	pointwise stabiliser of $\Delta$ .
$\text{Sym}(\Omega)$	symmetric group on the set $\Omega$ .
$G_\Delta$	setwise stabiliser of $\Delta$ .
$S_n$	$\text{Sym}(\{1, \dots, n\})$ .
$[L : K]$	degree of the extension $L/K$ , page 74.
$\text{H}(\alpha)$	height of the algebraic number $\alpha$ , page 77.
$\mathbb{O}_K$	ring of algebraic integers of number field $K$ .
$\mathbb{F}_{p^r}$	unique finite field of cardinality $p^r$ .
$\mathbb{Q}, \mathbb{R}, \mathbb{C}$	field of rational, real and complex numbers respectively.
$\text{N}(\mathfrak{a}), \text{N}(\alpha)$	norm of the ideal $\mathfrak{a}$ and $\alpha \mathbb{O}_K$ respectively, page 78.
$d_K$	discriminant of the number field $K$ .
$K[X]$	polynomials in $X$ with coefficients from $K$ .

$K_f$                     splitting field of the polynomial  $f(X)$  over  $K$ , page 75.  
 $\text{Fix}(L, G)$             fixed field of  $L$  under  $G$ , page 76.

# Chapter 1

## Introduction

Considerable progress has been made recently in the design of efficient algorithms for computational problems in permutation group theory. Many of these results exploit the structure of permutation groups. As permutation groups arise naturally in many computational problems, algorithmic breakthrough in this area often led to progress in solving, at least partially, other computational problems; Graph Isomorphism being a striking example. It is reasonable to expect that these group-theoretic and algorithmic advances would lead to better insights into the complexity of computational problems which are connected to permutation group theory. In this thesis we study Graph Isomorphism and problems that arise in Galois theory. Our aim is to use the structural properties of permutation groups together with other algebraic techniques to prove complexity upper bounds.

Complexity theory is the study of resource bounded computations. Efficiency is measured in terms of the resource required to solve the problem as a function of the input size. Two of the most important measures are the time and space required to solve the problem on a Turing machine. Often problems have a trivial exponential time brute force algorithm that searches for a potential solution in the set of all possible solutions. Such exponential time algorithms are impractical as they take considerable time for solving instances of reasonable sizes. Following the suggestion of Edmonds [25] it is widely accepted that computational problems in P, i.e. problems that are solvable in polynomial time on a deterministic Turing machine, are those that are tractable. This assumption is called the *extended Church-Turing hypothesis*. The complexity class NP is the class of problems that can be solved on a nondeterministic Turing machine in polynomial time. It is exactly the class of decision problems for which yes instances have polynomial

time verifiable certificates. Clearly NP contains the class P but whether this containment is strict is a central open problem in complexity theory.

An important concept in complexity theory is the notion of completeness. A problem  $P$  in a complexity class  $\mathcal{C}$  is said to be complete for  $\mathcal{C}$  if for any other  $P'$  in  $\mathcal{C}$  instance of  $P'$  can efficiently reduced to  $P$ . Problems complete for a complexity class  $\mathcal{C}$  are in some sense the hardest problems of  $\mathcal{C}$ . For the class NP starting with the work of Cook [23] and Levin [45] and subsequently Karp [31] many important computational problems have been show to be complete (see the book of Garey and Johnson [29]). If any of these problems have polynomial time algorithm then  $P = NP$ . Hence a problem being NP-complete is a strong evidence that it has no efficient (i.e. polynomial time) algorithm.

Classifying natural problems by showing it to be complete for a complexity class is an important goal in complexity theory. For a computational problem, proving complexity theoretic upper and lower bounds often requires novel insights into the mathematics underlying the problem. The tight classification of the complexity of the permanent [70], determinant [66, 72] are classic examples. Despite serious efforts many problems still elude such a tight classification.

In this thesis we study the complexity Graph Isomorphism and problems associated with Galois theory with an aim of classifying these in the frame work of complexity theory. A common thread that connects these two is the role of permutation group theory. The structure of permutation groups and the numerous efficient algorithms for permutation group problems play an important role in our results.

Permutation groups, apart from being a source of interesting computational problems, have played important role in algorithms for Graph Isomorphism like for example in the polynomial time algorithm of Luks [46] for bounded valence graphs. Group theory has played important role in various complexity theoretic results. Babai's [10]  $AM \cap co-AM$  upper bounds for matrix group problems and Barrington's [15] group theoretic characterisation of  $NC^1$  are two classic examples.

## 1.1 Overview of this thesis

We now give an overview of the thesis. Chapter 2 is a brief survey of the complexity theory required for this thesis and Chapter 3 develops the required group theory. For our results on Galois theory we need some results from algebraic number theory. We describe these in Chapter 6. Our results

on Graph Isomorphism and related problems are explained in Chapters 4 and 5. We describe our results on computational problems in Galois theory in Chapters 7, 8 and 9.

## Graph Isomorphism

Given two undirected graphs  $X_1 = (V_1, E_1)$  and  $X_2 = (V_2, E_2)$  the Graph Isomorphism problem is to check whether  $X_1$  and  $X_2$  are isomorphic, i.e. to check whether there is a one-to-one map  $f : V_1 \rightarrow V_2$  such that for every unordered pair  $\{u, v\}$  from  $V_1$ ,  $\{u, v\} \in E_1$  if and only if  $\{f(u), f(v)\} \in E_2$ . In this thesis we also study a special case of Graph Isomorphism problem called the *bounded colour multiplicity Graph Isomorphism problem*, BCGI for short. Given two vertex-coloured graphs  $X_1$  and  $X_2$  such that the number of vertices with a given colour is less than a constant  $b$ , we want to check whether there is a colour preserving isomorphism, i.e. an isomorphism  $f$  from  $X_1$  to  $X_2$  such that  $u \in V(X_1)$  and  $f(u) \in V(X_2)$  are of the same colour. We call this problem *bounded colour multiplicity graph isomorphism problem*, BCGI $_b$  for short.

In Chapter 4 we show that the Graph Isomorphism problem is in the complexity class SPP. In fact we prove a more general result: We show that the generic group theoretic problem FINDGROUP is in the complexity class  $\text{FP}^{\text{SPP}}$ . As a consequence many interesting problems in permutation group theory like Graph Isomorphism, Set stabiliser problem and the Hidden subgroup problem over permutation groups are in SPP (or  $\text{FP}^{\text{SPP}}$  for functional problems). Computational problem in SPP (or  $\text{FP}^{\text{SPP}}$  in case they are functional problems) are *low* for many important complexity classes like  $\oplus\text{P}$  (in fact  $\text{Mod}_k\text{P}$  for all  $k$ ),  $\text{C}=\text{P}$  etc. Hence by proving the Graph Isomorphism problem to be in SPP we have show it to be low for each of these classes. Earlier it was not even know whether GI was in  $\oplus\text{P}$ .

In Chapter 5 we prove that BCGI $_b$  is in the  $\text{Mod}_k\text{L}$ -hierarchy where the constant  $k$  and the level of the hierarchy depends only on  $b$ . Recently Torán [68] has shown the Graph Isomorphism problem to be hard for various complexity classes. In particular he has proved that BCGI is hard for  $\text{Mod}_k\text{L}$  for all  $k$ . The graph gadgets that he construct can be used to show the hardness of BCGI for the entire  $\text{Mod}_k\text{L}$ -hierarchy [8, Appendix], a stronger result. Our result on BCGI complements his result and gives a fairly tight classification of BCGI in terms of logspace counting classes.

Another consequence of our result is on the parallel complexity of BCGI. Our results improve the NC upper bound of Luks [47] to  $\text{NC}^2$  (even  $\text{TC}^1$ ).

## Galois theory

Consider a number field  $K$ , a field extension of  $\mathbb{Q}$  the field of rational numbers. The Galois group of  $K$ , denoted by  $\text{Gal}(K/\mathbb{Q})$ , is the group of *field automorphisms* of  $K$  that when restricted to  $\mathbb{Q}$  is identity. For a polynomial  $f(X) \in \mathbb{Q}[X]$ , the splitting field  $\mathbb{Q}_f$  is the smallest extension of  $\mathbb{Q}$  that contains all the roots of  $f$ . By the Galois group of  $f$  we mean the Galois group  $\text{Gal}(\mathbb{Q}_f/\mathbb{Q})$ .

The Galois group of a degree  $d$  polynomial  $f$  can be thought of as a subgroup of  $S_d$ , the group of permutations on  $d$  objects. This follows from the fact that the Galois group of  $f$  is fully specified by giving its action on the roots of  $f$ .

Computing the Galois group of a polynomial is a fundamental problem in algorithmic number theory. Often one is interested in verifying whether the Galois group of a polynomial satisfies certain properties instead of actually computing the Galois group. Asymptotically, the best algorithm for computing the Galois group of a polynomial  $f(X) \in \mathbb{Q}[X]$  is due to Landau [37] and runs in time polynomial in size ( $f$ ) and the order of the Galois group of  $f$ . Since the Galois group of a polynomial  $f(X)$  of degree  $n$  can have  $n!$  elements, Landau's algorithm takes exponential-time in the worst case.

Besides being a natural computational problem, knowing the Galois group of a polynomial  $f$  or knowing certain properties of the Galois group of  $f$  gives information about the roots of  $f$ . A classic example is the seminal work of Galois showing that a polynomial  $f$  is solvable by radicals if and only if its Galois group is solvable. Thus checking whether a polynomial is solvable by radicals amounts to checking whether its Galois group is solvable and hence has an exponential time algorithm. Landau and Miller [39] gave a remarkable polynomial time algorithm for solvability checking. This algorithm manages to check solvability without actually computing the entire Galois group. This remarkable result gives hope that certain non-trivial properties of Galois groups can be tested efficiently. Chapter 7 deals with such efficiently testable properties of Galois group. We give polynomial time algorithms for nilpotence testing and  $\Gamma_d$ -testing.

We generalise the Landau-Miller algorithm and give a polynomial-time algorithm for testing whether the Galois group of a given polynomial is in  $\Gamma_d$  for constant  $d$ . The class of groups  $\Gamma_d$  often crops up in permutation group theoretic problems, e.g. Luks' polynomial-time algorithm [46] for testing isomorphism of bounded degree graphs.

Even though nilpotent groups are solvable the Landau-Miller solvability

test does not give a polynomial time nilpotence test. The Landau-Miller algorithm gives a way to test whether all composition factors of the Galois group are abelian. Nilpotence however is a more “global” property in the sense that it cannot be inferred by knowing the composition factors alone. In Chapter 7 we give a characterisation of nilpotent permutation groups and this characterisation yields a polynomial time nilpotence test.

Many computational problems in algebraic number theory are hard. In the absence of non-trivial upper bounds, conditional results, i.e. results whose validity depends on widely believed yet unproven conjectures of number theory, are of great interest. We now look at complexity theoretic results of this thesis that depend on the validity of the generalised Riemann hypothesis. An important ingredient used in our results is the Chebotarev density theorem, a result on the distribution of primes. For the complexity theoretic applications of this thesis we need an effective version of Chebotarev density theorem due to Lagarias and Odlyzko [35] proved assuming the generalised Riemann hypothesis.

The problem of interest in Chapter 8 is order finding of Galois groups. Given a polynomial  $f(X) \in \mathbb{Q}[X]$  we are interested in computing the order of  $\text{Gal}(f)$  (or equivalently the degree  $[\mathbb{Q}_f : \mathbb{Q}]$  of the extension  $\mathbb{Q}_f/\mathbb{Q}$ ). For permutation groups of degree  $n$  presented via a generating set, the order can be computed in time polynomial in  $n$ . Hence computing the order is no more difficult than computing the Galois group and there is an exponential time algorithm for it. We prove better upper bounds assuming generalised Riemann hypothesis.

Given a polynomial  $f(X) \in \mathbb{Q}[X]$  we show that there is a polynomial time algorithm making one query to a  $\#P$  oracle that computes the order of the Galois group of  $f$  [7]. Furthermore using Stockmeyer’s result on approximating  $\#P$  functions [64], we show that there is a randomised algorithm with NP oracle to approximate the order of the Galois group.

For polynomials with Galois group in  $\Gamma_d$ ,  $d$  a constant, we give a polynomial time reduction from exact order finding to approximate order finding. Thus for polynomials with Galois group in  $\Gamma_d$ ,  $d$  a constant, we have a randomised algorithm with an NP-oracle to compute the order assuming the generalised Riemann hypothesis.

Finally in Chapter 9 we give nontrivial upper bounds on computing the Galois group of some special polynomials. We show that given a polynomial  $f(X) \in \mathbb{Q}[X]$  with abelian Galois group, there is a randomised algorithm for computing the Galois group. This we achieve by giving a polynomial time randomised algorithm for sampling almost uniformly from the Galois group of  $f$ . The effective version of the Chebotarev density theorem plays

a crucial role here. The only nontrivial bound of non-abelian Galois group computation is the following. Given a polynomial  $f(X) \in \mathbb{Q}[X]$  such that every irreducible factor  $g$  of  $f$  has non-abelian simple Galois group of small size, there is a polynomial time deterministic algorithm for computing the Galois group of  $f$ . This result uses a special property of non-abelian semi-simple groups called Scott's Lemma (Lemma 3.6) and is unconditional.

## Chapter 2

# Complexity theory

In this chapter we recall the complexity theory required for this thesis. A detailed presentation is available in any standard textbook on complexity theory ([13, 14]). The survey article of Fortnow and Homer [27] gives a historical perspective together with pointers to many important results of complexity theory.

By an *alphabet* we mean a finite set  $\Sigma$  of *letters*. A *string* of length  $n$  over an alphabet  $\Sigma$  is a finite sequence  $x_1 \dots x_n$  of letters from  $\Sigma$ . For a string  $x$  we will use  $|x|$  to denote the length of  $x$ . By  $\Sigma^*$  we mean the set of all strings over  $\Sigma$ . We will use  $\epsilon$  to denote the *empty string*, the unique string of length 0. A *language* over  $\Sigma$  is a subset of  $\Sigma^*$ .

A *decision problem* is a computational problem where we expect a yes/no answer for e.g. the Graph Isomorphism problem. By suitably encoding instances of a problem, any decision problem can be seen as a language over  $\{0, 1\}$ ; the language corresponding to a decision problem is the set of encodings of input instances which evaluate to “yes”. We will use the terms language and decision problem interchangeably.

Often computational problem require more than a yes/no answer for e.g. consider the problem of sorting a list of numbers. The functions of interest for us are functions from  $\Sigma^*$  to  $\Sigma^*$ . Again for countable sets  $A$  and  $B$  by suitable encoding, functions from  $A$  to  $B$  can be thought of as functions from  $\Sigma^*$  to  $\Sigma^*$ . Computing such functions are called *functional problems*.

The complexity class P is the class of decision problems that can be solved in time bounded by a polynomial in the size of its inputs on a Turing machine. The class P is robust because Turing machines can simulate other reasonable models of computation with a polynomial time overhead. Moreover, most natural problems that have polynomial time algorithms are

tractable in practice. These properties led Edmonds [25] to suggest P as the class of tractable problems and is now widely accepted as the *extended Church-Turing hypothesis*<sup>1</sup>. By FP we mean the class of functions from  $\Sigma^*$  to  $\Sigma^*$  that can be computed on a polynomial time bounded Turing machine.

There are certain problems for which a candidate solution can be verified in polynomial time. The complexity class NP captures exactly this. It is the class of problems that can be solved on a nondeterministic Turing machine in polynomial time. Clearly NP contains the class P but whether this containment is strict is a central open problem in complexity theory. Although widely believed that  $P \neq NP$ , the P vs NP conjecture has successfully resisted attempts of resolution till date. This question gained importance after the concept of NP-completeness was formalised due to the seminal work of Cook [23] and Levin [45] which proved that checking satisfiability of boolean formulae, SAT, is NP-complete. Subsequently Karp [31] showed a number of combinatorial problems including clique problem and travelling salesman problem to be NP-complete. A problem being NP-complete is a strong evidence that it has no polynomial time algorithm. The class of NP-complete problems is particularly important in view of the large number of important problems that it contains. The book of Garey and Johnson [29] gives a through review of NP-completeness and intractability with a list of important NP-complete problems.

Are there problems that are of intermediate complexity in NP? Ladner [34] showed that if  $P \neq NP$  then there are problems that are neither in P nor are NP-complete. It is of interest to know whether there are natural problems of this kind. Graph Isomorphism seems to be one such and is one of the topics of this thesis.

Analogous to the arithmetic hierarchy in computability, Stockmeyer defined the polynomial hierarchy [65]. However, unlike the arithmetic hierarchy, it is not known whether the polynomial hierarchy is infinite. Many interesting problems have been shown to be at different levels of the polynomial hierarchy. Like the working assumption that  $P \neq NP$ , it is widely believed that the polynomial time hierarchy is infinite.

Important subclasses of P are the complexity classes L and NL. The class L consists of problems for which input instance of size  $n$  can be solved with  $O(\log n)$  space on a deterministic Turing machine. Recently, Reinhold [57] proved that L contains undirected  $s$ - $t$  connectivity problem: given an undirected graph and two nodes  $s$  and  $t$  check whether there is a path from  $s$  to  $t$ . As a consequence many problems that involve connectivity in

---

<sup>1</sup>Quantum computing is a potential challenge to this hypothesis.

undirected graphs can be solved in logspace. We summarise these results here for use in later chapters.

**Lemma 2.1** (Reingold). *Given a undirected graph, computing the connected components, find a maximal spanning forest etc. can be solved in logspace.*

The class NL is the nondeterministic version of L consisting of languages that can be accepted by *nondeterministic* logspace bounded Turing machines. A complete problem for NL is the directed  $s$ - $t$  connectivity problem: given a directed graph and two distinguished points  $s$  and  $t$  check if there is a path from  $s$  to  $t$ .

In order to capture complexity classes below P, we need to restrict the oracle access mechanism for nondeterministic and randomised logspace machines. A widely accepted oracle access mechanism is the “Ruzzo-Simon-Tompa” oracle access [58] mechanism in which the oracle machine is restricted to write oracle queries deterministically. In this thesis we will follow this mechanism when we deal with NL oracle machines.

Circuit depth and size gives an elegant way of capturing parallel complexity of a problem. The class  $AC^k$  consists of polynomial sized circuits of depth  $O(\log^k n)$ . If there is an additional constraint that each gate has bounded fanin we get the class  $NC^k$ . It is known that  $NC^k \subseteq AC^k \subseteq NC^{k+1}$ . The class NC is the union  $\cup_{k=1}^{\infty} NC^k$  and captures problems that have efficient parallel algorithms: problems that can be solved in polylog time on a parallel machine with the number of processors bounded by a polynomial in the input size.

## 2.1 Counting complexity classes

*Counting complexity classes* are defined based on the number of accepting and rejecting paths of a nondeterministic computation. Consider the functional problem #SAT of counting the number of satisfying assignments of a boolean formula. Functional problems like #SAT are problems in the complexity class #P. The complexity class #P consists of all functions  $f$  from strings to non-negative integers for which there is a NP machine  $M_f$  such that  $f(x)$  is the number of accepting paths of  $M_f$  on input  $x$ . The functions that are complete for #P are hard to compute functions as they directly give a way of solving NP-complete problems. Surprisingly, certain decision problems that have polynomial time algorithms have counting versions that are #P-complete. A classic example is the problem of counting the number of matchings of a bipartite graph. Counting the number of matching in a

bipartite graph is equivalent to computing the permanent of a  $(0, 1)$ -matrix which was shown to be  $\#P$  complete by Valiant [70].

The class  $\#P$  is closed under sum and product. However it is not closed under subtraction. The closure of  $\#P$  under subtraction is the class GapP. Alternatively, GapP can be defined as the class of all functions  $f$  for which there is a NP-machine  $M_f$  such that  $f(x)$  is the difference of the accepting and rejecting paths of  $M_f$  on input  $x$ . Apart from being closed under subtraction GapP inherits all the nice closure properties of  $\#P$ . We summarise these closure properties below (see [26]).

**Theorem 2.2.** *The class  $\#P$  and GapP are closed under exponential summation and polynomial product, i.e. if  $f(x, y)$  be a function in  $\#P$  (GapP) then for any polynomial  $r(\cdot)$  the functions*

$$g(x) = \sum_{|y| \leq r(|x|)} f(x, y)$$

and

$$h(x) = \prod_{y \leq r(|x|)} f(x, y)$$

are in  $\#P$  (GapP).

The functions in  $\#P$  are hard to compute — Toda's [67] results shows that the entire polynomial hierarchy is contained in  $P^{\#P}$ . Nonetheless, certain  $\#P$  functions can be efficiently approximated, for example  $\#DNFSAT$  has polynomial time approximation algorithms. For approximating general  $\#P$  function the best known result is due to Stockmeyer [64].

**Theorem 2.3.** *For every function  $f$  in  $\#P$  and any fixed constant  $c$  there is a randomised polynomial time algorithm with NP-oracle that on input  $x$  computes a value  $N_x \in \mathbb{N}$  such that*

$$\left(1 - \frac{1}{|x|^c}\right) N_x \leq f(x) \leq \left(1 + \frac{1}{|x|^c}\right) N_x$$

The class PP consists of all languages  $L$  for which there is a GapP function  $f$  such that  $x \in L$  if and only if  $f(x) > 0$ . Surprisingly the entire polynomial hierarchy is contained in  $P^{PP}$  as shown by Toda [67]. The class  $\text{Mod}_kP$  consists of all languages  $L$  for which there is a  $\#P$  function  $f$  such that  $x \in L$  if and only if  $f(x)$  is not divisible by  $k$ . By  $\oplus P$  we mean the class  $\text{Mod}_2P$ .

## UP and SPP

A language  $L$  is in UP if there is a #P function  $f$  such that  $x$  is in  $L$  if  $f(x) = 1$  and  $x$  is not in  $L$  if  $f(x) = 0$ . The class UP was introduced by Valiant [69] to capture the complexity of one-way functions. One-way functions are functions that are easy to compute but hard to invert and their study is central to cryptography. The existence of one-way functions is equivalent to the complexity theoretic assumption that  $UP \neq P$ . The class SPP is the UP analogue of GapP. A language  $L$  is in SPP if there is a function  $f$  in GapP such that for all strings  $x$ ,  $x \in L$  if  $f(x) = 1$  and  $x \notin L$  if  $f(x) = 0$ .

The class SPP is probably one of the most natural counting complexity classes. An important property of SPP is that it is exactly the class of languages that are low for GapP. A language  $L$  is said to be *low* for a complexity class  $\mathcal{C}$  if  $\mathcal{C}^L = \mathcal{C}$ . Schöning [59] introduced the concept of lowness as a tool for classifying complexity theoretic problems and showed that  $NP \cap \text{co-AM}$  is low for  $\Sigma_2^P$ .

Due to the lowness of SPP for GapP, languages in SPP are in and low for all reasonable *gap-definable* complexity classes including itself [26]. Many interesting counting complexity classes like  $\oplus P$ ,  $\text{Mod}_k P$ ,  $\text{PP}$ ,  $\text{C=P}$  are gap definable and hence showing a language  $L$  to be in SPP in one stroke shows that it is in and low for each of these classes. Since SPP is low for itself, the class  $\text{FP}^{\text{SPP}}$  also share these interesting lowness properties. The class  $\text{FP}^{\text{SPP}}$  is essentially SPP as the bits of functions of  $\text{FP}^{\text{SPP}}$  can be computed in SPP. Computational problems that are NP-hard are not expected to share these lowness properties and hence languages in SPP (or functional problems in  $\text{FP}^{\text{SPP}}$ ) are unlikely to be NP-hard. In Chapter 4 we show that the Graph Isomorphism problem is in SPP.

We now describe an important technique that is used to give SPP-upper bounds. Let  $A$  be a language in NP. An polynomial time oracle machine  $M^A$  is said to make UP-like queries to  $A$  if there is an NP machine  $N$  accepting  $A$  such that for all inputs  $x$  and for all queries  $y$  made by  $M$  on input  $x$ ,  $N$  has at most one accepting computation on  $y$ , i.e. for queries made by  $M$  the machine  $N$  behaves like a UP machine. Again due to the closure properties of GapP and the lowness properties of SPP we have the following important theorem [32].

**Theorem 2.4.** *Any language accepted by (function computed by) a polynomial time oracle machine  $M^A$  making UP-like queries to  $A \in \text{NP}$  is in SPP ( $\text{FP}^{\text{SPP}}$ ).*

## Logspace Counting classes

Analogous to GapP and #P by considering NL machines we can define classes GapL and #L. The class #L consists of functions  $f$  for which there is an NL machine  $M_f$  such that  $f(x)$  is the number of accepting paths of  $M_f$  on  $x$ . Similarly we say that a function  $f(x)$  is in  $\#L^A$  for some language  $A$  if there is an oracle  $NL^A$  machine  $M_f^A$  such that  $f(x)$  is the number of accepting paths of  $M_f^A$  on  $x$ . Recall that the oracle machine  $M_f^A$  follows the Ruzzo-Simon-Tompa access mechanism for making queries to  $A$ .

Logspace counting classes have played an important role in classifying natural problems in  $NC^2$ . For example it follows from the work of Toda [66] and Vinay [72] that the problem of computing the determinant of an integer matrix is complete for GapL (for a detailed study see the article of Mahajan and Vinay [51]). Also the complexity of perfect matching is now quite well characterised by Allender *et al* [5] using logspace counting classes and the isolation lemma.

The complexity class  $\text{Mod}_kL$  is the logspace analogue of  $\text{Mod}_kP$ . The class  $\text{Mod}_kL$  consists of languages  $L$  for which there is a function  $f$  in  $\#L$  such that  $x$  is in  $L$  if and only if  $f(x) \not\equiv 0 \pmod{k}$ . It is known that if  $k_1 \mid k_2$  then we have  $\text{Mod}_{k_1}L \subseteq \text{Mod}_{k_2}L$ . For a prime  $p$  the complexity class  $\text{Mod}_pL$  captures the complexity of determinant over  $\mathbb{F}_p$  quite accurately (cf. [20]). Recently, Allender *et al* [4] showed that many important linear algebraic problems like finding the rank and checking feasibility of linear equations over  $\mathbb{F}_p$  are intimately connected to the complexity class  $\text{Mod}_pL$ . A survey of important results in this area is given in the article of Allender [3]. We summarise these results in the following theorem.

**Theorem 2.5** (Buntrock *et al*). *Let  $p$  be a prime. Given a  $m \times n$  matrix  $A$  and a  $m \times 1$  column vector  $\mathbf{b}$  over  $\mathbb{F}_p$  the problem of testing whether the system of linear equations  $A\mathbf{x} = \mathbf{b}$  is feasible is in  $\text{Mod}_pL$ . In case the system is feasible finding a nontrivial solution for the vector  $\mathbf{x}$  of indeterminates is in  $\text{FL}^{\text{Mod}_pL}$ .*

We now define the  $\text{Mod}_kL$  hierarchy. The first level of the  $\text{Mod}_kL$ -hierarchy is the class  $\text{Mod}_kL$ . A language  $L$  is said to be in the  $l + 1$ th level of the  $\text{Mod}_kL$ -hierarchy if there is a function  $f$  in  $\#L^A$ ,  $A$  a language in the  $l$ th level of the  $\text{Mod}_kL$ -hierarchy, such that for all  $x$ ,  $x$  is in  $L$  if and only if  $f(x) \not\equiv 0 \pmod{k}$ .

The  $\text{Mod}_kL$  hierarchy can also be seen as languages accepted by constant depth circuits with  $\text{Mod}_kL$  oracle, i.e. the  $\text{Mod}_kL$  hierarchy is exactly  $\text{AC}^0(\text{Mod}_kL)$ . It is not known whether the  $\text{Mod}_kL$ -hierarchy is infinite.

However for primes  $p$  the  $\text{Mod}_p\mathbf{L}$ -hierarchy collapses to  $\text{Mod}_p\mathbf{L}$ . In Chapter 5 we see the connections of BCGI with the  $\text{Mod}_k\mathbf{L}$ -hierarchy.

## Chapter 3

# Group Theory

In this chapter we review the group theory in particular the theory of permutation groups required for this thesis. The groups we encounter here will all be finite. For a detailed presentation any standard text book on group theory (for example [30]) may be consulted. We follow the notation of Wielandt [74] for permutation groups.

We use the following notation: For groups  $G$  and  $H$ ,  $H \leq G$  means that  $H$  is a subgroup of  $G$ . By  $H < G$  we mean that  $H$  is a strict subgroup of  $G$  i.e.  $H \leq G$  and  $H \neq G$ . By  $G \geq H$  and  $G > H$  we mean  $H \leq G$  and  $H < G$  respectively. Similarly by  $H \trianglelefteq G$  we mean  $H$  is a normal subgroup of  $G$ . When  $H$  is a strictly smaller normal subgroup we denote it by  $H \triangleleft G$ . As before we use  $G \trianglerighteq H$  and  $G \triangleright H$  to mean  $H \trianglelefteq G$  and  $H \triangleleft G$  respectively. Let  $G$  be a group and  $A$  be any subset of  $G$ . By the normal closure of  $A$  in  $G$ , denoted by  $\text{NCL}_G(A)$ , we mean the smallest normal subgroup of  $G$  containing  $A$ . The centraliser  $C_G(A)$  is the subgroup of  $G$  that commutes with all the elements of  $A$ .

Let  $H$  be any subgroup of  $G$ . By the index of  $H$  in  $G$ , denoted by  $[G : H]$ , we mean the number of distinct  $H$  cosets in  $G$ . We have  $[G : H] = \frac{\#G}{\#H}$ . We say that  $H$  embeds into a group  $G$ , denoted by  $H \hookrightarrow G$  if there is a one-to-one homomorphism from  $H$  to  $G$ . In other words  $H$  is isomorphic to a subgroup of  $G$ .

Consider a normal subgroup  $N$  of  $G$ . There is a canonical homomorphism from  $G$  to  $G/N$  that maps an element  $g$  in  $G$  to its coset  $Ng$ . The canonical homomorphism gives a one-to-one correspondences between subgroups of  $G/N$  and subgroups of  $G$  containing  $N$ . For a subgroup  $L$  of  $G/N$ , by the *pullback* of  $L$  in  $G$  we mean the unique subgroup of  $G$  that contains  $N$  under this correspondence. More generally suppose  $\psi$  is a homomor-

phism from  $G$  onto  $H$  then there is a one-to-one correspondence between subgroups of  $G$  containing  $\ker(\psi)$  and subgroups of  $H$  given by  $L \mapsto \psi(L)$ . The *pullback* of a subgroup  $H'$  of  $H$  is the unique  $G'$  such that  $\psi(G') = H'$ .

Let  $K$  and  $H$  be subgroups of  $G$  then the set  $KH$  is also a subgroup if and only if  $KH = HK$  and has order given by  $\#KH = \frac{1}{\#K \cap H} \cdot \#K \cdot \#H$ . If in addition  $H$  is a normal subgroup of  $KH$  and  $K \cap H$  is trivial we say that  $KH$  is the *semidirect product* of  $K$  and  $H$  which we denote by  $K \ltimes H$ . The semidirect product  $K \ltimes H$  is the *direct product* (or just product)  $K \times H$  if both  $K$  and  $H$  are normal subgroup of  $KH$ .

Let  $G$  be any group. For a subgroup  $H$ , a series of groups  $G = G_0 > \dots > G_t = H$  is called a *tower* of groups between  $G$  and  $H$ . The subgroup  $H$  is *subnormal* if there exists a *subnormal tower of groups* between  $G$  and  $H$ , i.e. a tower of groups  $G = G_0 \triangleright \dots \triangleright G_t = H$  such that for all  $0 \leq i < t$ ,  $G_{i+1}$  is a normal subgroup of  $G_i$ . For any group  $G$  the trivial group  $\{1\}$  is subnormal and any subnormal tower of groups between  $G$  and  $\{1\}$  is called a *subnormal series* for  $G$ . A *composition series* for  $G$  is a subnormal series  $G = G_0 \triangleright \dots \triangleright G_t = \{1\}$  such that each of the quotients  $G_i/G_{i+1}$  are simple.

**Definition 3.1** (Solvable groups). *A group  $G$  is said to be solvable if there is a subnormal series  $G = G_0 \triangleright \dots \triangleright G_t = \{1\}$  such that for all  $0 \leq i < t$  the quotient  $G_i/G_{i+1}$  is abelian.*

We now define the class  $\Gamma_d$  of groups. The class  $\Gamma_d$  is a generalisation of the class of solvable groups; if  $G$  is solvable then  $G$  is in  $\Gamma_d$  for any  $d$ . Computational problems for groups in  $\Gamma_d$  occur in many permutation group theoretic algorithms for example in Luks' polynomial time algorithm for bounded degree graphs [46].

**Definition 3.2.** *A group  $G$  is said to be in  $\Gamma_d$  if there is a subnormal series  $G = G_0 \triangleright \dots \triangleright G_t = \{1\}$  such that for all  $0 \leq i < t$  either  $G_i/G_{i+1}$  is abelian or is isomorphic to a subgroup of  $S_d$ , the group of permutations on  $d$  objects.*

For  $d < 5$ , since  $S_d$  is solvable,  $\Gamma_d$  is just the class of solvable groups. The  $\Gamma_d$ -testing, which we will describe in Chapter 7, will use the following closure properties of the class  $\Gamma_d$ .

**Proposition 3.3.** *Any subgroup of a group in  $\Gamma_d$  is also in  $\Gamma_d$ . For any group  $G$  and a normal subgroup  $H$ ,  $G$  is in  $\Gamma_d$  if and only if the groups  $G/H$  and  $H$  are in  $\Gamma_d$ .*

An important subclass of the class of solvable group is the class of nilpotent groups which we define below.

**Definition 3.4** (Nilpotent groups). *A group  $G$  is said to be nilpotent if all its Sylow subgroups are normal.*

The following lemma gives alternate characterisation of nilpotent groups (see Section 10.3 of Hall's book [30]).

**Lemma 3.5.** *Let  $G$  be a finite group then the following are equivalent.*

1.  $G$  is nilpotent.
2.  $G$  is the product of all its Sylow subgroups.
3. For every prime  $p$  that divides the order of  $G$  there is a unique  $p$ -Sylow subgroup.

Let  $G$  be a group and  $H \trianglelefteq G$  be a normal subgroup of  $G$ . A *normal tower* between  $G$  and  $H$  is a subnormal series  $G = G_0 \triangleright \dots \triangleright G_t = H$  such that each  $G_i$  is normal in  $G$ . A *normal series* for  $G$  is a normal tower between  $G$  and the trivial normal subgroup  $\{1\}$ .

Let  $G_1$  and  $G_2$  be two isomorphic groups and let  $\phi : G_1 \rightarrow G_2$  be an isomorphism. The *diagonal subgroup* with respect to  $\phi$ , denoted by  $\text{Diag}_\phi(G_1 \times G_2)$ , is the subgroup  $\{\langle g, \phi(g) \rangle : g \in G_1\}$  of  $G_1 \times G_2$ . Even though the diagonal group  $\text{Diag}_\phi(G_1 \times G_2)$  depends on the isomorphism  $\phi$ , it is isomorphic to  $G_1$  (and  $G_2$ ) and hence we will usually drop the isomorphism  $\phi$ .

A group  $G$  is said to be *simple* if the only proper normal subgroup of  $G$  is the trivial group. Let  $T$  be a simple group. A group  $G$  is said to be  $T$ -semisimple if there is a positive integer  $k$  such that  $G$  is isomorphic to  $T^k$ . An important property of non-abelian semisimple group which we will use in many occasions is Scott's lemma [60] (see Luks' course notes for a proof[48, page 38]).

**Lemma 3.6** (Scott's Lemma). *Let  $T_1, \dots, T_k$  be nonabelian finite simple groups. Let  $G$  be any subgroup of  $\prod_{i=1}^r T_i$  that projects onto each  $T_i$ . Then  $G$  is a direct product of diagonal subgroups. More precisely, there is a partition  $\cup_{j=1}^s I_j$  of  $\{1, \dots, r\}$  such that*

$$G = \prod_{j=1}^s \text{Diag} \left( \prod_{i \in I_j} T_i \right).$$

The Scott's lemma is valid only for nonabelian simple groups. We give a counter example to illustrate this. Consider the vector space  $\mathbb{F}_2^3$ . Let  $W$

be the subspace  $\{(x_1, x_2, x_3)^T \mid x_1 + x_2 + x_3 = 0\}$ . The space  $W$  project onto each of the component  $\mathbb{F}_2$  however it is easy to see that  $W$  is not a product of diagonal subgroups.

### 3.1 Permutation Groups

Let  $\Omega$  be a finite set. The *symmetric group*  $\text{Sym}(\Omega)$  is the group of all permutations on  $\Omega$ . By a *permutation group on  $\Omega$*  we mean a subgroup of  $\text{Sym}(\Omega)$ . By  $S_n$  we mean  $\text{Sym}(\{1, \dots, n\})$ . For a group  $G$  the action  $g : a \mapsto ag$  makes  $G$  a permutation group on itself. This action is called the *right regular action*. Similarly the left regular action is the action  $g : a \mapsto ga$ .

While dealing with permutation groups over  $\Omega$  we adopt the following convention: Lower case Greek letters will be used to denote elements of  $\Omega$  where as upper case Greek letters will be used to denote subsets of  $\Omega$ . Lower case Latin letters will be used to denote elements of  $\text{Sym}(\Omega)$  and upper case Latin letters will be used to denote subsets or subgroups of  $\text{Sym}(\Omega)$ .

The image of  $\alpha \in \Omega$  under the permutation  $g \in \text{Sym}(\Omega)$  will be denoted by  $\alpha^g$ . The advantage of this notation is that group action behave similar to exponentiation, i.e.  $(\alpha^g)^h = \alpha^{gh}$ . For  $A \subseteq \text{Sym}(\Omega)$ ,  $\alpha^A$  denotes the set  $\{\alpha^g : g \in A\}$ . In particular, for  $G \leq \text{Sym}(\Omega)$  the  $G$ -orbit containing  $\alpha$  is  $\alpha^G$ . The  $G$ -orbits form a partition of  $\Omega$ . Given a generating set of  $G$ , a straight forward transitive closure algorithm can be used to compute all the orbits (cf. [49]).

**Theorem 3.7.** *Given  $G \leq \text{Sym}(\Omega)$  by a generating set  $A$  and  $\alpha \in \Omega$ , there is a polynomial-time algorithm to compute  $\alpha^G$ . Moreover for each  $\beta \in \alpha^G$  the above mentioned algorithm can compute a  $g_\beta \in G$  such that  $\alpha^{g_\beta} = \beta$ .*

Let  $G$  be a permutation group action on  $\Omega$ . For  $\Delta \subseteq \Omega$  and  $g \in \text{Sym}(\Omega)$ ,  $\Delta^g$  denotes  $\{\alpha^g : \alpha \in \Delta\}$ . The *set-wise stabiliser* of  $\Delta$ , i.e.  $\{g \in G : \Delta^g = \Delta\}$ , is denoted by  $G_\Delta$ . If  $\Delta$  is the singleton set  $\{\alpha\}$  we write  $G_\alpha$  instead of  $G_{\{\alpha\}}$ . For any  $\Delta$  by  $G|_\Delta$  we mean  $G_\Delta$  restricted to  $\Delta$ . For a set  $\Delta \subseteq \Omega$  the *pointwise stabiliser* will be denoted by  $G(\Delta)$ . Notice that  $G(\Omega) = \{1\}$  and  $G(\{\alpha\}) = G_\alpha$ .

An often used result is the orbit-stabiliser formula stated below [74, Theorem 3.2].

**Theorem 3.8** (Orbit-Stabiliser formula). *Let  $G$  be a permutation group on  $\text{Sym}(\Omega)$  and let  $\alpha$  be any element of  $\Omega$  then the order of the group  $G$  is given by  $\#G = \#G_\alpha \cdot \#\alpha^G$ .*

### 3.2 Strong generator set

Let  $G$  be a group and  $H$  be a subgroup of  $G$ . By a *right traversal* of  $H$  in  $G$  we mean a collection of coset representatives one from each right coset of  $H$  in  $G$ . Similarly we can define the left traversal of  $H$  in  $G$ . Let  $G = G_0 \geq \dots \geq G_t = \{1\}$  be a decreasing tower of subgroups of  $G$ . Let  $C_i$  denote the right traversal of  $G_i$  in  $G_{i-1}$  then the collection  $\cup_{i=1}^t C_i$  is a generator set of  $G$  which we call a *strong generator set* of  $G$  with respect to the given tower. The strong generator set depends on the choice of the traversals  $C_i$  at each stage. Also  $\#C_i = [G_i : G_{i-1}]$  and hence the order of  $G$  is given by  $\prod_{i=1}^t \#C_i$ .

We now describe a particularly useful strong generating set for permutation groups of degree  $n$ . Let  $G$  be a permutation group over  $\Omega$ , a set of cardinality  $n$ . Without loss of generality we assume that  $\Omega$  is the set  $\{1, \dots, n\}$ . Let  $G^{(i)}$  be the point-wise stabiliser of  $\{1, \dots, i\}$ . The tower of groups  $G = G^{(0)} \geq \dots \geq G^{(n-1)} = \{1\}$  gives rise to a strong generator set called the Schreier-Sims strong generating set. For any permutation group  $G \leq S_n$  note that  $\#C_i \leq n - i$  and hence the Schreier-Sims strong generator set is a succinct presentation of  $G$ . There are polynomial time algorithm for computing the Schreier-Sims strong generator set [62, 63, 28]. Many algorithmic tasks involving permutation groups can be solved once a strong generator set is computed. We collect some of the useful computational results in the following theorem.

**Theorem 3.9.** *Let  $G$  a permutation group on  $\Omega$  presented by giving a generator set of  $G$ . The following tasks can be done in polynomial time.*

1. *Computing the Schreier-Sims strong generator set.*
2. *Computing the order of  $G$ .*
3. *Given  $g \in \text{Sym}(\Omega)$  checking whether  $g \in G$ .*
4. *Given a subset  $\Delta$  of  $\Omega$  compute the pointwise stabiliser  $G(\Delta)$ .*

A detailed treatment of computational issues in permutation groups is available in the book by Seress [2].

### 3.3 Transitivity, Blocks and Primitivity

A permutation group  $G$  on  $\Omega$  is *transitive* if there is only one  $G$ -orbit. Suppose  $G \leq \text{Sym}(\Omega)$  is transitive. Then  $\Delta \subseteq \Omega$  is a  $G$ -*block* if for all  $g \in G$

either  $\Delta^g = \Delta$  or  $\Delta^g \cap \Delta = \emptyset$ . For every  $G$ ,  $\Omega$  is a block and each singleton  $\{\alpha\}$  is a block. These are the *trivial blocks* of  $G$ . A transitive group  $G$  is *primitive* if it has only trivial blocks and it is *imprimitive* if it has nontrivial blocks. Examples for primitive groups are  $S_n$  and  $A_n$ . These are the “giants”. However the following bound on primitive groups in  $\Gamma_d$  shows that they are small [11].

**Theorem 3.10** (Babai, Cameron, Pálffy). *Let  $G \leq S_n$  be a primitive permutation group in  $\Gamma_d$ . Then  $\#G \leq n^{O(d)}$ .*

The above mentioned bound is a generalisation of Pálffy’s bound [56] on the order of primitive solvable subgroups of  $S_n$  that was used in the Landau-Miller solvability test [39]. Bounds on sizes of primitive groups such as Theorem 3.10 are important in runtime analysis of various permutation group theory problems. In particular our  $\Gamma_d$ -testing algorithm depends on Theorem 3.10.

A  $G$ -block  $\Delta$  is a *maximal subblock* of a  $G$ -block  $\Sigma$  if  $\Delta \subset \Sigma$  and there is no  $G$ -block  $\Upsilon$  such that  $\Delta \subset \Upsilon \subset \Sigma$ . Let  $\Delta$  and  $\Sigma$  be two  $G$ -blocks. A chain  $\Delta = \Delta_0 \subset \dots \subset \Delta_t = \Sigma$  is a *maximal increasing chain* of  $G$ -blocks between  $\Delta$  and  $\Sigma$  if for all  $i$ ,  $\Delta_i$  is a maximal subblock of  $\Delta_{i+1}$ .

If  $\Delta$  is a  $G$ -block then  $\Delta^g$  is also a  $G$ -block, for each  $g \in G$ . Two  $G$ -blocks  $\Delta_1$  and  $\Delta_2$  are *conjugates* (more precisely  $G$ -conjugates) if there is a  $g \in G$  such that  $\Delta_1^g = \Delta_2$ . It is not difficult to see that the conjugate relation on the set of  $G$ -blocks forms an equivalence relation. Let  $\Delta$  and  $\Sigma$  be two  $G$ -blocks such that  $\Delta \subseteq \Sigma$ . The  $\Delta$ -block system of  $\Sigma$ , is the collection

$$\mathcal{B}(\Sigma/\Delta) = \{\Delta^g : g \in G \text{ and } \Delta^g \subseteq \Sigma\}.$$

The  $\Delta$ -block system of  $\Sigma$  gives a partition of  $\Sigma$ . It follows that  $\#\Delta$  divides  $\#\Sigma$  and by *index* of  $\Delta$  in  $\Sigma$ , which we denote by  $[\Sigma : \Delta]$ , we mean  $\#\mathcal{B}(\Sigma/\Delta) = \frac{\#\Sigma}{\#\Delta}$ . We will use  $\mathcal{B}(\Delta)$  to denote  $\mathcal{B}(\Omega/\Delta)$ .

Blocks are fundamental structures associated with permutation groups and have intimate connections with subgroups of  $G$ . To illustrate this consider a finite group  $G$  as a permutation group on itself under the right regular action. A subset  $H$  of  $G$  is a subgroup if and only if  $H$  is a  $G$ -block containing the identity. For a subgroup  $H$  of  $G$ , which is a  $G$ -block under the right regular action, any other conjugate block of  $H$  is a right coset of  $H$ . More generally if  $G$  is a transitive permutation group on  $\Omega$ , we have the following Galois correspondence between blocks and subgroups [74, Theorem 7.5].

**Theorem 3.11** (Galois correspondence of blocks). *Let  $G \leq \text{Sym}(\Omega)$  be transitive and  $\alpha \in \Omega$ . There is a one-to-one correspondence between  $G$ -blocks containing  $\alpha$  and subgroups of  $G$  containing  $G_\alpha$  given by  $\Delta \mapsto G_\Delta$ . Also for blocks  $\Delta \subseteq \Sigma$  we have  $[G_\Sigma : G_\Delta] = [\Sigma : \Delta]$ .*

In particular the above theorem implies that  $G$  is primitive if and only if for all  $\alpha \in \Omega$ ,  $G_\alpha$  is a maximal proper subgroup of  $G$ .

Let  $G \leq \text{Sym}(\Omega)$  be transitive and  $\Delta$  and  $\Sigma$  be two  $G$ -blocks such that  $\Delta \subseteq \Sigma$ . Let  $G(\Sigma/\Delta)$  denote the group  $\{g \in G : \Upsilon^g = \Upsilon \text{ for all } \Upsilon \in \mathcal{B}(\Sigma/\Delta)\}$ . We write  $G^\Delta$  for the group  $G(\Omega/\Delta)$ . For any  $g \in G_\Sigma$ , since  $g$  setwise stabilises  $\Sigma$ ,  $g$  permutes the elements of  $\mathcal{B}(\Sigma/\Delta)$ . Hence for any  $\Upsilon \in \mathcal{B}(\Sigma/\Delta)$  we have  $\Upsilon^{g^{-1}G(\Sigma/\Delta)g} = \Upsilon$ . Thus,  $G(\Sigma/\Delta)$  is a normal subgroup of  $G_\Sigma$ . In particular,  $G^\Delta$  is a normal subgroup of  $G$ . The following lemma lists important properties of  $G(\Sigma/\Delta)$ .

**Theorem 3.12.**

1. For  $G$ -blocks  $\Delta \subseteq \Sigma$ ,  $G(\Sigma/\Delta)$  is the largest normal subgroup of  $G_\Sigma$  contained in  $G_\Delta$ .
2. Let  $\Sigma$  be  $G$ -block then  $G^\Sigma \hookrightarrow \prod_{\Upsilon \in \mathcal{B}(\Sigma)} G|_\Upsilon$ .
3. Let  $\Delta$  be a  $G$ -subblock of  $\Sigma$  then  $\frac{G_\Sigma}{G(\Sigma/\Delta)}$  is a faithful permutation group on  $\mathcal{B}(\Sigma/\Delta)$  and is primitive if and only if  $\Delta$  is a maximal subblock.
4. The quotient group  $G^\Sigma/G^\Delta$  can be embedded into the product group  $\left(\frac{G_\Sigma}{G(\Sigma/\Delta)}\right)^l$  for some  $l$ .

*Proof.* Let  $N$  be any normal subgroup of  $G_\Sigma$  contained in  $G_\Delta$ . We have for  $\Delta^N = \Delta$ . Consider any  $\Upsilon \in \mathcal{B}(\Sigma/\Delta)$ . Since  $G_\Sigma$  acts transitively on  $\mathcal{B}(\Sigma/\Delta)$  there is a  $g \in G_\Sigma$  such that  $\Upsilon = \Delta^g$ . Since  $gN = Ng$  we have  $\Upsilon^N = \Delta^{gN} = \Delta^{Ng} = \Upsilon$ . This proves that for all  $\Upsilon \in \mathcal{B}(\Sigma/\Delta)$ ,  $\Upsilon^N = \Upsilon$ . Hence  $N$  is contained in  $G(\Sigma/\Delta)$ . This proves part 1.

The group  $G^\Sigma$  consists of all elements  $g$  of  $G$  that fixes setwise every block  $\Upsilon \in \mathcal{B}(\Sigma)$  and hence we have the embedding of part 2.

That  $\frac{G_\Sigma}{G(\Sigma/\Delta)}$  acts faithfully on  $\mathcal{B}(\Sigma/\Delta)$  follows from the fact that for any two  $g$  and  $h$  in  $G_\Sigma$ ,  $g$  and  $h$  has the same action on  $\mathcal{B}(\Sigma/\Delta)$  if and only if  $gG(\Sigma/\Delta)$  and  $hG(\Sigma/\Delta)$  are equal. Any nontrivial  $\frac{G_\Sigma}{G(\Sigma/\Delta)}$ -block of  $\mathcal{B}(\Sigma/\Delta)$  gives a nontrivial  $G$ -block between  $\Delta$  and  $\Sigma$  and vice versa. Thus,  $\frac{G_\Sigma}{G(\Sigma/\Delta)}$  is primitive if and only if  $\Delta$  is a maximal subblock of  $\Sigma$ .

Finally for the last statement notice that we have the group isomorphism

$$\frac{G|_{\Upsilon}}{G(\Upsilon/\Delta_{\Upsilon})|_{\Upsilon}} \cong \frac{G_{\Upsilon}}{G(\Upsilon/\Delta_{\Upsilon})}.$$

Also since  $G^{\Delta} = G^{\Sigma} \cap \prod G(\Upsilon/\Delta_{\Upsilon})|_{\Upsilon}$  where  $\Upsilon$  varies over  $\mathcal{B}(\Sigma)$  and  $\Delta_{\Upsilon}$  is any conjugate of  $\Delta$  contained in  $\Upsilon$  we have

$$G^{\Sigma}/G^{\Delta} \hookrightarrow \prod_{\Upsilon \in \mathcal{B}(\Sigma)} \frac{G|_{\Upsilon}}{G(\Upsilon/\Delta_{\Upsilon})|_{\Upsilon}} = \prod_{\Upsilon \in \mathcal{B}(\Sigma)} \frac{G_{\Upsilon}}{G(\Upsilon/\Delta_{\Upsilon})}.$$

Let  $g \in G$  be any element that maps  $\Delta$  to  $\Delta_{\Upsilon}$  then it follows that  $G_{\Upsilon} = g^{-1}G_{\Sigma}g$  and  $G(\Upsilon/\Delta_{\Upsilon}) = g^{-1}G(\Sigma/\Delta)g$ . Hence the quotient groups  $\frac{G_{\Sigma}}{G(\Sigma/\Delta)}$  and  $\frac{G_{\Upsilon}}{G(\Upsilon/\Delta_{\Upsilon})}$  are isomorphic. Thus, we see that  $G^{\Sigma}/G^{\Delta}$  is isomorphic to a subgroup of  $\left(\frac{G_{\Sigma}}{G(\Sigma/\Delta)}\right)^l$  for some  $l$ .  $\square$

**Lemma 3.13.** *Let  $G \leq \text{Sym}(\Omega)$  be transitive and  $N$  be a normal subgroup of  $G$ . Let  $\alpha \in \Omega$ . Then the  $N$ -orbit  $\alpha^N$  is a  $G$ -block and the collection of  $N$ -orbits is an  $\alpha^N$ -block system of  $\Omega$  under  $G$  action. If  $N \neq \{1\}$  then  $\#\alpha^N > 1$ . Furthermore, if  $G_{\alpha} \leq N \neq G$  then the  $\alpha^N$ -block system is nontrivial implying that  $G$  is not primitive.*

*Proof.* Let  $\alpha \in \Omega$  and  $g \in G$ . Then  $(\alpha^N)^g = \alpha^{Ng} = \alpha^{gN} = (\alpha^g)^N$ . Thus  $(\alpha^g)^N$  and  $\alpha^N$  are identical if  $\alpha^g \in \alpha^N$  and disjoint otherwise, since they are distinct  $N$ -orbits. Hence  $\alpha^N$  is a  $G$ -block and the orbits of  $N$  is an  $\alpha^N$ -block system of  $\Omega$  under  $G$  action. If  $\alpha^N = \{\alpha\}$  then for all  $\beta \in \Omega$ ,  $\beta^N = \{\beta\}$ . Thus,  $N = \{1\}$ .

Finally, note that by the Orbit-Stabiliser formula  $\#G = \#\Omega \cdot \#G_{\alpha}$  and  $\#N = \#\alpha^N \cdot \#G_{\alpha}$ . Thus, if  $\{1\} \neq N \neq G$  then  $\alpha^N$  is a proper  $G$ -block. This completes the proof.  $\square$

### 3.4 Structure Tree and Structure Forest

An important structure associated with a transitive permutation group is its structure tree. Structure trees have proved useful in analysing various divide and conquer algorithms for permutation groups (see, e.g. [49]). Let  $G$  transitive permutation group acting on  $\Omega$ . Consider a maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_t = \Omega$ . For each such maximal chain we can associate a structure tree as follows:

**Definition 3.14** (Structure Tree). *Let  $G$  be a transitive permutation group acting on  $\Omega$  and let  $\Omega = \Delta_0 \supset \dots \supset \Delta_t = \{\alpha\}$  be any maximal decreasing chain of  $G$ -blocks. A structure tree of  $G$  with respect to this maximal chain is a labelled rooted tree of depth  $t$  satisfying the following conditions.*

1. *The root is labelled  $\Omega$ .*
2. *The leaves are labelled with singleton sets  $\{\gamma\}$ ,  $\gamma \in \Omega$ .*
3. *For each node labelled  $\Sigma$  at level  $i$  the children of  $\Sigma$  are  $\{\Delta \in \mathcal{B}(\Delta_{i+1}) : \Delta \subset \Sigma\}$ .*

We will identify the nodes of the tree with its labels (which are  $G$ -blocks). Any root-to-leaf path  $\Omega = \Delta_0 \supset \dots \supset \Delta_t = \{\alpha\}$  in a structure tree is a maximal decreasing chain of  $G$ -blocks. Conversely any maximal decreasing chain  $\Omega = \Delta_0 \supset \dots \supset \Delta_t = \{\alpha\}$  of  $G$ -blocks gives a structure tree for which it is a root-to-leaf path. If two maximal chains  $\Omega = \Delta_0 \supset \dots \supset \Delta_t = \{\alpha\}$  and  $\Omega = \Sigma_0 \supset \dots \supset \Sigma_s = \{\beta\}$  gives the same structure tree then  $t = s$  and there is a permutation  $g \in G$  such that  $\Delta_i^g = \Sigma_i$  for each  $i$  (pick any  $g$  that maps  $\alpha$  to  $\beta$ ). For two nodes  $\Delta$  and  $\Sigma$  of a structure tree  $T$ ,  $\Sigma$  is an ancestor of  $\Delta$  if and only if  $\Delta \subseteq \Sigma$ .

Let  $G$  be a transitive permutation group on  $\Omega$  and let  $T$  be any structure tree of  $G$ . There is a natural action of  $G$  on the nodes of  $T$ ; a node  $\Delta$  under the action of  $g \in G$  goes to the node  $\Delta^g$ . This action embeds  $G$  into the group of automorphisms of the labelled graph  $T$ . For any node  $\Sigma$  the subgroup of  $G$  that fixes  $\Sigma$  is the group  $G_\Sigma$  and for any child  $\Delta$  of  $\Sigma$  the group  $G(\Sigma/\Delta)$  is the subgroup of  $G$  that fixes all the siblings of  $\Delta$ . It follows from Theorem 3.12 that  $G$  acts primitively on the children of the root of  $T$ .

For intransitive groups  $G$  on  $\Omega$  instead of a structure tree we have a *structure forest*. Let  $\Omega_1, \dots, \Omega_k$  be the distinct  $G$ -orbits then  $G$  restricted to  $\Omega_i$  is transitive on  $\Omega_i$ . A structure forest of  $G$  is the collection of structure trees  $\{T_1, \dots, T_k\}$  where  $T_i$  is a structure tree of transitive group  $G|_{\Omega_i}$  on  $\Omega_i$ . Many permutation group algorithm uses a divide and conquer strategy by first computing the orbits and then finding the maximal blocks. The structure forest thus appear naturally in the analysis of such algorithms.

Given a permutation group  $G$  acting on  $\Omega$  there is a polynomial time algorithm to compute a structure tree for  $G$ . A key step involved is to compute a minimal  $G$ -block. The polynomial time algorithm for finding the minimal  $G$ -block follows from the fact that the smallest block that contains  $\alpha$  and  $\beta$  is exactly the connected component of  $\alpha$  in the undirected graph  $X$  where the vertex set is  $\Omega$  and the edge set is  $\{\{\alpha, \beta\}^g : g \in G\}$ . The

graph  $X$  can be constructed in polynomial time as it amounts to finding the orbit of  $\{\alpha, \beta\}$  under the action of  $G$  on the set of unordered pairs of  $\Omega$ . To summarise the above discussion we have the following lemma [46, Lemma 1.3]

**Lemma 3.15.** *Given a permutation group  $G$  on  $\Omega$  via a generating set  $A$  and a  $G$ -orbit  $\Omega'$  there is a polynomial time algorithm for computing the  $G$ -block system of  $\Omega'$  corresponding to a minimal  $G$ -block.*

## Chapter 4

# The Graph Isomorphism problem

In this chapter we study the Graph Isomorphism problem (GI for short). A *graph*  $X$  for us is an undirected graph i.e. a finite set of vertices  $V(X)$  and a set  $E(X)$  of unordered pairs of elements of  $V(X)$ . Two graphs  $X_1$  and  $X_2$  are isomorphic if there is a one-to-one map  $f$  from  $V(X_1)$  onto  $V(X_2)$  that preserves the edge relations i.e. for every unordered pair  $\{u, v\}$ ,  $u, v \in V(X_1)$ ,  $\{u, v\} \in E(X_1)$  if and only if  $\{f(u), f(v)\} \in E(X_2)$ . The function  $f$  we will call an isomorphism between  $X_1$  and  $X_2$ .

**Definition 4.1** (Graph Isomorphism problem). *Given two graphs  $X_1$  and  $X_2$  test whether they are isomorphic.*

The Graph Isomorphism problem is believed to be one of the natural problems that lie between the complexity classes P and NP. Even though no polynomial time algorithm is known, it is not expected to be NP-complete. In fact, under reasonable complexity theoretic assumptions, it appears that the graph isomorphism problem is unlikely to be NP-complete. It was shown by Boppana *et al* [19] that Graph Isomorphism is not NP-complete unless the polynomial hierarchy collapses to  $\Sigma_2^P$ . They also showed that graph non-isomorphism is in AM and hence GI is in  $\text{NP} \cap \text{co-AM}$ . Schöning [59] generalised the result of Boppana *et al* [19] and proved that  $\text{NP} \cap \text{co-AM}$  is low for  $\Sigma_2^P$ . It then follows that any language in  $\text{NP} \cap \text{co-AM}$  cannot be NP-complete unless the polynomial hierarchy collapses to  $\Sigma_2^P$ . Lowness for PP is another property that NP-complete problems are unlikely to have. It was shown by Köbler *et al* [32] that GI is in LWPP and hence is low for the class PP. For a detailed study of the Graph Isomorphism problem from a complexity theoretic perspective see the book of Köbler *et al* [33].

The main result of this chapter is the SPP upper bound for Graph Isomorphism [6]. Our result is an improvement on the LWPP upper bound of Köbler *et al* [32]. As mentioned in Chapter 2 a problem in SPP (or  $\text{FP}^{\text{SPP}}$ ) is low for many interesting complexity classes like  $\oplus\text{P}$  for example. Hence our result shows that GI is low for each of these complexity classes. Earlier it was not even known whether GI is in  $\oplus\text{P}$ .

To prove the SPP upper bound for GI it is sufficient to give an  $\text{FP}^{\text{SPP}}$  algorithm for AUT. In fact in Section 4.4 we show that a generic permutation group theoretic problem which we will call FINDGROUP problem, has a  $\text{FP}^{\text{SPP}}$  algorithm. Many interesting permutation group problems including AUT will be special cases of this generic problem.

## 4.1 Group theoretic formulation of Graph Isomorphism problem

We now formulate the Graph Isomorphism problem as a group theory problem. Many important upper bounds for GI were achieved by making use of this group theoretic formulation. The polynomial time algorithm for bounded valance graphs [46] and the fastest known algorithm for general graphs [75, 12] are group theoretic in nature.

Consider the family of  $n$ -vertex graph. With out loss of generality we fix their vertex set to be an  $n$  element set  $\Omega$ , say  $\{1, \dots, n\}$  for example. Let  $\mathcal{G}(\Omega)$  be the set of graphs with vertex set  $\Omega$ . The permutations  $g \in \text{Sym}(\Omega)$  has a natural induced action on the set of unordered pairs  $\binom{\Omega}{2}$  namely  $\{u, v\}^g = \{u^g, v^g\}$ . This natural action induced action on  $\mathcal{G}(\Omega)$ ; a permutation  $g \in \text{Sym}(\Omega)$  maps the graph  $X = (\Omega, E)$  to  $X^g = (\Omega, E^g)$ . The Graph Isomorphism problem can be formulated as follows: Given two graphs  $X_1$  and  $X_2$  in  $\mathcal{G}(\Omega)$  check whether they are in the same orbit under the  $\text{Sym}(\Omega)$  action.

An *automorphism of a graph*  $X$  in  $\mathcal{G}(\Omega)$  is an element of  $\text{Sym}(\Omega)$  such that  $X^g = X$ . The set of all automorphisms of a graph  $X$ , which we will denote by  $\text{Aut}(X)$ , is but the stabiliser subgroup of the point  $X$  under  $\text{Sym}(\Omega)$  action. We now define the Graph Automorphism problem, AUT for short, which is closely related to GI.

**Definition 4.2** (Graph Automorphism problem). *Given an undirected graph  $X$  compute a generator set of  $\text{Aut}(X)$  as a permutation group on  $V(X)$ .*

By #GI we mean the counting problem where given two graphs  $X_1$  and  $X_2$  we want to compute the number of isomorphisms between  $X_1$  and  $X_2$ .

Similarly by #AUT we mean the counting version of AUT, i.e. given a graph  $X$  counting the number of automorphisms of  $X$ .

The problem #AUT is polynomial time Turing reducible to AUT as there are polynomial time algorithm for computing the order of a permutation group given by a generating set. Let  $X_1$  and  $X_2$  be isomorphic graphs in  $\mathcal{G}(\Omega)$ . Let  $g \in \text{Sym}(\Omega)$  be any isomorphism between  $X_1$  and  $X_2$  then the set of all isomorphism between  $X_1$  and  $X_2$  is the coset  $\text{Aut}(X_1)g$  and hence  $\#\text{Iso}(X_1, X_2) = \#\text{Aut}(X_1) = \#\text{Aut}(X_2)$ . Mathon [52] proved that the Graph Isomorphism problem and Graph Automorphism problem are equivalent under polynomial time Turing reductions. As a result we have the following theorem.

**Theorem 4.3** (Mathon). *The computational problems GI, AUT, #GI and #AUT are all equivalent under polynomial-time Turing reductions.*

Mathon's result together with Toda's [67] theorem gives another reason to believe that GI is unlikely to be NP-complete. By Toda's theorem  $\text{P}^{\#\text{P}}$  contains the entire polynomial hierarchy. Therefore the counting problem #GI is not #P-complete unless the polynomial hierarchy collapses to  $\Delta_2^p$ . Counting versions of almost all known NP-complete problems are complete for #P. In fact counting versions of certain problems in P, like perfect matching, are also complete for #P.

## 4.2 Problems related to Graph Isomorphism

We look at problems that are closely related to GI. Many isomorphism problems of combinatorial structures are closely connected to the Graph Isomorphism problem. For a detailed account of these problems, their complexity and their relation to GI see the book of Köbler *et al* [33, Chapter 1].

First we consider slight variations of the Graph Isomorphism problem. A directed graph consists of a finite set of vertices  $V$  and a collection  $E \subseteq V \times V$ . Isomorphisms of directed graphs should also preserve the direction of edges, i.e. a bijection  $f$  from  $V(X_1)$  to  $V(X_2)$  is an isomorphism if for all  $u$  and  $v$  in  $V(X_1)$  the ordered pair  $(u, v) \in E(X_1)$  if and only if  $(f(u), f(v)) \in E(X_2)$ . The problem of directed Graph Isomorphism is to check whether two directed graphs  $X_1$  and  $X_2$  are isomorphic. A vertex coloured graph is a graph together with a colouring function, i.e. a map  $\psi : V \rightarrow C$  where  $C$  is the set of colours. For coloured graphs  $X_1$  and  $X_2$  a map  $f : V(X_1) \rightarrow V(X_2)$  is an isomorphism if  $f$  should preserve the

edge relations and also the colours, i.e. for any  $u \in V(X_1)$  both  $u$  and  $f(u)$  should of the same colour. The problem of coloured Graph Isomorphism is to check whether two coloured graphs are isomorphic.

By attaching suitable graph gadgets one can show that each of these problems are polynomial time equivalent to GI (see [53]).

We now consider the Group Isomorphism problem. We are given a group  $G$  via its multiplication table, i.e. a two dimensional array indexed by elements of the group  $G$  where for each  $g$  and  $h$  in  $G$  the  $(g, h)$ th entry is  $gh$ . The Group Isomorphism problem, GRPI for short, is to check whether two groups presented via their multiplication table is isomorphic. It turns out that  $\text{GRPI} \leq_m^p \text{GI}$  (see [53]) however a reduction in the other direction is not known. If instead of groups we consider semigroup Isomorphism, SEMIGRPI we have and equivalence result, i.e.  $\text{SEMIGRPI} \equiv_m^p \text{GI}$  [18]. We now define the Setwise stabiliser problem SETSTAB.

**Definition 4.4** (Setwise Stabiliser Problem). *Given a generator set for a permutation group  $G$  over  $\Omega$  and a subset  $\Delta \subseteq \Omega$  compute a generator set for  $G_\Delta$  the set-wise stabiliser of  $\Delta$ .*

There is a polynomial time many one reduction from AUT to SETSTAB. To see this consider a graph  $X = (V, E)$  consider the action of  $G = \text{Sym}(V)$  on the set  $\Omega = \binom{V}{2}$  of unordered pairs of  $V$ . Then  $\text{Aut}(X)$  is nothing but  $G_E$  (or  $G_{\Omega \setminus E}$ ). However no reduction is known in the other direction. The Setwise stabiliser problem seems to be harder than the Graph Isomorphism problem.

We now define the *hidden subgroup problem*, HSP for short. Many interesting computational problems for which there are efficient quantum algorithms are variants of the HSP. This include the Shor's polynomial time algorithm for integer factoring and discrete logarithm [61]. Many other group theoretic problems including AUT can be cast as a Hidden subgroup problem.

For a group  $G$  and a set  $X$ ,  $\phi : G \rightarrow X$  is a *right hiding function* for a subgroup  $H$  of  $G$  if  $\phi$  is constant and distinct on the right cosets of  $H$ , i.e. for any  $g_1$  and  $g_2$  in  $G$ ,  $\phi(g_1) = \phi(g_2)$  if and only if  $Hg_1 = Hg_2$ .

**Definition 4.5** (Hidden Subgroup Problem). *Given a group  $G$  by its generator set and a hiding function  $\phi : G \rightarrow X$  for a subgroup  $H$  compute a generator set for  $H$ .*

### 4.3 Computing the lex-least element of a Coset

Consider a finite totally-ordered set  $(\Omega, <)$ . The order  $<$  on  $\Omega$  induces a natural order, the lexicographic order, on  $\text{Sym}(\Omega)$  as follows:  $g < h$  if there is an  $\alpha \in \Omega$  such that  $\alpha^g < \alpha^h$  and for all  $\beta < \alpha$ ,  $\beta^g = \beta^h$ . It is not difficult to see that  $<$  is a total order on  $\text{Sym}(\Omega)$  and the least element is 1. In this section we give a polynomial time algorithm that computes the least element of  $Gg$  given  $g$  and a generating set for  $G$ . This algorithm plays a crucial role in our SPP algorithm for FINDGROUP.

**Theorem 4.6.** *Given a permutation group  $G$  on a totally ordered set  $(\Omega, <)$  via a generator set. Let  $g \in \text{Sym}(\Omega)$  be any permutation of  $\Omega$ . There is a polynomial-time algorithm that computes the lexicographically least element of  $Gg$ .*

*Proof.* Let  $g^*$  be the lex-least element of  $Gg$ . Let  $\alpha$  be the least element of  $\Omega$ . We first compute the set  $\alpha^{Gg}$  in polynomial time — First compute the  $G$ -orbit  $\Sigma = \alpha^G$  using Theorem 3.7 and then compute  $\Sigma^g$ . Furthermore we assume that we have actually computed for each  $\eta \in \alpha^{Gg}$  an element  $g_\eta \in Gg$  such that  $\alpha^{g_\eta} = \eta$ . Let  $\beta$  be the least element of  $\alpha^{Gg}$  then we have  $\alpha^{g^*} = \beta$ . Otherwise  $\alpha^{g_\beta} = \beta < \alpha^{g^*}$  and hence  $g_\beta \in Gg$  is lesser than  $g^*$  in the lexicographic order which is a contradiction.

Every element of the coset  $G_\alpha g_\beta \subseteq Gg$  maps  $\alpha$  to  $\beta$ . Conversely consider any  $h \in Gg$  that maps  $\alpha$  to  $\beta$ . The elements  $hg_\beta^{-1}$  fixes  $\alpha$  and hence  $h \in G_\alpha g_\beta$ . Therefore  $G_\alpha g_\beta$  is exactly the set of elements that map  $\alpha$  to  $\beta$  and hence contains  $g^*$ . We can use the above idea repeatedly as follows: Let  $G^{(i)}$  be the subgroup of  $G$  that fixes pointwise the first  $i$  elements  $\alpha_1, \dots, \alpha_i$  of  $\Omega$ . By Theorem 3.9 we can compute the strong generator set of  $G$  and hence compute the generator sets of each of the groups  $G^{(i)}$  in polynomial time. Starting with  $g_0 = g$ , for  $1 \leq i < n$  we compute an element  $g_i \in Gg$  such that  $g^* \in G^{(i)}g_i$ . In the  $i$ th step we compute the least element  $\beta_i$  in the set  $\alpha_i^{G^{(i-1)}g_{i-1}}$  and a permutation  $h$  that maps  $\alpha_i$  to  $\beta_i$ . The permutation  $g_i$  is  $h$ . Since  $G^{(n-1)} = \{1\}$ , where  $n = \#\Omega$ , we have  $g_{n-1} = g^*$ . Algorithm 1 is the detailed presentation of the above discussion.  $\square$

**Input:** An ordered set  $\Omega$ , a generator set for  $G \leq \text{Sym}(\Omega)$ , and a  $g \in \text{Sym}(\Omega)$

**Output:** Lexicographically least element in  $Gg$

Let  $\alpha_1 < \dots < \alpha_n$  be the ordered list of element of  $\Omega$ .

Let  $G^{(i)}$  be the pointwise stabiliser of  $\{\alpha_1, \dots, \alpha_i\}$ .

$g_0 = g$

**for**  $i = 0$  **to**  $n - 1$  **do**

Find the element  $\gamma$  in  $\alpha_i^{G^{(i)}}$  and  $h \in G^{(i)}$  such that  $\alpha_i^h = \gamma$  and

$\beta = \gamma^{g_i}$  is minimum (Use Theorem 3.7);

$g_{i+1} = hg_i$

**end**

**return**  $g_n$

**Algorithm 1:** Lexicographically least in a Right Coset

We can easily extend the above result to show the following.

**Theorem 4.7.** *For an ordered set  $\Omega$  there is a polynomial time algorithm that on input a generator set for permutation group  $G$  on  $\Omega$  and  $g_1, g_2 \in \text{Sym}(\Omega)$ , computes the lexicographically least element of  $g_1Gg_2$ . In particular there is a polynomial time algorithm to compute the lex-least element for a left coset  $gG$ .*

*Proof.* Since  $g_1Gg_2 = g_1Gg_1^{-1}g_1g_2$  and the lex-least element of  $g_1Gg_2$  is equal to the lex least element of  $Hg'$  where  $H = g_1Gg_1^{-1}$  and  $g' = g_1g_2$ . If  $A$  is a generator set of  $G$  then  $g_1Ag_1^{-1} = \{g_1hg_1^{-1} : h \in A\}$  is a generator set for  $H$ . The result then follows from Theorem 4.6.  $\square$

## 4.4 The FINDGROUP problem

In this section we study the generic group theoretic problem FINDGROUP. We give an  $\text{FP}^{\text{SPP}}$  upper bound for FINDGROUP. Many permutation group theoretic problems AUT, HSP and SETSTAB special cases of FINDGROUP. Hence giving an  $\text{FP}^{\text{SPP}}$  bound for FINDGROUP in one stroke gives SPP (or  $\text{FP}^{\text{SPP}}$ ) upper bounds for each of these problems.

We define a generic permutation group problem called FINDGROUP. To each instance  $\langle x, 0^n \rangle$  of FINDGROUP there is an associated an unknown subgroup  $G_x \leq S_n$  for which there is polynomial time membership test, i.e. there is a polynomial-time function  $\text{mem}(x, g)$  that takes  $x$  and  $g \in S_n$  as input and evaluates to **true** if and only if  $g \in G_x$ . The FINDGROUP problem is to compute a generating set for  $G_x$  given  $\langle x, 0^n \rangle$  as input. The rest of the section is devoted to the  $\text{FP}^{\text{SPP}}$  upper bound for FINDGROUP

Fix an input instance  $\langle x, 0^n \rangle$  be an input instance of FINDGROUP. Our goal is to compute a strong generator set for  $G_x \leq S_n$  using the membership function  $mem(.,.)$  as a subroutine. The input instance being fixed, we will sometimes drop the subscript and write  $G$  instead of the group  $G_x$ . Let  $G^{(i)} \leq G$  denote pointwise stabiliser of  $\{1, \dots, i\}$ . Starting from  $i = n-1$  we compute a strong generator set for  $G^{(i)}$  for decreasing values of  $i$ . Assuming that we have a generator set for  $G^{(i)}$ , we show that a generator set for  $G^{(i-1)}$  can be computed in  $\text{FP}^{\text{SPP}}$ . We give a polynomial time deterministic algorithm making UP-like queries to a language  $L$  in NP which we now define. Consider the NP-machine  $M$  defined in Algorithm 2 and let  $L$  be the language accepted by  $M$ .

- Input:**  $x \in \{0, 1\}^*$ , an integer  $0 \leq i \leq n$ , a subset  $S \subseteq S_n$  and a partial permutation  $\pi$
- Verify using the membership test  $mem(.)$  that  $S \subseteq G^{(i)}$
- 1 Guess  $g \in G^{(i-1)}$  i.e. guess  $g \in S_n$  and verify using  $mem(.)$ .  
Let  $H$  be the group generated by  $S$ .  
Use Theorem 4.6 to compute the lexicographically least element  $g^*$  of  $Hg$ .
  - 2 **if**  $g \neq g^*$  **then Reject.**
  - 3 **if**  $g^*$  *extends*  $\pi$  **then Accept.**  
**else Reject.**

**Algorithm 2:** The NP machine for  $A$

Here by a partial permutation we mean a partial function one-to-one function from  $\{1, \dots, n\}$  to itself. Let the  $G^{(i-1)}$ -orbit of  $i$  be  $\{i_1, \dots, i_k\}$ . The set  $\{g_1, \dots, g_k\}$  where  $g_s \in G^{(i-1)}$  is any permutation that maps  $i$  to  $i_s$  forms a right traversal of  $G^{(i)}$  in  $G^{(i-1)}$ . Let  $g_s^*$  denote the lexicographically least element in the coset  $G^{(i)}g_s$ . We have the following proposition for the language  $L$

**Proposition 4.8.** *Let  $S$  be a generator set for the group  $G^{(i)}$ . Consider a partial permutation  $\pi$  whose domain includes  $\{1, \dots, i\}$ . Then the tuple  $\langle x, i, S, \pi \rangle \in L$  if and only if  $\pi$  maps  $i$  to  $i_s$  and agrees with  $g_s^*$  for some  $s$ . For such an input, the machine  $M$  has only one accepting path.*

*Proof.* The nondeterminism in the definition of  $M$  is due to step 1 of Algorithm 2 where we guess an element  $g$  of  $G^{(i-1)}$ . Since  $g \in G^{(i-1)}$ ,  $i^g = i_s$  for some  $s$ . If  $S$  generates  $G^{(i)}$  then only the path that guessed  $g_s^*$  survives (on all other paths step 2 rejects). Furthermore, if  $\langle x, i, S, \pi \rangle \in L$  then  $\pi$  agrees with  $g_s^*$  as well (we verified this in step 3). The proposition follows. More

generally if  $S$  generates a subgroup  $H$  of  $G^{(i)}$  then the number of accepting paths on such a partial permutation  $\pi$  will be the index  $[G^{(i)} : H]$ .  $\square$

We are ready to give the  $\text{FP}^{\text{SPP}}$  algorithm for FINDGROUP. To begin with we already know  $G^{(n-1)}$ . Assume that a generating set  $D_i$  of  $G^{(i)}$  is known. From  $D_i$  we will compute a right traversal  $C_i$  of  $G^{(i)}$  in  $G^{(i-1)}$  using the language  $L$  as oracle. The base algorithm will be a deterministic polynomial time algorithm that makes UP-like queries to  $L$ , i.e. for all queries that the machine makes to  $L$  the NP-machine of  $M$  described in Algorithm 2 will have at most one accepting path. To begin with we have a generating set  $D_{n-1} = \{1\}$  of  $G^{(n-1)}$ . The complete algorithm is give below.

```

 $C_i \leftarrow \emptyset$  for every  $0 \leq i \leq n - 2$ .
 $D_i \leftarrow \emptyset$  for every  $0 \leq i \leq n - 2$ .
 $D_{n-1} = 1$ 
1 for  $i = n - 1$  down to 1 do
    Let  $\pi_i$  be the partial permutation that fixes all elements from 1 to
     $i - 1$ .
2     for  $j = i + 1$  to  $n$  do
3          $\pi' \leftarrow \pi[i := j]$ .
         if  $\langle x, D_i, i, \pi' \rangle \in L$  then
              $C_i \leftarrow C_i \cup g$  where  $g = \text{prefixSearch}(x, D_i, i, \pi')$ .
         end
     end
     $D_{i-1} \leftarrow D_i \cup C_i$ .
end
Result:  $D_0$ .
function  $\text{prefixSearch}(x, D_i, i, \sigma)$ 
begin
4     for  $k \leftarrow i + 1$  to  $n$  do
         Find the element  $l$  not in the range of  $\pi'$  such that
          $\langle x, 0^n, D_i, i, j, \pi'[k := l] \rangle \in L$  by making queries to  $L$ .
          $\sigma := \sigma[k := l]$ .
     end
    return  $\sigma$ .
end

```

**Algorithm 3:**  $\text{FP}^L$  algorithm FINDGROUP

By  $\pi[l := m]$  we mean the partial permutation  $\sigma$  that agrees with  $\pi$  except at  $l$  where its value is  $m$ . The function  $\text{prefixSearch}(x, i, D_i, \pi')$  completes the partial permutation  $\pi'$  to an appropriate  $g_s^*$  using  $L$  as an oracle.

**Proposition 4.9.** *The Algorithm 3 computes the generator set for  $G$  and for all queries made to  $L$  the machine  $M$  described in Algorithm 2 has at most one accepting path.*

*Proof.* The invariant of the loop 1 is that  $D_i$  generates the subgroup  $G^{(i)}$ . In the beginning of the loop the invariant is true. Since inductively we have made sure that  $D_i$  generates  $G^{(i)}$  by Proposition 4.8 there is at most one accepting path for any queries made, whether in step 3 or in step 4. Hence the polynomial time oracle machine makes only UP-like query to  $L$  whether in the main loop or in the subroutine `prefixSearch()`.

Proposition 4.8 also guarantees that the query in step 3 gives a “yes” answer if and only if  $j$  is in the orbit  $i^{G^{(i-1)}}$ . When  $j$  is indeed in the orbit  $i^{G^{(i-1)}}$  then `prefixSearch()`, by making queries to  $L$ , returns the lexicographically least element in the coset  $G^{(i)}g$  where  $g$  is some permutation in  $G^{(i-1)}$  that maps  $i$  to  $j$ . Since we cycle through all  $i < j \leq n$  in the loop 2,  $C_i$  will be a right traversal of  $G^{(i)}$  in  $G^{(i-1)}$  at the end of loop 2. As  $D_{i-1} = D_i \cup C_i$  the loop invariant of loop 1 is maintained. Finally when  $i = 0$ ,  $D_0$  is the generator set for  $G$ .  $\square$

The following theorem is a direct consequence of Proposition 4.9 and Theorem 2.4.

**Theorem 4.10.** *The FINDGROUP problem is in  $\text{FP}^{\text{SPP}}$ .*

## 4.5 The complexity of Graph Isomorphism

We now give SPP upper bound for the Graph Isomorphism. Since  $\text{GI} \equiv_T^p \text{AUT}$  it follows from the closure properties of SPP that it is sufficient to give an  $\text{FP}^{\text{SPP}}$  algorithm for AUT. We show that AUT is a special case of the FINDGROUP problem. Without loss of generality assume that the vertex set of the graph is  $\{1, \dots, n\}$ . Assume a suitable encoding of graphs say via adjacency matrix. For encodings  $x$  of  $n$  vertex graph  $X$  let  $G_x$  be the automorphism subgroup of  $X$ . There is a polynomial time membership test for  $G_x$  as given the encoding  $x$  of a graph  $X$ , there is a polynomial time algorithm to test whether a given permutation  $g \in S_n$  is an automorphism of  $X$ . Hence AUT is a special case of FINDGROUP.

Similarly given permutation group  $G$  over  $\Omega$  and a subset  $\Sigma$  of  $\Omega$  there is a polynomial time membership test for elements of  $G_\Sigma$ . Hence the SETSTAB problem is a special case of FINDGROUP. For the hidden subgroup problem the hiding function  $\phi$  gives a membership test:  $h \in H$  if and only if  $\phi(h) = \phi(1)$ . Using Theorem 4.10 we have the main result of this chapter.

**Theorem 4.11.** *The Graph Isomorphism problem, the hidden subgroup problem over permutation groups, the set-wise stabiliser problem etc. are in SPP (or  $\text{FP}^{\text{SPP}}$  in the case of functional problems).*

## 4.6 Discussion

We have shown that the Graph Isomorphism problem is in the complexity class SPP. We proved this by shown that given a graph  $X$ , a generator set for  $\text{Aut}(X)$  can be computed by a polynomial time deterministic machine making UP-like queries to an NP language. It is still open whether the Graph Isomorphism problem is in UP.

An approach to Graph Isomorphism is via Graph Canonisation. A function  $f$  from  $\mathcal{G}(\Omega)$  to  $\mathcal{G}(\Omega)$  is a *canonising function* on graphs if it satisfies the following properties: (1) for all  $X \in \mathcal{G}(\Omega)$   $f(X)$  is isomorphic to  $X$  and (2) graphs  $X$  and  $Y$  in  $\mathcal{G}(\Omega)$  are isomorphic if and only if  $f(X) = f(Y)$ . Intuitively  $f$  pick a *canonical* element from each equivalence class. It is not difficult to see that testing for isomorphism reduces to canonisation. In fact the asymptotically fastest algorithm [75, 12] for Graph Isomorphism is through Graph Canonisation. However the best known complexity theoretic upper bound for Graph canonisation is  $\text{FP}^{\text{NP}}$ . It would be interesting to show better upper bounds for this problem.

## Chapter 5

# Bounded colour multiplicity Graph Isomorphism problem

In this chapter we study the *bounded colour multiplicity Graph Isomorphism* problem, a restricted version of vertex coloured Graph Isomorphism problem. For a finite set  $C$  of *colours*, a  $C$ -coloured graph is a triple  $X = (V, E, \psi)$  where  $V$  is the set of vertices,  $E \subseteq \binom{V}{2}$  is the set of edges and  $\psi : V \rightarrow C$  is the colouring that assigns to each vertex  $v \in V$  a *colour*  $\psi(v) \in C$ . An isomorphism  $f$  between two  $C$ -coloured graphs  $X_1 = (V_1, E_1, \psi_1)$  and  $X_2 = (V_2, E_2, \psi_2)$  if it exists, is an isomorphism from the graph  $(V_1, E_1)$  to graph  $(V_2, E_2)$  that preserves the colours, i.e. for all  $u \in V_1$ ,  $\psi_1(u) = \psi_2(f(u))$ . The Coloured Graph Isomorphism problem, for short, is to check whether two vertex coloured graphs  $X_1$  and  $X_2$  are isomorphic. Note that GI is the special case of CGI where every vertex has the same colour. On the other hand using suitable graph gadgets CGI is reducible to GI (details can be found in [33]).

We now define a restricted version of CGI, the bounded colour multiplicity Graph Isomorphism problem or BCGI for short. The colouring map  $\psi$  induces an equivalence relation on  $V$ ;  $u \sim_\psi v$  if  $\psi(u) = \psi(v)$ . A *colour class* is an equivalence class under this equivalence relation. For a colour  $c \in C$ , the  $c$ -*colour class* of  $X$  is the equivalence class  $\{v \in V : \psi(v) = c\}$ .

**Definition 5.1** (BCGI <sub>$b$</sub> ). *Given two  $C$ -coloured graphs  $X_1$  and  $X_2$  such that the size of each colour class of  $X_i$ ,  $i = 1, 2$ , is bounded by a constant  $b$ , check whether  $X_1 \cong X_2$ .*

One of the first versions of Graph Isomorphism problem that was studied using group theoretic methods is the BCGI problem. Babai gave a

randomised polynomial time algorithm for  $\text{BCGI}_b$  for each constant  $b$  [9]. This was improved to a deterministic polynomial time algorithm by Furst *et al* [28]. Subsequently, Luks [47] gave a remarkable NC algorithm.

Recently Torán [68] has proved various hardness results for Graph Isomorphism. In particular, he proved that  $\text{BCGI}_b$  is  $\text{AC}^0$ -many one hard for the logspace counting class  $\text{Mod}_k\text{L}$  for each constant  $k$ . A key step in the hardness proofs is the construction of certain graph gadgets that enables the simulation of addition modulo  $k$ . In fact, these graph gadgets can be used to prove that  $\text{BCGI}$  is hard for the entire  $\text{Mod}_k\text{L}$  hierarchy [8, Appendix].

In this chapter we prove that  $\text{BCGI}_b$  is in the  $\text{Mod}_k\text{L}$  hierarchy, where the constant  $k$  and the level of the hierarchy depends on  $b$  [8]. Together with the hardness for the  $\text{Mod}_k\text{L}$ -hierarchy, we have a fairly tight classification. Though not explicitly mentioned, it appears that Luks' NC-algorithm puts  $\text{BCGI}_b$  in  $\text{NC}^k$  where the constant  $k$  depends on  $b$  (Luks solves a more general problem and as a consequence derives the NC algorithm for  $\text{BCGI}_b$ ). Since the  $\text{Mod}_k\text{L}$  hierarchy is contained in  $\text{NC}^2$  (even  $\text{TC}^1$ ) our result is an improvement on Luks' result.

We first prove that there is a logspace Turing reduction from  $\text{BCGI}_b$  to the pointwise stabiliser problem  $\text{PWS}_c$  (definition in Section 5.1) for some constant  $c$  that depends only on  $b$ . In this sequel we often say that a function  $f$  can be computed in the  $\text{Mod}_k\text{L}$ -hierarchy if there is a logspace bounded oracle machine  $M^A$  that computes  $f$  for some language  $A$  in the  $\text{Mod}_k\text{L}$ -hierarchy. We prove in this chapter that  $\text{PWS}_c$  is in the  $\text{Mod}_k\text{L}$  hierarchy where  $k$  is the product of all primes less than  $c$ . This would imply that  $\text{BCGI}_b$  is in the  $\text{Mod}_k\text{L}$  hierarchy as  $\text{BCGI}_b \leq_T^{\log} \text{PWS}_c$ . We now define the problem  $\text{PWS}_c$  and give an outline of the  $\text{Mod}_k\text{L}$  algorithm for it.

## 5.1 The Pointwise stabiliser problem

Given a permutation group  $G$  on  $\Omega$  and a set  $\Delta \subseteq \Omega$ , recall that the pointwise stabiliser of  $\Delta$ ,  $G(\Delta)$ , is the subgroup  $\{g \in G : \delta^g = \delta \text{ for all } \delta \in \Delta\}$ . As opposed to the setwise stabiliser, the pointwise stabiliser can be computed in polynomial time (using Theorem 3.9 for example). However in this chapter we are interested in a restricted version where  $G$ -orbits are of bounded size which we show is in the  $\text{Mod}_k\text{L}$ -hierarchy. The polynomial time algorithm for the general case that uses Theorem 3.9 does not help us here because it is sequential.

**Definition 5.2** ( $\text{PWS}_c$ ). *Let  $G$  be a permutation group on  $\Omega$  such that each  $G$ -orbit is of cardinality at most  $c$ . Given a subset  $\Delta \subseteq \Omega$  compute  $G(\Delta)$ .*

Recall the permutation group theoretic formulation of the Graph Isomorphism problem from Section 4.1. We generalise this to coloured graphs in a straight forward manner: Let  $\mathcal{CG}(\Omega, C)$  denote the set of  $C$ -coloured graphs with vertex set  $\Omega$ . As before there is a natural action of the group  $\text{Sym}(\Omega)$  on  $\mathcal{CG}(\Omega, C)$ : The graph  $(V, E, \psi)$  goes to  $(V, E^g, \psi^g)$  where  $\psi^g$  is the map  $u \mapsto \psi(u^{g^{-1}})$ . For a  $C$ -coloured graph  $X \in \mathcal{CG}(\Omega, C)$ , the automorphism subgroup  $\text{Aut}(X)$  is the stabiliser of the point  $X$  under this action. It is easy to verify that the set of  $C$ -coloured graphs in  $\mathcal{CG}(\Omega, C)$  isomorphic to  $X$  is exactly the  $\text{Sym}(\Omega)$ -orbit containing  $X$ . If  $X$  and  $Y$  are two isomorphic coloured graphs in  $\mathcal{CG}(\Omega, C)$  and  $g \in \text{Sym}(\Omega)$  be such that  $X^g = Y$  then, as before, the set of all isomorphisms between  $X$  and  $Y$  are exactly  $\text{Aut}(X)g$ .

**Definition 5.3** ( $\text{AUT}_b$ ). *Given a  $C$ -coloured graph  $X$  such that each colour class is of cardinality bounded by  $b$  compute a generator set for  $\text{Aut}(X)$  as a subgroup of  $\text{Sym}(V(X))$ .*

Mathon's result (Theorem 4.3) generalises to coloured graphs as well and it can be show that the coloured graph isomorphism problem is logspace Turing reducible to coloured automorphism problem. In particular there is a logspace Turing reduction from  $\text{BCGI}_b$  to  $\text{AUT}_{2b}$ . To show that  $\text{BCGI}_b$  reduces to  $\text{PWS}_c$  for some constant  $c$  that depends on  $b$  it is therefore sufficient to give a logspace Turing reduction from  $\text{AUT}_b$  to  $\text{PWS}_c$ . We sketch the logspace reduction [47, Section 7]<sup>1</sup>.

Consider an instance  $X = (V, E, \psi)$  in  $\mathcal{CG}(V, C)$  of  $\text{AUT}_b$ . Recall that the colouring  $\psi$  of  $X$  partitions the vertex set  $V$  into disjoint colour classes  $V_1, \dots, V_m$  and for each  $i$ ,  $\#V_i \leq b$ . Consider the group  $G = \prod \text{Sym}(V_i)$  acting on  $V$ . For  $1 \leq i < j \leq m$  define the sets  $\Omega_{ij} = 2^{V_{ij}}$ , where  $V_{ij}$  denotes collection of all unordered pairs  $\{u, v\}$ ,  $u \in V_i$  and  $v \in V_j$ . The elements of set  $\Omega_{ij}$  are subsets of unordered pairs  $\{u, v\}$ ,  $u \in V_i$  and  $v \in V_j$ . In particular consider the collection  $E_{ij}$  defined as follows

$$E_{ij} = \{\{u, v\} \in E : u \in V_i \text{ and } v \in V_j\}.$$

The subsets  $E_{ij}$  are points of  $\Omega_{ij}$ . Define  $\Omega = \cup \Omega_{ij}$ . We have  $\#\Omega_{ij} \leq 2^{b^2}$  and  $\#\Omega \leq m^2 \cdot 2^{b^2}$ . The action of  $G$  on  $V(X)$  extends naturally to  $\Omega$  and hence  $G$  is a permutation group on  $\Omega$ . If  $\Delta = \{E_{ij} : 1 \leq i < j \leq m\}$  then the pointwise stabiliser of  $\Delta \subseteq \Omega$ ,  $G(\Delta)$ , is the group  $\text{Aut}(X)$ . Furthermore  $G$  maps a point in  $\Omega_{ij}$  to another point in  $\Omega_{ij}$  and hence  $G$ -orbits are of size bounded by  $c = 2^{b^2}$ . Given the instance  $X = (V, E, \psi)$  of  $\text{AUT}_b$ , a generator

<sup>1</sup>Luks gives a NC-reduction but for a more general version.

set of  $G$  as a permutation group on  $\Omega$  can be computed in FL. Hence the following proposition.

**Proposition 5.4.** *There is a logspace reduction from  $AUT_b$  to  $PWS_c$  where  $c \leq 2^{b^2}$ . Hence if for all constants  $c$  there is a constant  $k$  that depends only on  $c$  such that  $PWS_c$  is in the  $\text{Mod}_k\text{L}$ -hierarchy then for all constants  $b$  there is a constant  $k'$  that depends only on  $b$  such that  $BCGI_b$  and  $AUT_b$  are in the  $\text{Mod}_{k'}\text{L}$ -hierarchy.*

In the rest of the chapter we prove that  $PWS_c$  is in the  $\text{Mod}_k\text{L}$ -hierarchy. We give an outline of the strategy. The key step in our algorithm is what we call “target reduction”. Given an instance  $(G, \Omega, \Delta)$  of  $PWS_c$  we compute a subgroup  $G'$  of  $G$  such that

1.  $G \geq G' \geq G(\Delta)$ .
2. For every  $G$ -orbit  $\Sigma$  containing a point of  $\Delta$ ,  $G'|_{\Sigma}$  is a proper subgroup of  $G|_{\Sigma}$ .

We prove that target reduction can be performed in the  $\text{Mod}_k\text{L}$ -hierarchy where  $k$  is the product of all primes less than  $c$ .

We now argue that the target reduction procedure can be used to compute the pointwise stabiliser. Starting with  $G$ , by applying the target reduction procedure compute a subgroup  $G'$  which is strictly smaller than  $G$  on each of the orbits that contain points of  $\Delta$ . Since  $G \geq G' \geq G(\Delta)$ ,  $G(\Delta) = G'(\Delta)$ . Moreover for each  $G$ -orbit  $\Sigma$  such that  $\Sigma \cap \Delta \neq \emptyset$ , the projection  $G'|_{\Sigma}$  is a proper subgroup of  $G|_{\Sigma}$  and hence  $\#G'|_{\Sigma} \leq \frac{1}{2}\#G|_{\Sigma}$ . We then repeat the target reduction procedure with  $G$  replaced by  $G'$ . Since  $G$ -orbits are of size bounded by a constant  $c$ , after  $O(c \cdot \log c)$  iterations of the target reduction step we converge to  $G(\Delta)$ . Thus if the target reduction step is in the  $l$ th level of the  $\text{Mod}_k\text{L}$ -hierarchy then  $PWS_c$  can be solved in the  $l \cdot c \cdot \log c$  level of the  $\text{Mod}_k\text{L}$ -hierarchy. The detailed description of the target reduction procedure is given in Section 5.5.

For the target reduction procedure we require a special strong generating set for  $G$ . We consider a special normal series  $G = N_0 \triangleright \dots \triangleright N_l = 1$  of length  $l$  bounded by a constant that depends only on  $c$  such that each of the quotient group  $N_i/N_{i+1}$  is  $T_i$ -semisimple for some simple group  $T_i$ . Using this normal series we compute a strong generator set  $C$  of  $G$ . The computation of the strong generator set  $C$  proceeds in  $l$  stages. Each of this stage involves solving a certain normal closure problem for which we give a  $\text{FL}^{\text{Mod}_k\text{L}}$  algorithm. The detailed procedure for computing the strong generator set  $C$  is described in Section 5.4.

For computing the strong generator set  $C$  and for the target reduction procedure we require some more group theory. The two important group theoretic concepts we require is (1) the socle and (2) residual series of a group. The normal series  $G = N_0 \supseteq \dots \supseteq N_l = 1$  which is used to compute the strong generator set  $C$  is obtained by patching up the residue series of each of the constant sized groups  $G_i$ . The target reduction procedure makes use of the O’Nan-Scott theorem (Theorem 5.9), a result on the structure of socles of primitive permutation groups. In the next two sections we develop the group theory required for this chapter.

## 5.2 Characteristic subgroups and Socles

In this section we develop some more group theory relevant for this chapter. Most of the group theory that we require, albeit in a slightly different form, is developed by Luks [47] for his NC-algorithm.

**Definition 5.5** (Characteristic subgroup). *A subgroup  $H$  of a finite group  $G$  is a characteristic subgroup if all automorphisms of  $G$  maps  $H$  to itself.*

In this context, notice that a normal subgroup of  $G$  is a subgroup that is invariant under inner automorphisms of  $G$  whereas a characteristic subgroup is invariant under all automorphisms. Hence characteristic subgroups are normal subgroups. For a characteristic subgroup  $R$  of  $G$ , the restriction of any  $G$ -automorphism to  $R$  is an  $R$ -automorphism. The following proposition directly follows from the above discussion.

**Proposition 5.6.**

1. *If  $R_1$  is a characteristic subgroup of  $G$  and  $R_2$  is a characteristic subgroup of  $R_1$  then  $R_2$  is also a characteristic subgroup of  $G$ .*
2. *If  $R_1$  and  $R_2$  be characteristic subgroups of  $G$  then so is  $R_1R_2$  and  $R_1 \cap R_2$ .*
3. *Let  $R_1$  and  $R_2$  be normal subgroups of  $G$  such that  $R_1 \cap R_2 = \{1\}$ . If  $R_1R_2$  and  $R_1$  are characteristic subgroups of  $G$  then so is  $R_2$ .*

The entire group  $G$  and the trivial subgroup  $\{1\}$  are characteristic subgroups of  $G$ . The centre  $C_G(G)$  of  $G$ , the subgroup of elements of  $G$  that commute with all elements of  $G$ , is also a characteristic subgroup of  $G$ .

Let  $G$  be a nontrivial finite group. By a *minimal normal subgroup* of  $G$  we mean a normal subgroup  $N \trianglelefteq G$  different from  $\{1\}$  which is minimal

in the containment order, i.e. there is no proper subgroup of  $N$  other than  $\{1\}$  that is normal in  $G$ . For a simple group  $T$ , the only minimal normal subgroup is  $T$  itself. We now state an important lemma about minimal normal subgroups of a group  $G$  ([24, Theorem 4.3A]).

**Lemma 5.7.** *Let  $G$  be any group and let  $K$  be a minimal normal subgroup of  $G$ . Then  $K = T_1 \times \dots \times T_n$  where  $T_i$ 's are all isomorphic to a simple group  $T$  (i.e.  $K$  is  $T$  semisimple). Moreover for any  $i$  and  $j$  there is an element  $g \in G$  such that  $T_i = g^{-1}T_jg$ .*

Having defined the minimal normal subgroup we define the *socle* of a group  $G$ , an important characteristic subgroup of  $G$ .

**Definition 5.8** (Socle). *For a finite group  $G$  the socle  $\text{Soc}(G)$  is the subgroup generated by the set of all minimal normal subgroups of  $G$ .*

Clearly any automorphism of  $G$  maps minimal normal subgroups to minimal normal subgroups and hence fixes the socle. Therefore  $\text{Soc}(G)$  is a characteristic subgroup of  $G$ . We now state a restricted version of the O'Nan-Scott theorem, a theorem on the structure of socles of primitive permutation groups, suitable for our purposes. A complete statement of the theorem (Theorem 4.1A of [24]), its proof and its applications to the study of permutation groups can be found in Chapter 4 of the book by Dixon and Mortimer [24].

**Theorem 5.9** (O'Nan-Scott theorem). *Let  $G$  be a primitive permutation group on  $\Omega$  with socle  $\text{Soc}(G) = K$ . Then  $K$  is transitive and  $T$ -semisimple for some simple group  $T$ . Furthermore exactly one of the following is true for  $K$ .*

1.  $K$  is abelian in which case  $K$  is elementary abelian and regular on  $\Omega$ . Also  $K$  is the unique minimal normal subgroup of  $G$ . For an  $\alpha \in \Omega$ , the group  $K_\alpha$  is the trivial group  $\{1\}$ .
2.  $K$  is nonabelian and is the unique minimal normal subgroup of  $G$ . For  $\alpha \in \Omega$ ,  $K_\alpha$  is a proper subgroup of  $K$ .
3.  $K$  is nonabelian and is a product  $K = K_1 \times K_2$ , where  $K_1$  and  $K_2$  are isomorphic. The subgroups  $K_1$  and  $K_2$  are the only minimal normal subgroups of  $G$  and each  $K_i$  is regular on  $\Omega$ . Furthermore the centraliser of  $K_1$  in  $G$ ,  $C_G(K_1)$ , is  $K_2$  and vice-versa. For  $\alpha \in \Omega$ ,  $K_\alpha$  is a diagonal subgroup of  $K_1 \times K_2$ .

### 5.3 Residues and Residual Series

In the previous section we studied an important characteristic subgroup, the socle. Let  $G$  be a finite group. For any simple group  $T$ , we associate a characteristic subgroup of  $G$  called its  $T$ -residue.

**Definition 5.10** (Residue subgroup). *Let  $T$  be a finite simple group. For a group  $G$  we say that the normal subgroup  $N$  is a  $T$ -residue of  $G$  if  $G/N$  is  $T$ -semisimple and for all  $H \trianglelefteq G$  contained in  $N$ ,  $G/H$  is  $T$ -semisimple if and only if  $H = N$ .*

To prove that  $T$ -residues are unique, we require the following two lemmas on normal subgroups of semisimple groups.

**Lemma 5.11.** *Let  $G$  be a semisimple group with a normal subgroup  $H$ . Then  $G = L \times H$  for some normal subgroup  $L$  of  $G$ . Moreover  $G/H$  is also semisimple.*

*Proof.* Let  $G$  be  $T$ -semisimple. Depending on whether  $T$  is abelian or not we have two cases.

**$T$  is abelian** In this case  $T = \mathbb{F}_p$  for some prime  $p$ . The group  $G$  is therefore a vector  $V$  over  $\mathbb{F}_p$ . The subgroup  $H$  corresponds to a subspace  $W$  of  $V$ . We can decompose  $V$  as the direct sum  $V = W \oplus W'$ . The required group  $L$  is the subspace  $W'$ . Clearly  $G/H$  is isomorphic to the subspace  $L$  and is hence  $\mathbb{F}_p$ -semisimple.

**$T$  is nonabelian** Let  $G = T_1 \times \dots \times T_k$  where each  $T_i$  is isomorphic to  $T$ . Firstly the projection  $H_i$  of  $H$  on any of the group  $T_i$  is either trivial or the full group  $T_i$ . Otherwise  $H_i$  will be a nontrivial normal subgroup of  $T_i$  which contradicts the fact that  $T_i$  is simple. Thus we assume, with out loss of generality, that there is an integer  $l \leq k$  such that  $H$  projects onto each of the group  $T_i$  for  $1 \leq i \leq l$  and is trivial on  $T_j$  for  $l < j \leq k$ . By Scott's Lemma  $H$  is a product of diagonals of the groups  $\{T_i\}_{1 \leq i \leq l}$ . Consider any two indices  $i, j \leq l$ . Any diagonal group  $\text{Diag}(T_i \times T_j)$  is not a normal subgroup of  $T_i \times T_j$ . To see this consider an element  $\langle a, \psi(a) \rangle \in \text{Diag}_\psi(T_1 \times T_2)$  for some isomorphism  $\psi : T_i \rightarrow T_j$ . Let  $b$  be any element of  $T_i$  that does not commute with  $a$  then  $\langle 1, \psi(b) \rangle^{-1} \langle a, \psi(a) \rangle \langle 1, \psi(b) \rangle = \langle a, \psi(b^{-1}ab) \rangle \notin \text{Diag}_\psi(T_i \times T_j)$ . Therefore  $H$  is exactly the subgroup  $T_1 \times \dots \times T_l$ . The required group  $L$  is  $T_{l+1} \times \dots \times T_k$ . Clearly  $G/H = L$  is  $T$ -semisimple.  $\square$

The next lemma follows directly from Lemma 5.11 (consider the semisimple group  $G/N$  and its normal subgroup  $H/N$ ).

**Lemma 5.12.** *Let  $G$  be any group with a normal subgroup  $N$  such that  $G/N$  is semisimple. Let  $H$  be any subgroup of  $G$  containing  $N$ . Then there is a normal subgroup  $L$  of  $G$  containing  $N$  such that  $G = LH$  and  $L \cap H = N$ .*

We now prove that for any simple group  $T$ , the  $T$ -residue is unique. This is a slightly weaker version of Lemma 6.2 stated in Luks [47] and is sufficient for our purpose. The proof of the more general version [47, Lemma 6.2] is along similar lines.

**Lemma 5.13 (Luks).** *Let  $G$  be any finite group. For any simple group  $T$  there is a unique  $T$  residue, i.e. there is a normal subgroup  $N$  of  $G$  such that  $G/N$  is  $T$ -semisimple and for any  $H \trianglelefteq G$  such that  $G/H$  is  $T$ -semisimple,  $H$  contains  $N$ .*

*Proof.* The proof is via induction on the order of  $G$ . Firstly, if  $G$  itself is  $T$ -semisimple then lemma is clearly true; the unique  $T$ -residue is  $\{1\}$ . This is the base case of our induction.

We assume the assertion to be true for all groups of order less than  $k$ . Consider a group  $G$  of order  $k$ . If possible, let  $N_1$  and  $N_2$  be two distinct  $T$ -residues of  $G$ . Let  $H = N_1 \cap N_2$  and  $N = N_1 N_2$ . From the minimality of  $N_i$ 's it follows that  $H$  is a strict subgroup of  $N_i$  for  $i = 1, 2$ . We have two cases.

*Case 1 ( $H \neq \{1\}$ ):* In this case  $G/H$  is a group of smaller cardinality than  $G$  and hence, by induction hypothesis, has a unique  $T$ -residue  $L/H$  for some normal subgroup  $L$  of  $G$  containing  $H$ . Since  $N_i/H \trianglelefteq G/H$  and  $\frac{G/H}{N_i/H} \cong G/N_i$  is  $T$ -semisimple,  $N_i/H$  contains  $L/H$ . Therefore  $N_i$  contains  $L$  for  $i = 1, 2$ . Therefore  $L \subseteq N_1 \cap N_2 = H$  and since  $L$  contains  $H$ ,  $L = H$ . However  $G/L \cong \frac{G/H}{L/H}$  and hence is  $T$ -semisimple. This contradicts the minimality of  $N_i$ 's.

*Case 2 ( $H = \{1\}$ ):* We prove that in this case  $G$  itself is  $T$ -semisimple. Firstly,  $N = N_1 N_2 = N_1 \times N_2$ . Hence the subgroup  $N_1$  is isomorphic to  $N_1 N_2 / N_2$  and since  $N_1 N_2 / N_2 \trianglelefteq G / N_2$ , is itself  $T$ -semisimple (Lemma 5.11).

Consider the group  $G$  with normal subgroup  $N_2$ . The quotient group  $G/N_2$  is  $T$ -semisimple and  $N$  is a normal subgroup of  $G$  containing  $N_2$ . Using Lemma 5.12 we have a normal subgroup  $L$  of  $G$  such that  $L \cap N = N_2$  and  $G = LN$ . Since  $N = N_1 \times N_2$  and  $L \geq N_2$  it follows that  $G = LN_1$ . But  $L \cap N_1 = \{1\}$  and hence  $G = L \times N_1$ .

Having proved that  $G = N_1 \times L$  it is easy to see that  $L$  itself is  $T$ -semisimple. This is because  $L$  is isomorphic to  $G/N_1$  which is  $T$ -semisimple. Hence  $G = N_1 \times L$  is  $T$ -semisimple. This however contradicts the minimality of  $N_1$  and  $N_2$  as the unique  $T$ -residue of  $G$  is  $\{1\}$ .  $\square$

In view of Lemma 5.13, we use  $\text{Res}_T(G)$  to denote the unique  $T$ -residue of  $G$ . For any simple group  $T$  since the  $T$ -residue of  $G$  is unique, any  $G$ -automorphism has to map  $\text{Res}_T(G)$  to itself. Hence  $\text{Res}_T(G)$  is a characteristic subgroup of  $G$ . Based on residues, we can define an important normal series called the *residual series*.

**Definition 5.14** (Residual series). *A residual series of  $G$  is a series  $G = R_0 \triangleright \dots \triangleright R_l = \{1\}$  where for all  $1 \leq i \leq l$ ,  $R_i = \text{Res}_{T_i}(R_{i-1})$  for some simple group  $T_i$ .*

In fact from Proposition 5.6 it follows that the residual series is a series of characteristic subgroups. We now prove an important property of residual series of primitive permutation group due to Luks [47, Lemma 6.3].

**Lemma 5.15** (Luks). *Let  $G$  be a primitive permutation group acting on  $\Omega$  and let  $G = R_0 \triangleright \dots \triangleright R_t = \{1\}$  be any residual series then the last nontrivial subgroup in the series, is the socle of  $G$ .*

*Proof.* We assume that  $t > 1$  for otherwise  $G$  itself is  $T$ -semisimple, hence is its own socle and we are through.

Let  $S$  be the socle of  $G$ . The group  $G$  being primitive, it follows from the O’Nan-Scott theorem that the socle  $S$  and hence all the minimal normal subgroups of  $G$  are  $T$ -semisimple for some simple group  $T$ .

First let us suppose that  $R_{t-1}$  does not contain  $S$ . Since  $R_{t-1}$  is a normal subgroup of  $G$  there is a minimal normal subgroup  $K$  of  $G$  that is contained in  $R_{t-1}$ . This rules out cases 1 and 2 of the O’Nan-Scott theorem as in those cases  $G$  has a unique normal subgroup which is also the socle  $S$ . Thus  $G$  has exactly two minimal normal subgroups  $K_1$  and  $K_2$ ,  $S = K_1 \times K_2$  and  $R_{t-1}$  contains one of them say  $K_1$ . Let  $s$  be the largest index  $i$  such that  $R_i$  contains  $S$ . Clearly  $s < t - 1$  and  $R_{s+1} \geq R_{t-1} \neq 1$ . Moreover  $R_{s+1}$  contains  $K_1$  but not  $K_2$ .

The group  $R_s/R_{s+1}$  is semisimple and  $R_{s+1}S$  is a normal subgroup of  $R_s$  containing  $R_{s+1}$ . Therefore by Lemma 5.12 we have a subgroup  $L$  of  $R_s$  such that  $R_s = LR_{s+1}S$  and  $L \cap R_{s+1}S = R_{s+1}$ . However since  $L$  contains  $R_{s+1}$  and hence  $K_1$  it follows that  $R_s = LK_2$ . Furthermore  $L \cap K_2 = 1$ . Thus  $R_s = L \times K_2$  and every element of  $L$  commutes with  $K_2$ . This is possible only if  $L = K_1$  as by the O’Nan-Scott theorem  $C_G(K_2) = K_1$ .

For a  $T$ -semisimple group  $G$ ,  $\text{Res}_{T'}(G)$  is either 1 or the whole of  $G$  depending on whether  $T' = T$  or not. We have proved that if  $R_{t-1}$  does not contain the socle  $S$  then  $R_s = S$  which is  $T$ -semisimple by the O'Nan-Scott theorem. Since  $R_{s+1}$  is a proper subgroup of  $R_s$  it follows that  $R_{s+1} = \text{Res}_T(R_s) = 1$ . This however contradicts the fact that  $R_{s+1} \geq R_{t-1} \neq 1$ . Hence  $R_{t-1}$  contains the socle  $S$ .

Having proved that  $R_{t-1}$  contains  $S$  it is easy to see that  $R_{t-1}$  is indeed the socle. The group  $R_{t-1}$  being  $T$ -semisimple there is a subgroup  $L$  of  $R$  such that  $L \times S = R_{t-1}$  (Lemma 5.11). It follows from Proposition 5.6 that  $L$  is a characteristic subgroup of  $G$ . This is not possible unless  $L$  is the trivial group otherwise there is a minimal normal subgroup  $K$  of  $G$  contained in  $L$  and  $K \leq S \cap L$ . □

## 5.4 Strong generator set revisited

Recall that for every decreasing tower of groups  $G = G_0 \geq \dots \geq G_t = \{1\}$  we can associate a generator set called the strong generator set. We now generalise this to relative strong generator set. Let  $H$  be a subgroup of  $G$  and let  $G = G_0 \geq \dots \geq G_t = H$  be a decreasing sequence of groups from  $G$  to  $H$ . Let  $C_i$  denote the coset representatives of  $G_i$  in  $G_{i-1}$ . Then the set  $C = \cup C_i$  is called a *strong generator set* of  $G$  relative to  $H$ , SGS of  $G$  rel  $H$  for short. For any element  $g \in G$  there is a unique  $h$  in  $H$  such that  $g = g_1 \dots g_t h$  where  $g_i \in C_i$ . By sift of  $g$  with respect to the strong generator set  $C$  we mean this  $h$ . We will use  $\text{Sift}(g)$  to denote the sift of  $g$  with respect the strong generator set  $C$ . The sift of an element is not unique and depends on the choice of the coset representatives  $C_i$ .

A *semisimple series* from  $G$  to a normal subgroup  $N$  is a normal series  $G = N_0 \triangleright \dots \triangleright N_t = N$  where the quotient groups  $N_i/N_{i+1}$  are  $T_i$ -semisimple for simple groups  $T_i$ . We associate a strong generator set for such a series. Let the quotient group  $N_i/N_{i+1}$  be  $\prod_j T_{ij}$  where each  $T_{ij}$  is isomorphic to  $T_i$ . Consider a normal series (normal in  $N_i$ ) given by  $N_i = N_{i,0} \triangleright \dots \triangleright N_{i,n_i} = N_{i+1}$  where  $N_{i,s}/N_{i+1}$  is the group  $\prod_{j>s} T_{ij}$ . Let  $C_{ij}$  be the right (or left) traversal of  $N_{i,j}$  over  $N_{i,j+1}$ . Then  $C = \cup_{i,j} C_{i,j}$  forms a strong generator set for  $G$  rel  $N$  with respect to the subnormal series  $\{N_{i,j}\}$ .

We are interested in permutation group  $G$  over  $\Omega$  with bounded orbits. The simple groups  $\{T_i\}_{0 \leq i < t}$  that occur will all be of order bounded by a constant and the semisimple series which we construct for  $G$  will be of bounded length. Furthermore, the computation of strong generator set  $C$  is

done inductively by computing the strong generator set of  $G$  relative to  $N_i$  starting with  $i = 0$ . The fact that the series  $\{N_i\}_{i=1}^t$  is of bounded length is important for the  $\text{Mod}_k\text{L}$ -hierarchy upper bound. Hence in this context it is more natural to associate the semisimple series  $\{N_i\}_{i=1}^t$  to the strong generator set  $C$  than the subnormal series  $\{N_{i,j}\}$ .

We now prove a property analogues to Proposition 3.2 of Luks and McKenzie [50].

**Proposition 5.16.** *Given a group  $G$  via a generator set  $A$ . Let  $G = N_0 \supseteq \dots \supseteq N_t = N$  be a semisimple series from  $G$  to  $N$  and let  $C = \cup_{ij} C_{ij}$  be the associated strong generator set of  $G$  relative to  $N$ . Let  $S$  be the set containing the following elements:*

1.  $\text{Sift}(g)$  for all  $g \in A$ .
2.  $\text{Sift}(x^{-1}yx)$  for all  $x \in C_{ij}$  and  $y \in C_{lm}$ ,  $(i, j) < (l, m)$ .
3.  $\text{Sift}(xy)$  for all  $x, y \in C_{ij}$  for all  $i$  and  $j$ .

Then the normal closure  $\text{NCL}_G(S)$  of  $S$  in  $G$  is  $N$ .

*Proof.* The proof is similar to that of Proposition 3.2 of Luks and McKenzie [50]. The set  $S$  is clearly a subset of  $N$  and since  $N$  is a normal subgroup of  $G$  we have  $\text{NCL}_G(S) \leq N$ . To prove the converse consider any element  $h \in N$ . There exists elements  $y_1, \dots, y_m$  in  $A$  such that  $h = y_1 \dots y_l$ . For ease of notation we assume that  $l = 2$  and  $h = xy$  for  $x, y \in A$ . The general case is similar. Since  $S$  contains the sifts of all the elements of  $A$  there exists  $x_{ij} \in C_{ij}$  and  $y_{lm} \in C_{lm}$  such that  $x = \prod_{i,j} x_{ij} s_1$  and  $y = \prod_{l,m} y_{lm} s_2$  where  $s_1$  and  $s_2$  are elements of  $S$  and hence  $\text{NCL}_G(S)$ . Hence  $h$  is given by

$$h = \left( \prod_{ij} x_{ij} \right) s_1 \left( \prod_{lm} y_{lm} \right) s_2. \quad (5.1)$$

We prove that  $h$  can be written as  $\prod_{ij} z_{ij} s$  where  $s \in \text{NCL}_G(S)$ . The first task is to push down  $s_1$  to the end. For any  $y \in G$  since  $\text{NCL}_G(S)$  is normal subgroup of  $G$  we have  $y\text{NCL}_G(S) = \text{NCL}_G(S)y$  and therefore whenever we have a product of the form  $h = \dots sy \dots$ ,  $s \in \text{NCL}_G(S)$  and  $y \in C$ , we can replace it with  $h = \dots ys^* \dots$  for some  $s^* \in \text{NCL}_G(S)$ .

For products of the form  $h = \dots yx \dots$  where  $x \in C_{ij}$  and  $y \in C_{lm}$  with  $(i, j) < (l, m)$ , since  $S$  contains  $\text{Sift}(x^{-1}yx)$  we can rewrite it as

$$h = \dots yx \dots = \dots x \left( \prod_{(r,t) > (i,j)} u_{rt} \right) s \dots, u_{rt} \in C_{rt}. \quad (5.2)$$

Similarly when  $h = \dots xy \dots$  where  $x, y \in C_{ij}$ , since  $S$  contains  $\text{Sift}(xy)$  we can rewrite  $h$  as

$$h = \dots xy \dots = \dots z \left( \prod_{r>i} \prod_t u_{rt} \right) s \dots, u_{rt} \in C_{rt}. \quad (5.3)$$

By repeatedly rewriting the expression of  $h$  in Eq. 5.1 using Eqs. 5.2 and 5.3, we have  $h = (\prod_i \prod_j z_{ij})s$  for some  $s \in \text{NCL}_G(S)$ . However since  $h$  is in  $N$ , we have  $z_{ij} = 1$  for all  $i$  and  $j$ . Therefore  $h = s$  and hence  $h \in \text{NCL}_G(S)$ .  $\square$

### 5.4.1 Computing the strong generator set

We are given a generator set  $A$  for a permutation group  $G$  on  $\Omega$  with orbits of size bounded by a constant  $c$ . We will find the strong generator set for  $G$  with respect to a semisimple series of length bounded by a constant that depends only on  $c$ . The semisimple series which we consider is similar to the residual series of  $G$ .

Consider a permutation group  $G$  on  $\Omega$  with orbits  $\Omega_1, \dots, \Omega_m$  all of size bounded by a constant  $c$ . Let  $G_i$ 's be the projection of  $G$  onto  $\Omega_i$ . Then  $G_i$ 's are all of order bounded by  $c!$  as  $\#\Omega \leq c$ . Let  $\mathcal{T} = \{T_1, \dots, T_k\}$  be the collection of all simple groups of order at most  $c!$  then  $k = \#\mathcal{T}$  is a constant for us that depends only on  $c$ . For  $1 \leq i \leq m$  define a  $k$  length normal series  $G_i = R_{i,0} \supseteq \dots \supseteq R_{i,k}$  where  $R_{i,s} = \text{Res}_{T_s}(R_{i,s-1})$ . The group  $R_{i,k}$  is a proper subgroup of  $G_i$  as there exists a normal subgroup  $H_i$  of  $G_i$  such that  $G_i/H_i$  is simple and isomorphic to some  $T_s$ . Repeat this process starting with  $R_{i,k}$  in place of  $G_i$ . We would have to repeat this at most  $c \cdot \log c$  times before we hit the trivial group  $\{1\}$ . Thus, for each  $1 \leq i \leq m$ , we have a residual series  $G_i = R_{i,0} \supseteq \dots \supseteq R_{i,l} = \{1\}$  where the constant  $l$  depends only on  $c$ . Let  $R_s$  denote the product group  $R_s = \prod_i R_{i,s}$  then  $R_0 \supseteq \dots \supseteq R_l$  is a residual series for the product group  $\prod_i G_i$ . Since for  $1 \leq i \leq m$  the group  $G_i$  is of order less than  $c!$  in FL we compute the groups  $R_{i,s}$  and hence the product groups  $R_s$  for each  $1 \leq s \leq l$ .

Let  $N_s \trianglelefteq G$  be the normal subgroup  $G \cap R_s$  then  $G = N_0 \supseteq \dots \supseteq N_l = \{1\}$  is a semisimple series for  $G$  as  $N_i/N_{i+1} = (G \cap R_i)/(G \cap R_{i+1}) \hookrightarrow R_i/R_{i+1}$  via the map  $xN_{i+1} \mapsto xR_{i+1}$ . We prove the following important property due to Luks [47, Lemma 6.4].

**Proposition 5.17** (Luks). *Let  $H \leq \text{Sym}(\Omega)$  be any subgroup of the product  $\prod_i G_i$ . For all  $i$ , if  $H|_{\Omega_i} = G_i$  then  $H \cap R_s|_{\Omega_i} = R_{i,s}$ ,  $1 \leq s \leq l$ .*

*Proof.* Let  $\psi$  denote the homomorphism that restricts an element of the product group  $R_0 = \prod_i G_i$  to its action on  $\Omega_i$ . Fix an  $s$ . Let  $L$  and  $M$  be the groups  $H \cap R_s$  and  $H \cap R_{s+1}$  respectively. The groups  $H \cap R_s|_{\Omega_i}$  and  $H \cap R_{s+1}|_{\Omega_i}$  are  $\psi(L)$  and  $\psi(M)$  respectively.

First we prove that  $\psi(M)$  is a normal subgroup of  $\psi(L)$  and the quotient group  $\psi(L)/\psi(M)$  is  $T$ -semisimple. As  $L \leq R_s$  and  $M = L \cap R_{s+1}$  the map  $gM \mapsto gR_{s+1}$  is an embedding of  $L/M$  into  $R_s/R_{s+1}$ . The quotient group  $L/M$  is thus  $T$ -semisimple. Let  $K$  be the kernel of the map  $\psi$  in  $L$ . Consider the normal subgroup  $MK$  of  $L$ . Since  $\psi$  maps  $K$  to 1 it follows that  $\psi(M) = \psi(MK)$ . However  $MK$  is a normal subgroup of  $L$  containing  $K$  and hence  $\psi(MK) = \psi(M)$  is a normal subgroup of  $\psi(L)$ . The quotient group  $\psi(L)/\psi(M)$  is thus  $\frac{L/K}{MK/K} = L/MK$ . However  $L \supseteq MK \supseteq M$  is a normal series with  $L/M$  being  $T$ -semisimple. The group  $MK/M$  is a normal subgroup of the semisimple group  $L/M$ . Hence by Lemma 5.11  $L/MK \cong \frac{L/M}{MK/M}$  is also  $T$ -semisimple. We have thus proved that  $\psi(M)$  is a normal subgroup of  $\psi(L)$  and the quotient group  $\psi(L)/\psi(M)$  is  $T$ -semisimple. If  $\psi(L)$  is  $R_{i,s}$  then this is impossible unless  $\psi(M)$  is  $R_{i,s+1}$ . Let  $\psi(H) = \psi(H \cap R_1) = G_i = R_{i,0}$ . Assume that  $\psi(H \cap R_s) = R_{i,s}$  for some  $s$ . Then we have just proved that  $\psi(H \cap R_{s+1}) = R_{i,s+1}$ . Now repeat the argument with  $s$  replaced by  $s+1$ . As result we have  $\psi(H \cap R_j) = R_{i,j+1}$  for all  $1 \leq j \leq l$ . This completes the proof.  $\square$

In particular, Proposition 5.17 proves that for all  $s$ ,  $N_s|_{\Omega_i}$  is  $R_{i,s}$ . Thus for any  $G$ -orbit  $\Sigma$ ,  $G|_{\Sigma} = N_0|_{\Sigma} \supseteq \dots \supseteq N_l|_{\Sigma}$  is a residual series for  $G_i$ . Hence we call this series a *locally residual* series. We show that a strong generator set for  $G$  with respect to this locally residual generator set can be computed in the  $\text{Mod}_k\text{L}$ -hierarchy. A property which we use repeatedly is the following:

**Proposition 5.18.** *Let  $N$  and  $K$  be two normal subgroups of  $G$  such that  $N \geq K$ . Let  $C$  and  $D$  be the strong generator set of  $G$  relative to  $N$  and  $N$  relative to  $K$  respectively. Then  $C \cup D$  gives a strong generator set of  $G$  relative to  $K$ .*

Firstly, since each of the groups  $G_i$  are constant sized, the residual series  $G_i = R_{i,0} \supseteq \dots \supseteq R_{i,l} = \{1\}$  for each  $G_i$  can be computed separately in logspace. We prove by induction on  $i$  that an SGS  $A_i$  of  $G$  rel  $N_i$  can be computed in the  $i$ th level of the  $\text{Mod}_k\text{L}$ -hierarchy where  $k$  is the product of all primes less than  $c$ . In addition, we prove inductively that given  $g \in G$ ,  $\text{Sift}(g)$  with respect to  $A_i$  can also be computed in  $i$ th level of the  $\text{Mod}_k\text{L}$ -hierarchy. This sifting procedure is required for our induction step.

To begin with we know the strong generator set of  $G$  relative to  $N_0$ . Assuming we have already computed the strong generator set  $A_i$  of  $G$  relative to  $N_i$ . Using the sifting procedure for  $A_i$  as an oracle, we compute a set  $S$  such that  $\text{NCL}_G(S) = N_i$  (Proposition 5.16). To complete the induction we give  $\text{FL}^{\text{Mod}_k\mathbb{L}}$  algorithms for the following.

- (1) Given  $S$  and the SGS of  $G$  rel  $N_s$  compute the strong generator set  $C$  of  $N_s$  rel  $N_{s+1}$ .
- (2) Given  $x \in N_s$  compute  $\text{Sift}(x)$  with respect to the SGS  $C$ .

Depending on whether  $N_s/N_{s+1}$  is abelian or not we have two cases. If  $N_s/N_{s+1}$  is abelian then it is  $\mathbb{F}_p$ -semisimple for some prime  $p$ . We prove that in this case both (1) and (2) can be done in  $\text{FL}^{\text{Mod}_p\mathbb{L}}$ . On the other hand when  $N_s/N_{s+1}$  is non-abelian we prove that both (1) and (2) can be done in FL.

### Computing the strong generating set: nonabelian case

Let  $L_i$  and  $M_i$  denote the group  $R_{i,s}$  and  $R_{i,s+1}$  respectively. Let  $L$  and  $M$  be the product groups  $R_s = \prod_{i=1}^m L_i$  and  $R_{s+1} = \prod_{i=1}^m M_i$ . Then  $N_s$  and  $N_{s+1}$  are the  $G \cap L$  and  $G \cap M$  respectively. Our task is to compute the strong generator set of  $N_s$  rel  $N_{s+1}$  for which we give an FL algorithm.

The group  $L/M$  is  $T$ -semisimple as each  $L_i/M_i$  is  $T$ -semisimple. Consequently,  $L/M$  is of the form  $T_1 \times \dots \times T_r$  where  $T_i \cong T$  for all  $1 \leq i \leq r$ . The quotient group  $N_s/N_{s+1}$  can be faithfully embedded into  $\prod_{i=1}^m L_i/M_i$  via the map  $xN_s \mapsto xM$  and hence can be seen as a subgroup of  $L/M$ . Furthermore since  $N_s$  projects onto  $L_i$  for  $1 \leq i \leq m$  (Proposition 5.17), by Scott's Lemma we know that  $N_s/N_{s+1}$  is a product of diagonal groups of  $T_1 \times \dots \times T_r$ , i.e. there is a partition  $\mathcal{I} = \{I_1, \dots, I_s\}$  of indices  $\{1, \dots, r\}$  such that

$$N_s/N_{s+1} = \prod_{i=1}^s \text{Diag} \left( \prod_{j \in I_i} T_j \right).$$

Let  $\phi_i : L \mapsto T_i$  be the homomorphism obtained by composing the natural quotient homomorphism from  $L$  to  $L/M$  and the projection map to  $T_i$ . Fix an index  $i_j \in I_j$  for each  $I_j$ . Since  $\phi_i$  restricted to  $N_s$  is onto (because  $N_s$  projects onto  $L_i$ ) for each  $x \in T_{i_j}$  one can associate a permutation  $x^*$  in  $N_s$  such that  $\phi_{i_j}(x^*) = x$  and for all  $i$  not in  $I_j$ ,  $\phi_i(x^*)$  is identity. Let  $B_j$  be the set of such  $x^*$  one for each  $x \in T_{i_j}$ . The set  $\cup_j B_j$  gives strong generator set of  $N_s$  rel  $N_{s+1}$ . We will show that this strong generator set for  $N_s$  rel

$N_{s+1}$  can be computed in FL. To this end we prove that the following can be computed in FL.

1. The partition  $\mathcal{I}$ .
2. The collection of sets  $\{B_j\}_j$
3. Sifts of  $g \in N_s$  with respect to the SGS  $\cup_t B_t$ .

**Computing  $\mathcal{I}$ :** For indices  $i$  and  $j$  we say that  $i$  is *linked* to  $j$  if  $i$  and  $j$  falls in the same partition. Clearly  $i$  and  $j$  are linked if and only if  $N_s/N_{s+1}$  restricted to  $T_i \times T_j$  is a diagonal group  $\text{Diag}(T_i \times T_j)$ . The relation  $i \sim j$  if  $i$  is linked to  $j$ , is an equivalence relation and the equivalence classes give the partition  $\mathcal{I}$ . Consider an undirected graph  $\mathcal{G}$  with vertex set  $V = \{1, \dots, r\}$  and edge set  $\{\{i, j\} : i \text{ and } j \text{ are linked}\}$ . Each connected component  $\mathcal{C}_k$  in  $\mathcal{G}$  corresponds to diagonal part of  $N_s/N_{s+1}$ . Hence to compute  $\mathcal{I}$  it is sufficient to compute the connected components of  $\mathcal{G}$ .

To compute the graph  $\mathcal{G}$ , it is sufficient to give an algorithm to check whether  $T_i$  and  $T_j$  are linked. For this we compute  $N_s/N_{s+1}$  restricted to  $T_i \times T_j$ . Let  $\phi_{ij}$  denote the projection of  $L/M$  to  $T_i \times T_j$ . We give an FL algorithm (Algorithm 4) that computes a subset  $D_{i,j}$  of elements in  $N_s$  such that the projection from  $D_{i,j}$  to  $T_i \times T_j$  is  $\phi_{ij}(N_s)$ . Since  $\phi_{ij}(N_s)$  is of order bounded by a constant that depends only on  $c$ , one can easily determine whether it is  $T_i \times T_j$  or  $\text{Diag}(T_i \times T_j)$  (by checking the order for example).

Initialise  $D_{i,j}$  to be the set of  $x^*$  one for each  $x \in \phi_{ij}(S)$ .

**repeat**

Let  $S'$  be the set  $g^{-1}sg$  for each  $g \in A$  and  $s \in D_{i,j}$ .

Add to  $D_{i,j}$  all elements  $s$  in  $S'$  such that no two elements of  $D_{i,j}$  have the same image under  $\phi_{ij}$ .

**until**  $D_{i,j}$  is not modified;

**return**  $D_{i,j}$

**Algorithm 4:** Computing  $N_s/N_{s+1}$  restricted to  $T_i \times T_j$

Using Algorithm 4 we compute the edges of the graph  $\mathcal{G}$ . The graph  $\mathcal{G}$  is a disconnected set of cliques one for each diagonal component. In FL we compute its connected components. Let  $\mathcal{C}_1, \dots, \mathcal{C}_s$  be the connected components of  $\mathcal{G}$ . Then the vertices of  $\mathcal{C}_k$  gives us  $I_k$ . Thus in FL we compute the partition  $\mathcal{I}$ .

**Computing  $B_k$ :** To compute the set  $B_k$  the main algorithmic step is the computation of elements  $g_k \in N_s$ ,  $1 \leq k \leq s$  such that  $\phi_{i_k}(g_k) \neq 1$  and

$\phi_{i_j}(g_k) = 1$  for all  $j$  not equal to  $k$ . Given two  $i$  and  $j$  such that  $T_i$  and  $T_j$  are not linked, using Algorithm 4 one can compute an element  $g$  that is nontrivial on  $T_i$  and trivial on  $T_j$ . However we want elements  $g_k$  that is nontrivial on  $T_{i_k}$  and trivial on all other  $T_{i_j}$  simultaneously. We make use of the following proposition.

**Proposition 5.19.** *Let  $x$  and  $y$  be permutations in  $N_s$ . Let  $X$  denote the indices  $i$  such that  $\phi_i(x)$  is trivial. Similarly let  $Y$  be the set of all  $j$  such that  $\phi_j(y)$  is trivial. Then the commutator  $[x, y]$  has the property that  $\phi_j([x, y]) = 1$  for all  $j \in X \cup Y$ .*

*Proof.* For all  $i \in X$  since  $\phi_i(x) = 1$  we have  $\phi_i([x, y]) = \phi_i(x^{-1}y^{-1}xy) = \phi_i(y^{-1}y) = 1$ . Similarly for all  $j \in Y$ ,  $\phi_j([x, y]) = 1$ . Therefore for all  $k \in X \cup Y$   $\phi_k([x, y]) = 1$ .  $\square$

We use Proposition 5.19 to compute the required permutations  $g_k$ . For this purpose we need *iterated commutators*. Let  $[h_1, \dots, h_k]$  be defined as

$$\begin{aligned} [h_1, h_2] &= h_1^{-1}h_2^{-1}h_1h_2, \\ [h_1, \dots, h_i, h_{i+1}] &= [[h_1, \dots, h_i], h_{i+1}]. \end{aligned}$$

To compute  $g_k$  we compute a sequence of elements  $h_1, \dots, h_s$  satisfying the following properties.

1.  $\phi_{i_k}(h_1) \neq 1$ ,
2.  $\phi_{i_k}([h_1, \dots, h_j]) \neq 1$  for all  $j$  and
3.  $\phi_{i_j}(h_j) = 1$  for all  $1 \leq j \leq s$  and  $j \neq k$ .

It follows from Proposition 5.19 that given  $h_1, \dots, h_s$  with the above mentioned properties,  $g_k = [h_1, \dots, h_s]$  has the required properties:  $\phi_{i_k}(g_k)$  is nontrivial and  $\phi_{i_j}(g_k) = 1$  for  $1 \leq j \leq s$  and  $j \neq k$ . We give the logspace algorithm (Algorithm 5) to find such a sequence  $h_1, \dots, h_s$ .

In Algorithm 5 the step 1 is possible only because  $T_{i_k}$  is nonabelian and simple. The simplicity of  $T_{i_k}$  guarantees that its centre is trivial and hence for any nontrivial element  $g$  of  $T_{i_k}$  there is an  $h \in T_{i_k}$  such that  $g$  and  $h$  do not commute. The loop invariant is that  $g$ 's value is  $\phi_{i_k}([h_1, \dots, h_j]) \neq 1$ . Step 1 ensures that (1)  $\phi_{i_k}([h_1, \dots, h_j]) \neq 1$  and (2)  $\phi_{i_j}(h_j) = 1$ . Therefore Algorithm 5 indeed computes a sequence  $h_1, \dots, h_s$  with the desired properties.

Having got the sequence  $h_1, \dots, h_s$  we show that the iterated commutator  $[h_1, \dots, h_s]$  can be computed in logspace. It is sufficient to compute the

Let  $h_i$  be any permutation such that  $\phi_{i_k}(h_1) \neq 1$ . Such an element has to exist in the set  $S$  itself.

```

 $g \leftarrow \phi_{i_k}(h_1)$ 
for  $j = 1$  to  $s$  and  $j \neq k$  do
1 |   Using Algorithm 4 find an  $h_j$  such that  $\phi_{i_k}(h_j)$  does not commute
   |   with  $g$  and  $\phi_{i_j}(h_j) = 1$ .
   |    $h \leftarrow \phi_{i_k}(h_j)$ 
   |    $g \leftarrow [g, h]$ 
   |   output  $h_j$ 
end

```

**Algorithm 5:** Computing  $h_i$ 's

action of  $[h_1, \dots, h_s]$  separately for each  $G$ -orbit. The iterated commutator  $[h_1, \dots, h_s]$  is a formula over the  $h_i$ 's, and since each  $G$ -orbit is of bounded size, the action of  $[h_1, \dots, h_s]$  restricted to a  $G$ -orbit  $\Omega_i$  can be computed by a bounded width branching program. Hence the iterated commutator can be computed in FL (in fact even in NC<sup>1</sup>). Thus we have the following proposition.

**Proposition 5.20.** *Let  $G$  be a permutation group with bounded-size orbits. Given  $h_1, \dots, h_n \in G$ , the iterated commutator  $[h_1, \dots, h_n]$  can be computed in deterministic logspace.*

Using Algorithm 5 and Proposition 5.20, for all  $1 \leq k \leq s$ , we compute in FL a permutation  $g_k \in N_s$  such that  $\phi_{i_k}(g_k) \neq 1$  and for all  $1 \leq j \leq s$ ,  $j \neq k$   $\phi_{i_j}(g_k) = 1$ .

Finally from the permutations  $g_k$ , we now describe how the set  $B_k$  can be computed. Since  $T_{i_k}$  is simple,  $T_{i_k} = \phi_{i_k}(\text{NCL}_G(g_k))$ . We compute a set  $B_k$  of distinct inverse images of  $\phi_{i_k}(\text{NCL}_G(g_k))$ ,  $1 \leq k \leq l$ . Start with  $B_k = \{g_k\}$ . The algorithm consists of  $\#T$  stages in which we update  $B_k$ . At every stage update  $B_k$  by adding, for every element  $g$  in the generating set for  $G$  and  $x \in B_k$ , the elements  $y = g^{-1}xg$  to  $B_k$  if  $\phi_{i_k}(y) \notin \phi_{i_k}(B_k)$ . We repeat this process till  $\phi_{i_k}(B_k)$  generates  $T_{i_k}$ . Since  $\#T_{i_k} = \#T \leq c!$  we require at most  $c!$  stages each of which is in FL. Thus the sets  $B_k$ ,  $1 \leq k \leq s$  can be computed in FL.

Having computed the sets  $B_k$ , we compute the strong generator set  $B = \cup_{k=1}^s B_k$  of  $N_s$  rel  $N_{s+1}$ .

### Computing sifts:

Finally we explain how to compute  $\text{Sift}(x)$  for any  $x \in N_s$  with respect

to the computed strong generator set  $B = \cup_{k=1}^s B_k$  of  $N_s \text{ rel } N_{s+1}$ . Given  $x \in N_s$  in FL we compute for each,  $1 \leq k \leq s$ , a permutation  $y_k \in B_k$  such that  $\phi_{i_k}(y_k) = (\phi_{i_k}(x))^{-1}$ . The sift of  $x$  is given by  $\text{Sift}(x) = x \prod_{k=1}^s y_k$ . This completes the nonabelian case of our induction step.

### Computing the strong generating set: abelian case

We are given a set  $S \subset N_s$  such that  $\text{NCL}_G(S) = N_s$ . Our task is to compute the strong generator set of  $N_s \text{ rel } N_{s+1}$ . Since  $N_s/N_{s+1}$  is semisimple and abelian, it is  $\mathbb{F}_p$ -semisimple for some prime  $p \leq c$ . First we describe  $\text{FL}^{\text{Mod}_p L}$  algorithms for some basic linear algebraic problems over  $\mathbb{F}_p$  that follows from the results of Buntrock *et al* [20]. These will be used as subroutines in our  $\text{FL}^{\text{Mod}_p L}$  algorithm for computing the strong generator set of  $N_s \text{ rel } N_{s+1}$ .

**Proposition 5.21.** *For a prime  $p$  consider the vector space  $V = \mathbb{F}_p^r$  then*

1. *Let  $\mathcal{B} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  be a subset of  $V$ . Given  $\mathbf{v} \in V$ , in  $\text{Mod}_p L$  we can check whether  $\mathbf{v}$  is contained in the subspace  $U$  of  $V$  spanned by  $\mathcal{B}$ . Furthermore, if  $\mathbf{v} \in U$  then in  $\text{FL}^{\text{Mod}_p L}$  we can compute  $a_1, \dots, a_n \in \mathbb{F}_p$  such that  $\mathbf{v} = \sum_{i=1}^n a_i \mathbf{v}_i$ .*
2. *Let  $\mathcal{B} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  be a subset of  $V$  not necessarily linearly independent and let  $U$  be the subspace of  $V$  spanned by  $\mathcal{B}$ . Then in  $\text{FL}^{\text{Mod}_p L}$  we can compute a subset  $\mathcal{B}' \subseteq \mathcal{B}$  such that  $\mathcal{B}'$  is a basis for  $U$ .*

*Proof.* Let  $\mathbf{e}_1, \dots, \mathbf{e}_m$  denote the standard basis for  $V = \mathbb{F}_p^r$  and let  $\mathbf{v}_i = \sum_{j=1}^m v_{i,j} \mathbf{e}_j$  for  $1 \leq i \leq n$ . Let  $\mathbf{v} = \sum_{j=1}^m v_j \mathbf{e}_j$ . Let  $A$  be the matrix  $(v_{i,j})$ ,  $1 \leq i \leq n$  and  $1 \leq j \leq m$ . Let  $\mathbf{b}$  the column vector  $(v_1, \dots, v_m)^T$ . Then the vector  $\mathbf{v}$  is in the span of  $\mathcal{B}$  if and only if the system of linear equation  $A\mathbf{x} = \mathbf{b}$  has a solution. Furthermore if  $x_i = a_i$ ,  $1 \leq i \leq n$  is a solution to  $A\mathbf{x} = \mathbf{b}$  then  $\mathbf{v} = \sum_{i=1}^n a_i \mathbf{v}_i$ . Part 1 then follows from Theorem 2.5.

To prove part 2 consider the  $\text{FL}^{\text{Mod}_p L}$  algorithm that cycles over all  $1 \leq i \leq n$  and outputs  $\mathbf{v}_i$  if it is not in the span of the set  $\{\mathbf{v}_1, \dots, \mathbf{v}_{i-1}\}$ . Clearly the output  $\mathcal{B}'$  is a basis of the vector space spanned by  $\mathcal{B}$ .  $\square$

We fix some notations: Recall that  $R_{i,s}/R_{i,s+1}$  is isomorphic to vector space over  $\mathbb{F}_p$  which we denote by  $V_i$ . Let  $V$  be the direct sum  $\oplus_{i=1}^m V_i$ . Then  $R_s/R_{s+1}$  is isomorphic to  $V$ . For a permutation  $x \in R_s$  let  $\mathbf{v}_x$  denote the image of  $xR_{s+1}$  under the above mentioned isomorphism. If  $x$  and  $y$  are permutations in  $L$  then for integers  $a$  and  $b$  it follows that  $\mathbf{v}_{x^a y^b} = \tilde{a} \mathbf{v}_x + \tilde{b} \mathbf{v}_y$  where  $\tilde{a}$  and  $\tilde{b}$  are the elements  $a \pmod{p}$  and  $b \pmod{p}$  of  $\mathbb{F}_p$  respectively. Furthermore the vector space structure of  $R_s/R_{s+1}$  is obtainable effectively

in logspace, i.e. for an element  $x \in R_s$  one can compute the image  $\mathbf{v}_x$  of the coset  $xR_{s+1}$  in  $V$  in FL. This is because each of the groups  $R_{i,s}$  are constant sized. Since  $\frac{N_s}{N_{s+1}} \hookrightarrow R_s/R_{s+1}$  it is isomorphic to a subspace of  $V$  which denote by  $W$ . We are given a subset  $S \subseteq N_s$  such that  $\text{NCL}_G(S) = N_s$ .

The group  $\text{NCL}_G(S)$  is the group generated by the set  $\{g^{-1}sg | s \in S, g \in G\}$  and hence the conjugation action can be seen as a linear action of  $G$  on  $V$  as we now explain: For each element  $g \in G$ ,  $g$  maps  $\mathbf{v}_h$ ,  $h \in R_s$ , to the vector  $\mathbf{v}_{h^*}$  where  $h^* = g^{-1}hg$ . Since both  $R_s$  and  $R_{s+1}$  are normalised by  $G$ , each  $g \in G$  is an invertible linear transformation from  $V$  to  $V$ .

First  $S \subseteq N_s$  and  $N_s$  is a normal subgroup of  $G$ . Therefore  $\text{NCL}_H(S)$  is a subgroup of  $N_s$  that is closed under conjugation by elements of  $H$ . Thus we have the following observation of Luks and McKenzie [50] about the normal closure  $\text{NCL}_H(S)$ .

**Proposition 5.22** (Luks and McKenzie). *Let  $H \leq G$  and let  $W$  be the subspace  $\{\mathbf{v}_x | x \in \text{NCL}_H(S)\}$ . Then  $W$  is the smallest subspace of  $V$  containing  $\{\mathbf{v}_s | s \in S\}$  and closed under the action of elements of  $H$*

We compute the generator set of  $\text{NCL}_{N_j}(S)$  rel  $N_{s+1}$  inductively starting with  $j = s$  down to  $j = 0$  using the SGS of  $N_j$  rel  $N_s$  that is already computed. Let  $U_j$  denote the subspace of  $V$  associated to  $\text{NCL}_{N_j}(S)/N_{s+1}$  then it follows from 5.22 that  $U_j$  is the closure of  $\{\mathbf{v}_s | s \in S\}$  under  $N_j$ . We compute a basis for  $U_j$ .

To begin with since  $N_s/N_{s+1}$  is commutative. It follows that  $S \cup N_{s+1}$  is a generating set for  $\text{NCL}_{N_s}(S)$  and hence  $U_s$  is spanned by  $\{\mathbf{v}_s | s \in S\}$ . Assume that we have already computed a basis for  $U_{j+1}$ . Our task is to compute a basis for  $U_j$  using the basis for  $U_{j+1}$  and the strong generator set  $C$  of  $N_j$  rel  $N_{j+1}$ . The vector space  $U_j$  is the span of  $gU_{j+1}$  where  $g$  ranges over the distinct coset representative of  $N_{j+1}$  in  $N_j$ .

**Proposition 5.23.** *Given a basis for  $U_{j+1}$  we can compute a basis for  $U_j$  in  $\text{Mod}_p\text{L}$ .*

*Proof.* Recall that  $N_j/N_{j+1}$  is  $T$ -semisimple for some simple group  $T$ . Since  $U_{j+1}$  is stabilised by  $N_{j+1}$  we can assume that  $N_j/N_{j+1}$  is acting on  $U_{j+1}$ . Depending on whether  $T_j$  is abelian or not we have two cases.

**T is non-abelian** We prove that in this case it is sufficient to find the closure of  $U_{j+1}$  under all monomials  $M$  over  $C$  which are of degree bounded by a constant  $c'$  that depends only on  $c$ . Recall that  $R_j/R_{j+1} = T_1 \times \dots \times T_n$  for some integer  $n$  and there is a partition  $\mathcal{I} = \{I_1 \dots I_r\}$  such that

$N_j/N_{j+1}$  is product of diagonal groups  $\text{Diag} \left( \prod_{i \in I_k} T_i \right)$ ,  $1 \leq i \leq r$ . Each of the diagonal component  $\text{Diag} \left( \prod_{i \in I_k} T_i \right)$  is isomorphic to  $T$  and the strong generator set  $C$  is the union  $C = \cup C_k$  where  $C_k$  consists of one element  $g \in N_j$  for each  $gN_{j+1} \in \text{Diag} \left( \prod_{i \in I_k} T_i \right)$ .

Consider any element  $g \in C$ . We say that  $g$  is trivial on  $\Omega_i$  if  $g|_{\Omega_i} \in R_{i,j+1}$ . For each  $g \in C$  there exists  $h_g \in N_{j+1}$  such that  $g|_{\Omega_i} = h_g|_{\Omega_i}$  for all  $\Omega_i$  for which  $g$  is trivial. Consider the elements  $\mu_g = h_g - g$ ,  $g \in C$ . Then  $U_j$  is the space spanned by  $MU_{j+1}$  where  $M$  ranges over all (non-commutative) monomial in  $\{\mu_g : g \in C\}$ . If  $g \in C_i$  and  $h \in C_j$ ,  $i \neq j$  then since  $gh = hg$  for some  $x \in N_{j+1}$  we can assume that  $P(g)$  and  $Q(h)$  commutes for any two polynomials  $P(X)$  and  $Q(X)$ . Any monomial  $M$  in  $\mu_g$ 's can therefore be assumed to be in the form  $\mu_1 \dots \mu_n$  where  $\mu_i$  is either 1 or  $h_g - g$  for some  $g \in C_i$ .

For any  $i$  if  $g$  is trivial on  $\Omega_i$  then  $\mu_g V_i = 0$ . Since each orbit  $\Omega_i$  is of cardinality at most  $c$  there exists a constant  $c'$  that depends only on  $c$  such that for any orbit  $\Omega_i$  there are at most  $c'$  distinct  $\mu_g$ 's that are non-zero on  $V_i$ . Consider a monomial  $M = \mu_1 \dots \mu_n$  of degree  $n > c'$ . For any  $V_i$  there is a  $\mu_k$  such that  $\mu_k V_i = 0$ . Therefore since  $V$  is the direct sum  $V_1 \oplus \dots \oplus V_m$ ,  $MV = 0$ . As a consequence to obtain  $U_j$  it is sufficient to take the closure of  $U_{j+1}$  with respect to monomials in  $\{\mu_g | g \in C\}$  of total degree bounded by  $c'$ . In FL we can enumerate all monomials over  $C$  of degree bounded by  $c'$ . Hence  $U_j$  is the obtained by taking the span of  $MU_{j+1}$  where  $M$  is a monomial in  $C$  of degree at most  $c$ . A basis of for  $U_j$  can then be computed in  $\text{FL}^{\text{Mod}_p L}$  using Proposition 5.21.

**T is abelian** The vector space  $U_{j+1}$  is closed action of  $N_{j+1}$ . Therefore as far as computing the closure of  $U_{j+1}$  is concerned we assume that the group algebra of the quotient group  $N_j/N_{j+1}$  is acting on  $V$ . In this case the group algebra of  $N_j/N_{j+1}$  is abelian. Therefore elements  $g$  and  $h$  can be thought of as commuting linear transformations over  $V$ . Also there is a prime  $q < c$  such that  $g^q - 1 = 0$ . In  $\text{FL}^{\text{Mod}_p L}$  we can find a set  $\mathcal{T}$  of elements in the group algebra  $N_j/N_{j+1}$  such that the  $U_j$  is the span of  $\{\tau U_{j+1} : \tau \in \mathcal{T}\}$ .  $\square$

We now give the  $\text{FL}^{\text{Mod}_p L}$  algorithm for computing the strong generator set of  $N_s$  rel  $N_{s+1}$  problem. Let  $W$  denote the subspace of  $\{\mathbf{v}_x | x \in N_s\}$ . It follows from Proposition 5.23 that a basis  $\mathcal{B}$  for the space  $W$ . We can keep track of the entire permutations: Whenever we add the vector  $g\mathbf{v}_x$  into  $\mathcal{B}$  we add the corresponding permutation  $g^{-1}xg$  into  $B$ . We thus have a subset  $B$  of  $N_s$  such that  $\{\mathbf{v}_x | x \in B\}$  spans  $W$ . Let  $B = \{x_1, \dots, x_n\}$  then,

for  $1 \leq i \leq n$ , define the set  $C_i = \{x_i^a : 1 \leq a \leq p-1\}$ . The set  $\cup_{i=1}^n C_i$  is the strong generator set of  $N_s$  rel  $N_{s+1}$ . Clearly  $C$  can be computed in  $\text{FL}^{\text{Mod}_p L}$ .

Finally, we describe how to compute  $\text{Sift}(x)$  for any  $x \in N_s$  with respect to the above mentioned strong generator set. In logspace compute the vector  $\mathbf{v}_x \in V$  corresponding to the permutation  $x$ . Using Proposition 5.21 compute  $a_i \in \mathbb{F}_p$  such that  $\mathbf{v}_x = \sum a_i \mathbf{v}_{x_i}$ . The sift of  $x$  is given by  $\text{Sift}(x) = x \prod_{i=1}^r x_i^{-a_i}$ . This completes the abelian case of our induction step.

We have thus shown that the strong generator set of  $G$  rel  $N_{s+1}$  can be computed inductively starting from  $s = 0$ . Since the locally residual series  $G = N_0 \supseteq \dots \supseteq N_l$  is of length  $l$  bounded by a constant in  $c$  we have the following theorem.

**Theorem 5.24.** *Let  $G$  be a permutation group with orbits of size bounded by a constant  $c$ . Given a generator set  $A$  for  $G$ , we can compute the strong generator set for  $G$  with respect to the locally residual series in the  $\text{Mod}_k L$ -hierarchy. The constant  $k$  is the product of all primes less than  $c$  and the level of the hierarchy depends only on  $c$ .*

## 5.5 The target reduction procedure

Our goal is to show that  $\text{PWS}_c$  is in the  $\text{Mod}_k L$ -hierarchy. The heart of the algorithm is the target reduction procedure: Given an instance  $(G, \Omega, \Delta)$  of  $\text{PWS}_c$ , we compute a subgroup  $G'$  of  $G$  containing  $G(\Delta)$  such that for each  $G$ -orbit  $\Omega'$  that contains a point of  $\Delta$ ,  $G'|_{\Omega'}$  is a proper subgroup of  $G|_{\Omega'}$ . In this section we show that target reduction can be performed in the  $\text{Mod}_k L$ -hierarchy.

Let  $\Omega_1, \dots, \Omega_m$  be the set of  $G$ -orbits. We fix some terminologies and conventions local to this section. Points in  $\Delta$  will be called *target points*. *Target orbits* are  $G$ -orbits that contain target points, i.e. orbits  $\Omega_i$  such that  $\Omega_i \cap \Delta \neq \emptyset$ .

Firstly, if  $\Sigma \subseteq \Delta$  then  $G(\Sigma) \geq G(\Delta)$ . Let  $\Sigma$  be the subset of  $\Delta$  that contains exactly one target point from each target orbit. Any target orbit will continue to be a target orbit even if we replace  $\Delta$  by the subset  $\Sigma$ , i.e. if  $G'$  be the group obtained by performing target reduction on  $(G, \Omega, \Sigma)$  then  $G' \geq G(\Delta)$  and  $G'|_{\Omega'}$  is a proper subgroup of  $G|_{\Omega'}$  for all target orbit  $\Omega'$ . Therefore as far as target reduction is concerned, we can assume that the instance  $(G, \Omega, \Delta)$  is such that each  $G$ -orbit contains at most one target point.

We make an additional assumption that  $G$  acts primitively on each target orbit which we justify now. Consider a structure forest  $\mathcal{F} = \{\mathcal{T}_1, \dots, \mathcal{T}_m\}$  of  $G$  where  $\mathcal{T}_i$  is the structure tree of the transitive action of  $G$  on  $\Omega_i$ . Let  $\Omega^*$  denote the vertices of  $\mathcal{F}$ . We identify the set  $\Omega$  with the set of leaf nodes of  $\mathcal{F}$ . Recall from Section 3.4 that  $G$ 's action on  $\Omega$  can be extended to  $\Omega^*$  such that given the action of an element  $g \in G$  on  $\Omega^*$ , we can recover its action on  $\Omega$ . Furthermore, all the  $G$ -orbits of  $\Omega^*$  are of size bounded by  $c$ . In FL we can compute the structure forest  $\mathcal{F}$  of  $G$  by separately computing the structure tree  $\mathcal{T}_i$  for each  $1 \leq i \leq m$ . Furthermore for a given  $g$  in  $G$ , in FL we can compute the action of  $g$  on the  $\Omega^*$ .

Let  $\Omega_i^* \subseteq \Omega^*$ ,  $1 \leq i \leq m$ , denote the children of the root of  $\mathcal{T}_i$ . Then for every  $1 \leq i \leq m$ , the set  $\Omega_i^*$  is an orbit and  $G$  acts primitively on it (Theorem 3.12). Let  $\Delta^*$  be the ancestors of elements of  $\Delta$  in  $\cup_{i=1}^m \Omega_i^*$ .

**Proposition 5.25.** *The group  $G(\Delta^*)$  contains  $G(\Delta)$  and for any subgroup  $H$  of  $G$ , if  $H|_{\Omega_i^*} < G|_{\Omega_i^*}$  then  $H|_{\Omega_i} < G|_{\Omega_i}$ .*

*Proof.* Consider any  $\delta^* \in \Delta^*$ . There is a  $\delta \in \Delta$  such that  $\delta^*$  is the ancestor of  $\delta$  in the structure forest of  $G$ . Let  $\Sigma$  be the leaves of the structure tree rooted at  $\delta^*$  then  $\Sigma$  is a  $G$ -block that contains  $\delta$  (Section 3.4). Thus for a  $g \in G$  if  $\delta^g = \delta$  then  $\Sigma^g = \Sigma$  and hence  $\delta^{*g} = \delta^*$ . This proves that  $G(\Delta^*) \geq G(\Delta)$ .

Consider any subgroup  $H$  of  $G$ . Recall that the action of  $G$  on the structure tree  $\mathcal{T}_i$  depends only on the action of  $G|_{\Omega_i}$  on  $\Omega_i$ . Hence if  $H|_{\Omega_i} = G|_{\Omega_i}$  then  $H|_{\Omega_i^*} = G|_{\Omega_i^*}$ .  $\square$

Proposition 5.25 proves that target reduction for the instance  $(G, \Omega, \Delta)$  can be achieved by performing target reduction on  $(G, \Omega^*, \Delta^*)$ . Summarising the above discussions, for target reduction we assume that the given instance  $(G, \Omega, \Delta)$  has the following properties.

1. All  $G$ -orbits are of size bounded by a constant  $c$ .
2.  $G$  acts primitively on each target orbit.
3. Each target orbit contains a unique target point.

In this section we use the following notation:  $X$  denotes the set of all  $i$  such that  $\Omega_i$  is a target orbit. For each index  $i \in X$ ,  $\delta_i$  denotes the unique target point in  $\Omega_i$ . Identifying the indices of  $X$ , the corresponding target orbits and the target points leads to no confusion. Hence for a subset of  $Z$  of  $X$ , by target orbits of  $Z$  we mean the collection  $\{\Omega_i | i \in Z\}$ . Similarly by target points of  $Z$  we mean the set  $\{\delta_i | i \in Z\}$ .

### Overview of the target reduction step

First we use Theorem 5.24 to compute the strong generator set of  $G$  with respect to the locally residual series  $G = N_0 \supseteq \dots \supseteq N_l = \{1\}$ . In fact in the  $\text{Mod}_k\text{L}$ -hierarchy we obtain the following.

1. For each  $i$  a residual series  $G_i = R_{i,0} \supseteq \dots \supseteq R_{i,l} = 1$  such that,  $R_{i,s+1} = \text{Res}_{T_s}(R_{i,s})$ .
2. The product group  $R_s = \prod_i R_{i,s}$  and
3. The groups  $N_s = G \cap R_s$ .

Consider any target orbit  $\Omega_i$ . Recall that  $N_s|_{\Omega_i} = R_{i,s}$  (Proposition 5.17) and hence  $G_i = N_0|_{\Omega_i} \supseteq \dots \supseteq N_l|_{\Omega_i} = \{1\}$  is a residual series for  $G_i$ . Since  $G_i$  acts primitively on  $\Omega_i$ , the last nontrivial group in this series is  $\text{Soc}(G_i)$  (Lemma 5.15). Let  $X_s$  denote the set of all  $i$  such that  $N_s|_{\Omega_i} = \text{Soc}(G_i)$  and  $N_{s+1}|_{\Omega_i} = \{1\}$ . We have  $\cup X_s = X$ .

The target reduction is done inductively in  $l$  stages where in the  $s$ th stage we handle the target orbits in  $X_s$ . Inductively we compute the generator sets of the sequence of groups  $G = H_0 \geq \dots \geq H_{l+1}$  such that for all  $s$ ,  $H_s \geq G(\Delta)$  and  $H_s|_{\Omega_i}$  is a proper subgroup of  $G|_{\Omega_i}$  for each  $i$  in  $\cup_{j=0}^{s-1} X_s$ . In fact the group  $H_s$  that we compute in the  $s$ th stage will contain  $N_s$ . Since  $X = \cup_{j=0}^l X_s$ , the group  $H_{l+1}$  is the required group  $G'$ .

To begin with  $H_0 = G$ . Inductively assume that we have computed a generator set  $A_s$  of  $H_s$ . To compute  $H_{s+1}$  we first identify a subset  $Y_s \subseteq X_s$  of critical indices. A subset  $Y$  of  $X$  is said to be *critical* if it has the following properties:

1. Let  $N$  be the subgroup of  $N_s$  that fixes all the target points in  $Y$  then  $N|_{\Omega_i} < N_s|_{\Omega_i}$  for all  $i \in X_s$ .
2. For any  $x \in G$  there is a  $y \in N_s$  such that  $x^* = xy$  fixes all the points of  $Y$ .

**Proposition 5.26.** *Let  $H$  be the subgroup of  $H_s$  that fixes all the target points in a critical subset  $Y$  of  $X_s$  then  $H_s \geq H \geq G(\Delta)$  and for all  $i \in X_s$   $H|_{\Omega_i} < G_i$ .*

*Proof.* Let  $\Delta'$  be the subset of  $\Delta$  containing all the target points of  $Y$ . By induction hypothesis  $H_s \geq G(\Delta)$  and hence  $H = H_s(\Delta') \geq G(\Delta)$ . The group  $H_s \cap R_s = N_s$  we have  $H \cap R_s = N$ . Since  $N|_{\Omega_i} < R_{i,s}$  for all  $i \in X_s$ ,  $H \cap R_s|_{\Omega_i} < R_{i,s}$ . Therefore by Proposition 5.17 we have  $H|_{\Omega_i} < G_i$  for all  $i \in X_s$ .  $\square$

Recall that our goal is to compute a subgroup  $H_{s+1}$  of  $H_s$  that contains  $G(\Delta)$  and is strictly smaller than  $G_i$  on  $\Omega_i$  for each  $i \in X_s$ . By Proposition 5.26 it is sufficient to choose  $H_{s+1}$  to be the subgroup of  $H_s$  that fixes all the target points in  $Y$  for some critical subset  $Y$  of  $X_s$ . In the  $s$ th stage of the algorithm we identify a critical subset  $Y_s$  which is *effective*, i.e. given any  $x \in G$ , in  $\text{FL}^{\text{Mod}_k L}$  we can compute a  $y \in N_s$  such that  $x^* = xy$  fixes all the points in  $Y_s$ . The subgroup  $H_{s+1}$  is the subgroup of  $H_s$  that fixes all the target points of  $Y_s$ .

We now show how a generator set for  $H_{s+1}$  can be computed. Let  $A_s$  be the a generator set for  $H_s$ . Since  $Y_s$  is critical for each  $x \in A_s$  there is a  $y \in N_s$  such that  $x^* = xy$  fixes all the target points in  $Y_s$ . Let  $A_s^*$  denote the set  $\{x^* | x \in A_s\}$ .

**Proposition 5.27.** *Let  $N$  be the subgroup of  $N_s$  that fixes all the target points of  $Y_s$  and let  $B$  be a generator set for  $N$ . Then  $A_s^* \cup B$  generates the subgroup  $H_{s+1}$ .*

*Proof.* First, we claim that  $A_s^* N_s$  generates  $H_s$ . Consider any  $g \in H_s$ . Since  $A_s$  generates  $H_s$ , for some integer  $t \geq 0$  there exists  $t$  elements  $x_1, \dots, x_t$  in  $A_s$  such that  $g$  is the product  $\prod_{i=1}^t x_i$ . For any  $x \in A_s$  there is an element  $y \in N_s$  such that  $x^* = xy$  is contained in  $A_s^*$ . Therefore we have for  $1 \leq i \leq t$  elements  $x_i^* \in A_s^*$  and  $y_i \in N_s$  such that  $g = x_1^* y_1 \dots x_t^* y_t$ . This proves our claim.

The group  $N_s$  is a normal subgroup of  $G$  and hence is also a normal subgroup of  $H_s$ . Therefore every element  $g \in H_s$  can be written as  $g = g_1 g_2$  where  $g_1$  is contained in the group generated by  $A_s^*$  and  $g_2 \in N_s$ . Furthermore, since every element of  $A_s^*$  fixes all the target points of  $Y_s$ , so does  $g_1$ . As a consequence any element of  $H_{s+1}$ , the subgroup of  $H_s$  that fixes the target points of  $Y_s$ , is of the form  $uv$  where  $u$  is in the group generated by  $A_s^*$  and  $v \in N$ . Hence  $A_s^* N$  generates the group  $H_{s+1}$  and if  $B$  is a generator set of  $N$ ,  $A_s^* \cup B$  generates  $H_{s+1}$ .  $\square$

To complete the inductive procedure for target reduction it is thus sufficient to perform the following subtasks.

1. Compute a critical subset  $Y_s \subseteq X_s$ .
2. Given  $x \in G$  compute a  $y \in N_s$  such that  $xy$  fixes each of the target points of  $Y_s$ .
3. Compute a generator set of the subgroup  $N$  of  $N_s$  that fixes each of the target points of  $Y_s$ .

Depending on whether  $N_s/N_{s+1}$  is abelian or not we have two case. When  $N_s/N_{s+1}$  is abelian then  $N_s/N_{s+1}$  is  $\mathbb{F}_p$ -semisimple for some prime  $p \leq c$ . In this case we show that the steps 1, 2 and 3 can be done in  $\text{FL}^{\text{Mod}_p L}$ . On the other hand when  $N_s/N_{s+1}$  is nonabelian then the steps 1, 2 and 3 can be done in FL. We explain these two case in the next two subsections.

### 5.5.1 Computing the critical orbits: abelian case

In this case the quotient group  $N_s/N_{s+1}$  is  $\mathbb{F}_p$ -semisimple for some prime  $p \leq c$ . Let  $\Omega_1, \dots, \Omega_m$  be the  $G$ -orbits and let  $G_i = G|_{\Omega_i}$ . Then by the O’Nan-Scott theorem  $\text{Soc}(G_i)$  is regular on  $\Omega_i$ , i.e.  $\text{Soc}(G_i)$  is transitive on  $\Omega_i$  and for any  $\delta \in \Omega_i$ , the subgroup of  $\text{Soc}(G_i)$  that fixes  $\delta$  is the trivial group. As a consequence we have the following property.

**Proposition 5.28.** *Let  $Y$  be any subset of  $X_s$  and let  $K$  be the subgroup of  $N_s$  that fixes the target points of  $Y$ . Then we have.*

1.  $K \trianglelefteq G$ .
2. For any  $i$  in  $X_s$  either  $K|_{\Omega_i}$  is trivial or is  $\text{Soc}(G_i)$ .
3. For any  $i \in X_s$  such that  $K|_{\Omega_i}$  is nontrivial and for any two elements  $\delta$  and  $\delta'$  of  $\Omega_i$ , there is an element  $h$  of  $K$  such that  $\delta^h = \delta'$ .

*Proof.* The group  $K$  is a subgroup of  $N_s$  and hence for all  $i \in X_s$ ,  $K|_{\Omega_i}$  is a subgroup of  $\text{Soc}(G_i)$ . Moreover  $K$  fixes the target point  $\delta_i$  of  $\Omega_i$  for each  $i \in Y$ . Therefore the projection of  $K$  onto  $\Omega_i$  is the subgroup of  $\text{Soc}(G_i)$  that fixes  $\delta_i$ . However since  $\text{Soc}(G_i)$  is regular on  $\Omega_i$ ,  $K|_{\Omega_i}$  is trivial for all  $i \in Y$ . Thus  $K$  is the intersection of the groups  $N_s$  and  $\prod_{i \notin Y} R_{i,s}$ . The group  $G$  normalises the group  $\prod_{i \notin Y} R_{i,s}$ . Also  $N_s \trianglelefteq G$ . Hence their intersection  $K$  is a normal subgroup of  $G$ . This proves part 1.

Consider an  $i \in X_s$ . As argued before since  $K \leq N_s$ ,  $K|_{\Omega_i}$  is a subgroup of  $\text{Soc}(G_i)$ . Suppose that  $K|_{\Omega_i}$  is nontrivial. Then since  $K$  is a normal subgroup of  $G$  and since  $G|_{\Omega_i} = G_i$ , it follows that  $K|_{\Omega_i}$  is a normal subgroup of  $G_i$ . However by the O’Nan-Scott theorem  $\text{Soc}(G_i)$  is the unique minimal normal subgroup of  $G_i$ . Therefore  $K|_{\Omega_i} = \text{Soc}(G_i)$  which proves part 2.

Finally consider an  $i$  in  $X_s$  such that  $K|_{\Omega_i}$  is nontrivial. The group  $\text{Soc}(G_i)$  is regular on  $\Omega_i$  (O’Nan-Scott theorem). Hence for any two elements  $\delta$  and  $\delta'$  of  $\Omega_i$  we have an element  $g \in \text{Soc}(G_i)$  such that  $\delta^g = \delta'$ . Since  $K|_{\Omega_i} = \text{Soc}(G_i)$ , part 3 then follows as there is an element  $g^* \in K$  such that  $g^*|_{\Omega_i} = g$ .  $\square$

We fix the following notation for this subsection. Recall that  $N_s$  is the intersection of the product group  $R_s = \prod_{i=1}^m R_{i,s}$  and  $G$  (Subsection 5.4.1). The quotient groups  $R_{i,s}/R_{i,s+1}$  is  $\mathbb{F}_p$ -semisimple and hence is a vector space  $V_i$  over  $\mathbb{F}_p$ . Let  $V$  be the direct sum  $\bigoplus_{i=1}^m V_i$  then  $R_s/R_{s+1}$  is isomorphic to  $V$ . We identify the vector space  $V$  with the quotient group  $R_s/R_{s+1}$  under this isomorphism, i.e. for every element  $x \in R_s$  we associate isomorphic image  $\mathbf{v}_x$  of  $x$  in  $V$ . Clearly for any  $x$  and  $y$  in  $R_s/R_{s+1}$  the vector  $\mathbf{v}_{x+y}$  is  $\mathbf{v}_x + \mathbf{v}_y$  and for any integer  $a$   $\mathbf{v}_{x^a} = \tilde{a}\mathbf{v}_x$ , where  $\tilde{a} \in \mathbb{F}_p$  is the element  $a \pmod{p}$ .

**Proposition 5.29.** *Given any element  $x \in R_s$ , in FL we can compute the vector  $\mathbf{v}_x$ .*

*Proof.* Let  $\mathbf{w}_i$  denote the projection of  $\mathbf{v}_x$  onto the vector space  $V_i$ . Since the vector space  $V$  is the direct sum of the subspaces  $V_i$ ,  $\mathbf{v}_x = \sum_{i=1}^m \mathbf{w}_i$ . The order of the group  $R_{i,s}$  is at most  $c!$  as the size of the orbit  $\Omega_i$  is less than  $c$ . Hence in FL we can compute the projection  $\mathbf{w}_i$  and thus compute  $\mathbf{v}_x$ .  $\square$

The quotient group  $N_s/N_{s+1}$  is a subgroup of the  $R_s/R_{s+1}$  (more precisely  $N_s/N_{s+1} \hookrightarrow R_s/R_{s+1}$ ) and hence is a subspace  $U$  of  $V$ .

**Proposition 5.30.** *Given the strong generator set  $C$  of  $N_s$ , in  $\text{FL}^{\text{Mod}_p\text{L}}$  a subset  $B = \{x_1, \dots, x_r\}$  of  $C$  can be computed such that the vectors  $\mathbf{v}_{x_1}, \dots, \mathbf{v}_{x_r}$  forms a basis of  $U$ .*

*Proof.* The subset  $\mathcal{C} = \{\mathbf{v}_x | x \in C\}$  of  $V$  spans  $U$ . Using Proposition 5.21 in  $\text{FL}^{\text{Mod}_p\text{L}}$  compute a subset  $\mathcal{B}$  of  $\mathcal{C}$  that forms a basis of  $U$ . For each  $\mathbf{v} \in \mathcal{B}$  pick a permutation  $x \in C$ , say the lexicographically least such that  $\mathbf{v}_x = \mathbf{v}$  and form the subset  $B$  of  $C$ . Clearly  $B$  can be computed in  $\text{FL}^{\text{Mod}_p\text{L}}$ .  $\square$

Our goals are (1) to compute a critical subset  $Y_s$  of  $X_s$ , (2) compute a generator of the subgroup  $N$  of  $N_s$  that fixes all the points of  $Y_s$  and (3) give a  $x \in G$  compute a  $y \in N_s$  such that  $x^* = xy$  fixes all the target points of  $Y_s$ . By Proposition 5.28 it follows that the subgroup  $K$  of  $N_s$  that fixes some of the target points of  $X_s$  is such that for all  $i$  either  $K|_{\Omega_i} = 1$  or  $K|_{\Omega_i} = \text{Soc}(G_i)$ . The subset  $Y_s$  of  $X_s$  that we choose will be a minimal subset of target points such that the subgroup of  $N_s$  that fixes the points of  $Y_s$  will be trivial on all the target orbits of  $X_s$ . First we prove the following proposition that will be used to identify  $Y_s$  and later on to compute the set  $N$ .

**Proposition 5.31.** *Given any subset  $Y$  of  $X_s$  there is a  $\text{FL}^{\text{Mod}_p\text{L}}$  algorithm to compute the generator set of the subgroup  $K$  of  $N_s$  that fixes all the target points of  $Y$ .*

*Proof.* Since  $N_{s+1}$  is trivial on all the target orbits it follows that  $K$  contains  $N_{s+1}$ . Let  $W$  denote the vector space associated with the quotient group  $K/N_{s+1}$ . From Proposition 5.28 it follows that  $K$  is trivial on all the orbits of  $Y$ . Therefore  $K$  is the intersection of the groups  $N_s$  and  $\prod_{j \notin Y} R_{i,s}$ . Hence  $W$  is the subspace  $U \cap \bigoplus_{i \notin Y_s} V_i$ .

In  $\text{FL}^{\text{Mod}_p\text{L}}$  we first compute the set  $B = \{x_1, \dots, x_r\}$  such that  $\mathcal{B} = \{\mathbf{v}_{x_1}, \dots, \mathbf{v}_{x_r}\}$  is a basis for  $U$  (Proposition 5.30). Consider the projection of  $U$  on to the space  $\bigoplus_{i \in Y_s} V_i$ . Then  $W$  is the kernel of this projection. Let  $A$  be the matrix associated to this projection with respect to the basis  $\mathcal{B}$ . The subspace  $W$  consists of all vectors  $\sum a_i \mathbf{v}_{x_i}$  which are solutions of the linear equation  $A\mathbf{x} = 0$ . Using Theorem 2.5 we can compute a basis  $\mathbf{u}_1, \dots, \mathbf{u}_t$  for  $W$ . In fact the algorithm outputs the elements  $a_{ij} \in \mathbb{F}_p$  such that  $\mathbf{u}_i = \sum_{j=1}^r a_{ij} \mathbf{v}_{x_j}$ . Let  $g_i = \prod_{j=1}^r x_j^{a_{ij}}$  then clearly the set  $D = \{g_1, \dots, g_t\}$  generates the quotient group  $K/N_{s+1}$ . As part of the strong generator set of  $G$  we have already computed a strong generator set  $C'$  of  $N_{s+1}$ . The set  $C' \cup D$  gives a generator set for  $K$ .  $\square$

We now present the  $\text{FL}^{\text{Mod}_p\text{L}}$  algorithm (Algorithm 6) for computing  $Y_s$ .

```

 $Y \leftarrow \emptyset.$ 
foreach  $i \in X_s$  do
  | Let  $K_i$  be the subgroup of  $N_s$  that fixes target points of  $Y$ .
1 | if  $K$  is nontrivial on  $\Omega_i$  then  $Y \leftarrow Y \cup \{i\}$ 
end
Return the set  $Y_s = Y.$ 

```

**Algorithm 6:** Computing  $Y_s$

For the step 1 in  $\text{FL}^{\text{Mod}_p\text{L}}$  we first compute the generator set  $D_i$  of  $K_i$  (Proposition 5.31). The group  $K_i$  is trivial on  $i$  if and only if all the elements of  $D_i$  is trivial on  $\Omega_i$ . Thus step 1 can be performed by making a query to a  $\text{Mod}_p\text{L}$  oracle and hence Algorithm 6 is a  $\text{FL}^{\text{Mod}_p\text{L}}$  procedure. Having computed  $Y_s$  using Proposition 5.31 we compute in  $\text{FL}^{\text{Mod}_p\text{L}}$  the generator set of the subgroup  $N$  of  $N_s$  that fixes all the target points of  $Y_s$ . To complete the abelian case we show that for each  $x \in G$ ,  $x^*$  can be computed in  $\text{FL}^{\text{Mod}_k\text{L}}$ .

**Proposition 5.32.** *Given any  $x \in G$  there is a  $\text{FL}^{\text{Mod}_p\text{L}}$  algorithm to compute an element  $y$  of  $N_s$  such that  $xy$  fixes all the points of  $Y_s$ .*

*Proof.* Without loss of generality assume that  $Y_s = \{1, \dots, t\}$  for some integer  $t$ . Let  $\delta_i$  denote the target point in  $\Omega_i$ . Let  $K_0 = N_s$  and for  $1 \leq i \leq t$  let  $K_i$  denote the subgroup of  $N_s$  that fixes the target points  $\delta_1, \dots, \delta_i$ .

First, we prove by induction that there are elements  $h_i \in K_i$ ,  $0 \leq i < t$ , such that  $xh_0 \dots h_i$  fixes every element of the set  $\{\delta_1, \dots, \delta_{i+1}\}$ . Let  $x$  map  $\delta_1$  to  $\delta'_1$ . Since  $K_0|_{\Omega_1}$  is transitive on  $\Omega_1$  there is an element  $h_0 \in K_0$  that maps  $\delta'_1$  to  $\delta_1$ . Hence  $xh_0$  fixes  $\delta_1$ . Inductively assume that there exists elements  $h_j$ ,  $0 \leq j < i$  such that  $xh_0 \dots h_{i-1}$  fixes the target points  $\delta_1, \dots, \delta_i$ . Let  $xh_0 \dots h_{i-1}$  map  $\delta_{i+1}$  to  $\delta'_{i+1}$ . Since  $K_i$  is nontrivial on  $\Omega_{i+1}$ , it follows from part 3 of Proposition 5.28 that there is an element  $h \in K_i$  that maps  $\delta'_{i+1}$  back to  $\delta_{i+1}$ . Let  $h_i = h$ . The group  $K_i$  is trivial on all the  $G$ -orbits  $\Omega_1, \dots, \Omega_i$  and therefore  $xh_1 \dots h_i$  fixes all the points in the set  $\{\delta_1, \dots, \delta_{i+1}\}$ . The element  $y = h_0 \dots h_{t-1} \in N_s$  is such that  $xy$  fixes all the target points of  $Y_s$ . We now give the  $\text{FL}^{\text{Mod}_p\text{L}}$  algorithm for computing  $y$ .

Let  $x$  map  $\delta_i$  to  $\nu_i$ . We want to compute an element  $y \in N_s$  that maps  $\nu_i$  to  $\delta_i$  for all  $i \in Y_s$ . To this end consider the vector space  $V' = \bigoplus_{i \in Y_s} V_i$  and let  $U'$  be the projection of  $U$  onto  $V'$ . Recall that the groups  $R_{i,s}$  and  $N_{s+1}|_{\Omega_i}$  are trivial for all  $i \in Y_s$ . Therefore the vector spaces  $V'$  and  $U'$  are isomorphic to the groups  $R_s$  and  $N_s$  restricted to the target orbits of  $Y_s$ . For an element  $x \in R_s$  let  $\mathbf{u}_x$  be the image of  $x$  in  $V'$  under this restriction. In fact  $\mathbf{u}_x$  is the projection of  $\mathbf{v}_x$  onto  $V'$ . Analogues to Proposition 5.30, using Proposition 5.21 we compute in  $\text{FL}^{\text{Mod}_p\text{L}}$  a subset  $B' = \{x_1, \dots, x_t\}$  of  $B$  such that  $\mathcal{B}' = \{\mathbf{u}_{x_1}, \dots, \mathbf{u}_{x_t}\}$  forms a basis for  $U'$ . First we show that  $\mathbf{u}_y$  can be computed in FL and then we recover the permutation  $y$  in  $\text{FL}^{\text{Mod}_p\text{L}}$ .

Consider an  $i \in Y_s$ . Since  $R_{i,s}$  is a constant sized transitive permutation group on  $\Omega_i$ , in  $\text{FL}^{\text{Mod}_p\text{L}}$  we can compute an element  $y_i \in R_{i,s}$  that maps  $\nu_i$  back to  $\delta_i$ . The group  $N_s|_{\Omega_i} = \text{Soc}(G_i)$  and by the O'Nan-Scott theorem  $\text{Soc}(G_i)$  has a regular action on  $\Omega_i$ . The element  $yy_i^{-1} \in R_s$  fixes the point  $\nu_i$  and therefore restricted to  $\Omega_i$  is trivial. As a result if  $\mathbf{w}_i$  denotes the projection of  $\mathbf{u}_y$  onto  $V_i$  then  $\mathbf{w}_i = \mathbf{v}_{y_i}$ . Since  $R_{i,s}$  is a group of order bounded by  $c!$ , in FL we can compute the vector  $\mathbf{w}_i$ . The vector  $\mathbf{u}_y$  is given by  $\sum_{i \in Y_s} \mathbf{w}_i$  which can also be computed in FL.

To complete the algorithm we need to recover  $y$  from the vector  $\mathbf{u}_y$ . Using Proposition 5.21 we compute, in  $\text{FL}^{\text{Mod}_p\text{L}}$ , elements  $a_1, \dots, a_t \in \mathbb{F}_p$  such that  $\mathbf{u}_y = \sum_{i=1}^t a_i \mathbf{u}_{x_i}$ . The permutation  $y = \prod_{i=1}^t x_i^{a_i}$  is the required element of  $N_s$ .  $\square$

### 5.5.2 Computing the critical orbits: nonabelian case

Consider any  $i \in X_s$ . By O’Nan-Scott theorem  $\text{Soc}(G_i)$  is either  $K$  (type 2) for some minimal normal subgroup  $K$  of  $G_i$  or is of the form  $K_1 \times K_2$  (type 3) where  $K_1$  and  $K_2$  are the only minimal normal subgroups of  $G_i$ . For each  $i \in X_s$ , by a *socle part* associated to  $i$  we mean a minimal normal subgroup of  $G_i$ . For a  $T$ -semisimple group  $L = T_1 \times \dots \times T_r$  by a *simple part* we mean one of the subgroup  $T_i$ .

The quotient group  $N_s/N_{s+1}$  is a subgroup of the  $T_s$ -semisimple group  $R_s/R_{s+1}$ . Hence by Scott’s Lemma (Lemma 3.6),  $N_s/N_{s+1}$  is a product of diagonals of simple parts of  $R_s/R_{s+1}$ . Consider two simple parts  $T'$  and  $T''$  of  $R_s/R_{s+1}$ . As before we say that  $T'$  and  $T''$  are *linked* if in  $N_s/N_{s+1}$ ,  $T'$  and  $T''$  are in the same diagonal component. We now extend the “linking” relation to socle parts. Any socle part  $K$  is the product of certain subset of simple parts of  $R_s/R_{s+1}$ . We say that the socle parts  $K'$  and  $K''$  are *linked* if  $K' = \dots \times T' \times \dots$  and  $K'' = \dots \times T'' \times \dots$  such that  $T'$  and  $T''$  are linked. For socle parts  $K'$  and  $K''$  we prove that either they are fully linked or are unlinked.

**Proposition 5.33.** *Let  $K' = T'_1 \times \dots \times T'_u$  and  $K'' = T''_1 \times \dots \times T''_v$  be two socle parts. If  $K'$  and  $K''$  are linked then  $u = v$  and there is a permutation  $\pi \in S_u$  such that  $T'_i$  is linked to  $T''_{i\pi}$ .*

*Proof.* Let  $K'$  and  $K''$  be socle parts corresponding to orbits  $\Omega'$  and  $\Omega''$ . Let us assume without loss of generality that  $T'_1$  is linked to  $T''_1$ . Since  $K'$  is the minimal subgroup of  $G|_{\Omega'}$ , for any  $i$  there is an element  $g \in G$  such that  $g^{-1}T'_1g = T'_i$  (Lemma 5.7). The element  $g$  maps via conjugation  $T''_1$  to some  $T''_i$ . Thus for any simple part  $T'_i$  in  $K'$  there is a simple part  $T''_i$  in  $K''$  such that  $T'_i$  and  $T''_i$  are linked. However no two simple parts of  $K'$  are linked. Each simple part of  $K'$  therefore, is linked to distinct simple part of  $K''$ . By interchanging the role of  $K'$  and  $K''$  we can prove the converse. As a result, we have  $u = v$  and  $\pi \in S_u$  is the permutation that maps  $i$  to  $j$  if  $T'_i$  is linked to  $T''_j$ .  $\square$

Recall that for target reduction our goal is to (1) compute the set  $Y_s$  of critical orbits, (2) compute a generator set of subgroup  $N$  of  $N_s$  that fixes all the points of  $Y_s$  and (3) for each  $x \in G$  an element  $y \in N_s$  such that  $x^* = xy$  is trivial on all the target points of  $Y_s$ . We now show that each of these three tasks can be achieved by an FL algorithm.

## Computing $Y_s$

Let  $\mathcal{K}$  be the collection of socle parts of orbits of  $X_s$ . To construct critical subset  $Y_s$  of  $X_s$  consider the graph  $\mathcal{G} = (\mathcal{K}, \mathcal{E})$  where the edge set  $\mathcal{E}$  is partitioned into the set of red edges  $\mathcal{R}$  and the set of blue edges  $\mathcal{B}$ . The red edges  $\mathcal{R}$  consists of all unordered pairs  $\{K_1, K_2\}$  where  $K_1$  and  $K_2$  are linked. On the other hand the blue edges consists of all unordered pairs  $\{K_1, K_2\}$  such that  $K_1$  and  $K_2$  are distinct socle parts of the same  $G$ -orbit. We have the following proposition about the structure of the graph  $\mathcal{G}$ .

**Proposition 5.34.** *The red subgraph, i.e.  $\mathcal{G}_{\text{red}} = (\mathcal{K}, \mathcal{R})$ , consists of disconnected cliques and any blue edge is between two disconnected red cliques.*

*Proof.* The “linking” relation is an equivalence relation and hence the red subgraph consists of disconnected red cliques. Any blue edge is between two socle parts of the same  $G$ -orbit. Hence they cannot be linked. Therefore blue edges are always between two disconnected red cliques in the red subgraph.  $\square$

In logspace we compute the set  $\mathcal{C}$  of red cliques in the red subgraph  $\mathcal{G}_{\text{red}}$ . We partition the set  $\mathcal{C}$  into subsets  $\mathcal{C}'$  and  $\mathcal{C}''$ , where a red clique  $C$  is put in  $\mathcal{C}'$  if  $C$  contains an element  $K = \text{Soc}(G_i)$  for some  $i \in X_s$ . The remaining cliques are put in  $\mathcal{C}''$ . We now construct the subset of critical orbits  $Y_s$  as the union of  $Y'_s$  and  $Y''_s$ .

The set  $Y'_s$  consists of one index  $i$  per clique  $C \in \mathcal{C}'$  such that  $K = \text{Soc}(G_i) \in C$ . Shrink all the red cliques in  $\mathcal{G}$  and delete all vertices (and blue edges incident on them) that corresponds to cliques in  $\mathcal{C}'$ . Call the new graph  $\mathcal{G}'$ . In  $\mathcal{G}'$ , compute the lexicographically first spanning forest of blue edges. Let  $\mathcal{B}'$  be the blue edges in the spanning forest. Recall that each  $e \in \mathcal{B}'$  corresponds to the orbit  $\Omega_i$  where  $\text{Soc}(G_i) = K \times K'$ . The subset  $Y''_s$  of the critical subset  $X''_s$  consist of such indices  $i$  corresponding to edges in  $\mathcal{B}'$ . We prove the following proposition

**Proposition 5.35.** *The set  $Y_s = Y'_s \cup Y''_s$  can be computed in logspace. Let  $N$  be the subgroup of  $N_s$  that fixes all the target points of  $Y_s$  then  $N|_{\Omega_i} \leq R_{i,s}$  for all  $i \in X_s$ .*

*Proof.* The sets  $Y'_s$  and  $Y''_s$  can be computed in logspace as this involves reachability in undirected graphs (Lemma 2.1).

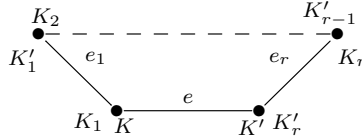
Let  $\Delta'$  be the subset containing all the target points of  $Y_s$  and let  $N = N_s(\Delta')$ . Depending on whether  $\text{Soc}(G_i)$  has one or two socle parts we have the following two cases.

**Case 1:** The socle  $\text{Soc}(G_i)$  is itself a socle part  $K$ . Consider the red clique  $C \in \mathcal{C}'$  that contains  $K$ . By the construction of  $Y'_s$  there is a  $j \in Y'_s$  such that  $\text{Soc}(G_j) = K' \in C$ . Since  $K'$  and  $K$  are linked, any  $h \in N_s$  when restricted to  $\Omega_i \cup \Omega_j$  is of the form  $\langle \phi(h'), h' \rangle$ ,  $h' \in K'$ , for some isomorphism  $\phi$  from  $K'$  to  $K$ . Therefore  $N$  when restricted to  $\Omega_i$  is  $\phi(K'_{\delta_j})$ . By O’Nan-Scott’s theorem  $K'$  is transitive and hence  $K'_{\delta_j}$  is a proper subgroup of  $K'$ . Thus  $N$  restricted to  $\Omega_i$  is also a proper subgroup of  $K = R_{i,s}$ .

**Case 2:** The socle  $\text{Soc}(G_i)$  is of the form  $K \times K'$ . Then there is blue edge  $e = \{K, K'\}$  in the graph  $\mathcal{G}$ . Firstly if  $e$  is one of the edges in the maximal spanning forest  $\mathcal{B}'$  then  $N$  fixes the target point corresponding to  $e$ . The group  $N|_{\Omega_i}$  is a diagonal group  $\text{Diag}(K \times K')$  (O’Nan-Scott theorem) and hence a proper subset of  $K \times K'$ . Thus we have disposed the case when  $e$  is an edge of the spanning forest  $\mathcal{B}'$ .

We now handle the case when  $e$  is not an edge of the spanning forest  $\mathcal{B}'$ . Suppose that the edge  $e = \{K, K'\}$  connects the distinct red cliques  $C_1$  and  $C_2$  with  $K \in C_1$  and  $K' \in C_2$ . If  $C_1$  (or  $C_2$ ) is a clique in  $\mathcal{C}'$  then there is a  $j \in Y'_s$  such that  $\text{Soc}(G_j) = K_j \in C_1$  (or  $C_2$ ). By an argument similar to the Case 1 it follows that  $N$  restricted to  $K$  is a strict subgroup  $K''$  isomorphic to the subgroup of  $K_j$  that fixes  $\delta_j$ . Hence  $N$  restricted to  $\Omega_i$  is  $K'' \times K'$  which is a strict subgroup of  $R_{i,s} = K \times K'$ .

Suppose that both  $C_1$  and  $C_2$  are cliques of  $\mathcal{C}''$ . Then since  $\mathcal{B}'$  forms a maximal spanning forest, adding edge  $e$  to  $\mathcal{B}'$  gives a cycle  $e_1, \dots, e_r, e$  (see the figure below).



Let  $\Omega_{j_t}$  be the orbit that corresponds to the edge  $e_t$  and let  $\text{Soc}(G_{j_t}) = K_t \times K'_t$ . The group  $N$  fixes all the points  $\delta_{j_t} \in \Omega_{j_t}$ . By case 3 of O’Nan-Scott theorem it follows that  $N|_{\Omega_{j_t}}$  is the diagonal group  $\text{Diag}(K_t \times K'_t)$ . Note that  $K_1$  and  $K'_t$  are linked to  $K$  and  $K'$  respectively in  $N_s$ . Hence the group  $N$  restricted to  $\Omega_i$  is a diagonal group  $\text{Diag}(K \times K')$  which is a strict subgroup of  $R_{s,t} = K \times K'$ .  $\square$

### Computing $x^*$

Given an  $x \in G$  we give an FL algorithm to compute a  $y \in N_s$  such that  $x^* = xy$  fixes all target points in  $Y_s$ . For any  $i \in Y_s$  let the target point  $\delta_i$

of  $\Omega_i$  be mapped to  $\nu_i$ . Then we want to find a  $y$  in  $N_s$  that maps  $\nu_i$  back to  $\delta_i$ .

**Proposition 5.36.** *Given an  $i \in X_s$ . There is an FL algorithm to compute a subset  $D_i \subseteq N_s$  elements such that (1) the projection of  $D_i$  to  $\text{Soc}(G_i)$  is one-to-one and (2) for any socle part  $K'$  of  $\text{Soc}(G_j)$ ,  $j \in X_s$ ,  $D_i$  projected to  $K'$  is trivial if  $K'$  is not linked to any of the socle parts of  $\text{Soc}(G_i)$ .*

*Proof.* Let  $R_s/R_{s+1} = T_1 \times \dots \times T_u$  where each  $T_i$  is isomorphic to  $T$ . Since  $N_s/N_{s+1} \hookrightarrow R_s/R_{s+1}$  there exists a partition  $\mathcal{I} = \{I_1, \dots, I_t\}$  of indices  $1, \dots, u$  such that  $N_s/N_{s+1}$  is the product  $\prod_{k=1}^t \text{Diag}\left(\prod_{j \in I_k} T_j\right)$ . We have computed the strong generator set  $C$  of  $N_s \text{ rel } N_{s+1}$  as part of the SGS of  $G$ . Recall that the strong generator set  $C$  of  $N_s \text{ rel } N_{s+1}$  consists of subset  $C_1, \dots, C_t$  where the subset  $C_k$  corresponds to the diagonal group  $\text{Diag}\left(\prod_{j \in I_k} T_j\right)$ , i.e. the projection of  $C_k$  on  $T_j$  is the group  $T_j$  if  $j \in I_k$  and 1 otherwise.

Let  $x_1, \dots, x_r$  be the elements of  $C$  whose action on  $\Omega_i$  is nontrivial. Then for any other socle part  $K_j$  that is not linked to any of the socle parts of  $\text{Soc}(G_i)$ ,  $x_i$ 's are trivial on  $K_j$ . Furthermore if  $z_i$  denotes the projection of  $x_i$  onto  $\text{Soc}(G_i)$ , then  $z_1, \dots, z_r$  generates  $\text{Soc}(G_i)$ . For each element  $z \in K$  we express  $z$  as a product  $z = z_{i_1} \dots z_{i_k}$ . Include into  $D_i$  the element  $x_z = x_{i_1} \dots x_{i_k}$ . Since  $\text{Soc}(G_i)$  is a constant sized group and each  $x_i$ 's are elements of the group  $G$  with constant sized orbits,  $D_i$  can be computed in FL. □

*Remark 5.37.* Consider the SGS  $C = \cup_{k=1}^t C_k$  of  $N_s \text{ rel } N_{s+1}$  where  $C_k$  corresponds to the diagonal component  $\text{Diag}\left(\prod_{j \in I_k} T_j\right)$ . For a  $j \in X_s$  the elements of  $C_k$  is nontrivial on  $\Omega_j$  if and only if a  $\text{Soc}(G_j) = \dots \times T_r \times \dots$  and  $r \in I_k$ . It follows from the proof of Proposition 5.36 that we can ensure  $C_k \subseteq D_j$  for any  $j$  such that  $C_k$  is nontrivial on  $\Omega_j$ . Therefore the set  $\cup_{i \in X_s} D_i \cup C'$  is a generator set for  $N_s \text{ rel } N_{s+1}$  where  $C'$  denotes the elements of  $C$  that are trivial on all the target orbits.

We now prove that given any  $x \in G$  we can compute in FL an element  $y \in N_s$  such that  $x^* = xy$  fixes all the target points of  $Y_s$ . Intuitively we want to choose an element  $y$  in  $N_s$  that “negates” the effect of  $x$  on  $\delta_i$  for all  $i \in Y_s$ .

First we handle the target points in  $Y'_s$ . Each  $i \in Y'_s$  can be handled independent of the other target points in  $Y_s$ , i.e. we can compute elements

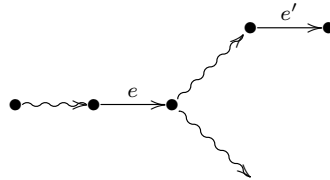
$y_i$  such that  $xy_i$  fixes  $\delta_i$  and for all  $j$  in  $Y_s \setminus \{i\}$ ,  $y_i|_{\Omega_j}$  is 1. That such an element exists follows from the fact that (1)  $\text{Soc}(G_i)$  is transitive and (2) none of the socle parts of  $\text{Soc}(G_j)$  is linked to  $K_i = \text{Soc}(G_i)$  for all  $j \in Y_s \setminus \{i\}$ . We compute  $y_i$  in FL using Proposition 5.36.

The target points in  $Y_s''$  however cannot be handles independently, i.e. the choice of a  $y_i$  for some  $i \in Y_s''$  will have a nontrivial action on some of the other target orbits in  $j \in Y_s''$  as illustrated below. Recall that for each  $i \in Y_s'$  there is an edge  $e_i \in \mathcal{B}'$ . The difficulty arises for  $i$  and  $j$  in  $Y_s''$  for which the edges  $e_i$  and  $e_j$  share a common vertex in  $\mathcal{G}'$  (see figure below).



In such a case  $\text{Soc}(G_i) = K_i \times K'_i$  and  $\text{Soc}(G_j) = K_j \times K'_j$  and  $K_i$  and  $K_j$  are linked. Thus any nontrivial element chosen from  $K_i$  will have a nontrivial action on  $\Omega_j$ . Any element  $y_j$  that we choose for the orbit  $\Omega_j$  has to negate this “propagated” effect. The main idea therefore is to systematically choose elements  $y_e$  for each edge in  $\mathcal{B}'$  keeping in view this propagated effect.

Recall that we have computed a lexicographically least maximal spanning forest  $\mathcal{F}$  of  $\mathcal{G}'$  with edge set  $\mathcal{B}' \subseteq \mathcal{B}$ . Each tree  $\mathcal{T}$  in the forest  $\mathcal{F}$  can be considered as a rooted tree with root at the lexicographically least vertex of  $\mathcal{T}$ . Thus each edge in  $\mathcal{B}$  acquires a direction: the tail at the vertex closer to the root (see figure below). This gives a partial order on  $\mathcal{B}'$ : edges  $e < e'$  if  $e$  and  $e'$  belong to the same tree and the unique path from the root to the tail of  $e'$  contains the edge  $e$ .

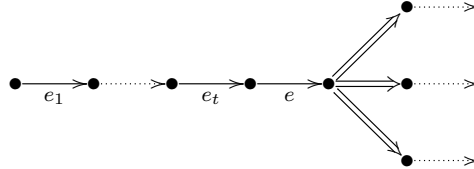


Since the vertices of  $\mathcal{G}'$  corresponds to red-cliques in  $\mathcal{G}$  which in turn corresponds to linked socle parts the following proposition is direct.

**Proposition 5.38.** *Let  $e$  be any edge with tail at  $K_1$  and head at  $K_2$ . Fix an edge  $e' \not\sim e$  and let  $\Omega_i$  be the orbit corresponding to  $e'$ . Then the socle part  $K_2$  is not linked to any of the socle parts of  $\text{Soc}(G_i)$ .*

Consider any edge  $e \in \mathcal{B}'$  and let  $\mathcal{T}_e$  be rooted tree in the forest  $\mathcal{B}'$  containing  $e$ . We give an FL algorithm (Algorithm 7) to compute a permutation

$y_e \in N_s$  that is trivial on all orbits corresponding to edges  $e' \not\prec e$  and for the unique path  $e_1, \dots, e_t$  from the root of  $\mathcal{T}_e$  to the tail of  $e$ , negates the action of  $xy_{e_1} \dots y_{e_t}$  on the target point corresponding to  $e$ . In fact  $y_e$  will be trivial on all orbits other than those that corresponds to  $e$  and edges going out of the head of  $e$  (indicated by a double arrow in the figure).



The main idea behind algorithm is that once the path  $e_1, \dots, e_t$  is computed (which can be done in FL using Reingold's algorithm [57]), the propagated effect of  $y_{e_i}$ 's on the orbit associated to  $e$  can be kept track of using constant amount of space. This is because for each  $i$  since  $e_i < e_j$  the permutation  $y_{e_i}$  is trivial on all the orbits associated to  $e_j$  for  $j > i+1$ . Let  $e$  be the directed edge  $(K_1, K_2)$  then since  $K_2$  is transitive on the orbit associated to  $e$ , the permutation  $y_e$  will be chosen such that  $y_e$  does not effect any of the orbits associated to edges  $e' \not\prec e$ . Proposition 5.38 guarantees that such a  $y_e$  exists.

**Proposition 5.39.** *Algorithm 7 is a FL algorithm.*

*Proof.* The algorithm can be seen as the composition of two stages (1) computing the paths  $e_1, \dots, e_t$  (step 1 of Algorithm 7) and (2) computing  $y_e$  (loop 2 of Algorithm 7). Computing the path involves applying Reingold's  $s$ - $t$  connectivity algorithm [57] (also Lemma 2.1) repeatedly and hence is in FL. The permutation  $y_i$  (step 3 of Algorithm 7) is computed by first computing the set  $D_i$  (using Proposition 5.36). Recall that  $D_i$  projects onto  $\text{Soc}(G_i) = K_1 \times K_2$  and  $K_2$  is transitive (O'Nan-Scott). Hence in FL by examining all the  $\#\text{Soc}(G_i)$  elements of  $D_i$  we can find a desired  $y_i$ . All these can be achieved by logspace bounded computations.  $\square$

Let  $e_1, \dots, e_r$  denote a topological sorting of edges in  $\mathcal{B}'$  and let  $y'' = y_{e_1} \dots y_{e_r}$  then  $y''$  is trivial on all orbits of  $Y'_s$  and  $xy''$  fixes all the target points in  $Y''_s$ . Recall that we have already computed a  $y'$  such that  $y'$  is trivial on all the orbits in  $Y''_s$  and  $xy'$  fixes all points of  $Y'_s$ . Let  $x^* = xy'y''$ .

**Proposition 5.40.** *The element  $x^* \in G$  fixes all the points in  $Y_s$ .*

*Proof.* We prove by induction. Let  $z = xy'$  then as argued before  $z$  fixes all the target points on  $Y'_s$ . Consider any topological ordering  $e_1, \dots, e_r$  of

**Input:** The permutation  $x \in G$  and an edge  $e \in \mathcal{B}'$

**Output:** The permutation  $y_e$

- 1 Compute a path  $e_1, \dots, e_t = e$  from the corresponding root of the tree containing  $e$  to  $e$ .  
Let  $\Sigma_i$  be the  $G$ -orbit associated to  $e_i$ .  
Let  $\delta_i$  be the target point associated with the edge  $e_i$ .  
 $y^* \leftarrow 1$ .
- 2 **for**  $i = 1$  **to**  $t$  **do**
  - $\delta \leftarrow \delta_i^{xy^*}$ .
  - if**  $\delta = \delta_i$  **then**  $y_i \leftarrow 1$ .
  - else**
    - Let  $e_i$  be the directed edge  $(K_1, K_2)$  (tail at  $K_1$  and head at  $K_2$ ).
    - 3 Using Proposition 5.36 compute  $y_i$  such that  $\delta^{y_i} = \delta_i$  with the additional property that  $y_i$  is trivial and all socle parts not linked to  $K_2$ .
  - end**
  - if**  $i < t$  **then**  $y^* = y_i|_{\Sigma_{i+1}}$ .
- end**
- return**  $y_t$

**Algorithm 7:** Computing the permutations  $y_e$ .

edges in  $\mathcal{B}'$ . Let  $\Omega_i$  denote the  $G$ -orbit corresponding to  $e_i$ . Let  $0 \leq k \leq r+1$  be the largest index such that the permutation  $z$  fixes the target points of  $\cup_{j=1}^{k-1} \Omega_j$ . If  $k = r+1$  then we are through. Otherwise Algorithm 7 computes a  $y_{e_k}$  such that  $zy_{e_k}$  fixes  $\delta_k$ , the target point of  $\Omega_k$ . Let  $e_k$  be the directed edge  $(K_1, K_2)$  with the head at  $K_2$ . By step 3 we have ensured that  $y_{e_k}$  projected onto  $\Omega_k$  is in  $K_2$ . Since  $e_1, \dots, e_r$  is a topological sort,  $e_i \not\prec e_k$  for every  $i < k$ . Therefore for any  $i < k$ , the socle parts of  $\text{Soc}(G_i)$  is not linked to  $K_2$  (Proposition 5.38). Hence  $y_i$  is trivial on all  $\Omega_i$  for  $1 \leq i < k$  and  $z' = zy_{e_k}$  fixes all the target points in  $\cup_{j=1}^k \Omega_j$ . We now repeat the argument with  $z$  replaced with  $z' = zy_{e_k}$ . At each stage  $k$  increase by 1 and hence after  $r$  steps  $k = r+1$  are we are through.  $\square$

*Remark 5.41.* Consider any  $x \in G$  and  $x^* = xy$  where  $y \in N_s$  be the permutation computed by our algorithm such that  $x^*$  fixes all points in  $Y_s$ . Let  $e = (K_1, K_2)$  be a minimal edge in the  $<$  order (or in other words an edge going out of a root node) with head at  $K_2$ . Then it is straight forward to verify that  $y$  restricted to  $K_1$  is trivial. This follows from the choice of

$y_i$ 's in Algorithm 7.

Given an element  $x \in G$  we have associated for each  $i \in Y_s$  an element  $y_i$  such that  $x \prod_{i \in Y_s} y_i$  (note the order in which the indices of  $Y_s''$  are taken does matter; the corresponding edges should be topologically sorted). We can ensure that the choice of  $y_i$ 's depended only on the action of  $x$  on the target points of  $Y_s$ . In other words for two elements  $x_1$  and  $x_2$  in  $G$  and let  $\{y_i\}_{i \in Y_s}$  and  $\{y'_i\}_{i \in Y_s}$  be the elements chosen. Then if  $\delta^{x_1} = \delta^{x_2}$  for all target point  $\delta$  of  $Y_s$ ,  $y_i = y'_i$  for all  $i \in Y_s$  and

$$\prod_{i \in Y_s} y_i = x_1^{*-1} x_1 = x_2^{*-1} x_2 = \prod_{i \in Y_s} y'_i.$$

Furthermore we can assume that if  $x$  fixes all the target points of  $Y_s$  then each of the  $y_i = 1$  for all  $i \in Y_s$ . We will make this additional assumption which will be helpful in computing the generator set for  $N$ .

### Computing $N$

Finally we give the FL algorithm to compute the generator set of  $N$ , the subgroup of  $N_s$  that fixes all the target points in  $Y_s$ . To this end we examine the strong generator  $C$  of  $N_s$  rel  $N_{s+1}$  which we have computed as part of the generator set of  $G$ .

Let  $R_s/R_{s+1} = T_1 \times \dots \times T_u$  where each  $T_i$  is isomorphic to a non-abelian simple group  $T$ . We have a partition  $\mathcal{I} = \{I_1, \dots, I_t\}$  of indices  $\{1, \dots, u\}$ , such that the quotient group  $N_s/N_{s+1}$  is the product of diagonals  $\prod_{k=1}^t \text{Diag} \left( \prod_{j \in I_k} T_j \right)$ . Recall that the strong generator set  $C$  of  $N_s$  rel  $N_{s+1}$  consists of subset  $C_1, \dots, C_t$  where each  $C_k$  corresponds to the diagonal group  $\text{Diag} \left( \prod_{j \in I_k} T_j \right)$ .

Without loss of generality we assume that  $Y_s = \{1, \dots, r_1, r_1 + 1, \dots, r\}$  where  $Y'_s = \{1, \dots, r_1\}$  and  $Y''_s = \{r_1 + 1, \dots, r\}$ . We assume further without loss of generality that the ordering  $r_1 + 1, \dots, r$  of elements of  $Y''_s$  is compatible with the ordering of edges in the forest  $\mathcal{F}$ , i.e. if  $e_j$  is the edge associated to  $j > r_1$  then  $e_{r_1+1}, \dots, e_{r_2}$  is a topological sort of edges of  $\mathcal{F}$ .

Using Proposition 5.36 in FL for each  $i \in Y_s$  compute the sets  $D_i$  such that (1) the projection of  $D_i$  on to  $\text{Soc}(G_i)$  is one-to-one and (2) for any socle part  $K$  of  $\text{Soc}(G_j)$  not linked to socle parts of  $\text{Soc}(G_i)$  the projection of  $D_i$  is 1.

Recall that for each  $x \in G$  we gave an FL to compute  $x^* = xy$ ,  $y \in N_s$  such that  $x^*$  fixes all the target points of  $Y_s$ . This we achieved by computing

for each  $i \in Y_s$  an element  $y_i \in N_s$  such that  $xy_1 \dots y_{r_1+r_2} = x^*$ . Furthermore we assume that the  $y_i$ 's are canonical as described in Remark 5.41. For each  $i \in Y_s$  let  $D_i^* = \{x^* | x \in D_i\}$ .

**Proposition 5.42.** *Let  $A$  be a generator set of  $N_{s+1}$  and let  $C'$  be the elements of  $C$  that is trivial on the target orbits of  $Y_s$ . Then  $D^* = (\cup_{i \in Y_s} D_i^*) \cup C' \cup A$  is a generator set of  $N$ , the subgroup of  $N_s$  that fixes all the target points of  $Y_s$ .*

*Proof.* Clearly every element of  $D^*$  is contained in  $N$  therefore  $N^*$ , the group generated by  $D^*$ , is contained in  $N$ . We now prove the converse.

It follows from Remark 5.37 that  $D = (\cup_{i \in Y_s} D_i) \cup C'$  forms a generator set of  $N_s$ . Hence any element  $x \in N$  can be written as  $x = x_1 \dots x_r y z$  where  $x_i \in D_i$ ,  $y$  is in the group generated by  $C'$  and  $z \in N_{s+1}$ .

For any  $i \in Y'_s$  notice that  $x_i$  is trivial on all target orbits in  $Y_s \setminus \{i\}$ . Hence  $x_i$  fixes all the target points of  $Y_s$ . By Remark 5.41 it follows that  $x_i^* = x_i \in D_i^*$ . Thus to prove that  $x$  is in  $N^*$  it is sufficient to prove that  $z = x_{r_1+1} \dots x_r \in N^*$ . Let  $e_j$  denote the edge corresponding to the point  $j \in Y''_s$ . Let  $k \leq r+1$  be the largest integer such that  $x_j$  restricted to  $\Omega_j$  is 1 for all  $r_1 < j < k$ . By the construction of  $D_k^*$  we have  $x_k^* \in D_k^*$  such that  $x_k^* = x_k y$  fixes all the target points of  $Y_s$  where  $y$  satisfies the properties of Remark 5.41.

Consider the permutation  $z' = x_k^{*-1} z = y x_{k+1} \dots x_r$ . Since both  $z$  and  $x_k^{*-1}$  fixes the target points of  $Y_s$  so does  $z'$ . We show that  $z'$  is trivial on all orbits  $\Omega_j$ ,  $1 \leq j \leq k$ . First for any  $1 \leq i < k$ ,  $x_j$  is trivial for all  $j > k$ . Recall that if  $e_k$  be the directed edge  $(K_1, K_2)$  with the head at  $K_2$  then  $y$  is trivial on  $\Omega_j$  for all  $j < k$  and projected to  $\Omega_k$ , is an element of the subgroup  $K_2$  (this follows from Remark 5.41). Therefore  $z' = y x_{k+1} \dots x_r$  is trivial on  $\Omega_j$  for all  $r_1 < j < k$ . Furthermore note that none of the socle parts of  $\text{Soc}(G_j)$  is linked to  $K_1$  for  $k < j \leq r$  and hence  $x_j$  projected on  $\Omega_k$  is also an element of  $K_2$ . This proves that  $z'$  projected to  $\Omega_k$  is an element of  $K_2$ . However by the O'Nan-Scott theorem  $K_2$  is regular. This is possible if and only if  $z'$  restricted to  $\Omega_k$  is 1 as  $z'$  fixes the target point of  $\Omega_k$ .

We repeat this argument with  $z$  replaced with  $z'$  and in each step  $k$  increases by at least 1. Thus it follows that there exists element  $x_j^* \in D_j^*$ ,  $r_1 < j \leq r$ , such that  $z \prod_{j=r_1+1}^r x_j^*$  is 1 on all the target orbits of  $Y_s$  and hence is of the form  $gh$  where  $g$  is in the group generated by  $C'$  and  $h$  is in  $N_{s+1}$ . It follows that  $x$  is in the group generated by  $D^*$ . □

Given  $x \in G$  since  $x^*$  can be computed in FL it follows that a generator

set for  $N$  can be computed in FL. This completes the algorithm for target reduction in the nonabelian case.

**Theorem 5.43.** *Given an instance  $(G, \Omega, \Delta)$  of  $PWS_c$  subgroup  $G'$  of  $G$  satisfying the properties (1)  $G \geq G' \geq G(\Delta)$  and (2) for all  $G$ -orbit  $\Sigma$  such that  $\Sigma \cap \Delta \neq \emptyset$ ,  $G|_{\Sigma} > G'|_{\Sigma}$  can be computed in the  $\text{Mod}_k\text{L}$  hierarchy where  $k$  is the product of all primes less than  $c$  and the level of the hierarchy is a constant that depends only on  $c$ .*

## 5.6 Complexity of $\text{BCGI}_b$

We now show that  $PWS_c$  is in the  $\text{Mod}_k\text{L}$ -hierarchy. The complete algorithm is given below (Algorithm 8). Each iteration of the loop 1 is in the  $\text{Mod}_k\text{L}$ -hierarchy: step 2 uses Theorem 5.24 and step 3 uses Theorem 5.43. Since the  $G$ -orbits are of size bounded by  $c$ , we will have to iterate through the loop 1 at most  $c \cdot \log c$  times before each point in  $\Delta$  is a  $G$ -orbit in itself (i.e.  $H$  is  $G(\Delta)$ ).

**Input:** An instance  $(G, \Delta)$  of  $PWS_c$

**Output:** A generator set for  $G(\Delta)$ .

$H \leftarrow G$ .

1 **repeat**

2     Compute a strong generator set for  $H$  with respect to a locally residual series.

3     Compute the generator set of  $H'$  such that  $H \geq H' \geq H(\Delta)$  and such that  $H'|_{\Sigma} < H|_{\Sigma}$  for each  $H$ -orbit  $\Sigma$  containing a point of  $\Delta$ .  
 $H \leftarrow H'$

**until**  $H$  fixes all points in  $\Delta$ ;

**return** the generator set for  $H$

**Algorithm 8:** Complete algorithm for  $PWS_c$

Using Proposition 5.4 we have the main theorem of this chapter.

**Theorem 5.44.** *The  $PWS_c$ ,  $\text{AUT}_b$  and  $\text{BCGI}_b$  are in the  $\text{Mod}_k\text{L}$ -hierarchy.*

## 5.7 Discussion

We have proved the  $\text{BCGI}$  is in the  $\text{Mod}_k\text{L}$ -hierarchy. In fact we proved this by proving that  $PWS_c$  is in the  $\text{Mod}_k\text{L}$ -hierarchy. The algorithm involved two stages; computing the strong generator set and the target reduction.

Both these stages handled the abelian and non-abelian quotients separately. It is surprising that even though the group theory is more involved the non-abelian quotients could be handled in logspace where as the abelian quotient required  $\text{Mod}_p\text{L}$  as an oracle for some appropriate prime  $p$ . Thus if the composition series of  $G$  had only nonabelian simple groups then  $\text{PWS}_c$  for  $(G, \Omega, \Delta)$  can be solved in logspace. In view of the hardness result of Torán [68], we cannot improve on the complexity of handling the abelian quotients unless  $\text{Mod}_p\text{L}$  is in  $\text{L}$ .

There is a gap between our upper bound for  $\text{BCGI}$  and the lower bound that follows from Torán's results [68]. It follows from Torán's result that  $\text{BCGI}_b$  is hard for the  $l$ th level  $\text{Mod}_k\text{L}$  where  $b$  is exponential in  $k$  and  $l$ . Our upper bound however places  $\text{BCGI}_b$  is a higher level of the  $\text{Mod}_k\text{L}$ -hierarchy.

## Chapter 6

# Computational Galois theory

We now move on to the next part of this thesis where we show upper bounds on certain computational problems in Galois theory. Given a polynomial  $f(X)$  of degree  $n$  over  $\mathbb{Q}$  we are interested in the following three fundamental tasks.

1. Compute the Galois group as a permutation group on the roots of  $f(X)$ ,
2. Compute the order of the Galois group of  $f(X)$  or equivalently the degree  $[\mathbb{Q}_f : \mathbb{Q}]$  of the splitting field extension of  $f(X)$ .
3. Check whether the Galois group of  $f(X)$  satisfies certain properties.

Given a polynomial  $f(X)$  over  $\mathbb{Q}$ , in Chapter 7 we give polynomial time algorithms for (1) checking whether the Galois group of  $f(X)$  is nilpotent and (2) checking whether the Galois group of  $f(X)$  is in  $\Gamma_d$ . Chapter 8 deals with computing the order of the Galois group of a polynomial. We prove certain upper bounds assuming the generalised Riemann hypothesis. Finally in Chapter 9 we give some algorithms for computing the Galois group of certain special polynomials.

For a polynomial  $f(X)$ , the Galois group  $G$  can be seen as a permutation group on the set of roots of  $f(X)$ . Combinatorial structures like orbits and blocks associated with the Galois group  $G$  play an important role in our results. The Galois correspondence between blocks and subgroups on one hand (Theorem 3.11) and subgroups and subfields on the other hand (Theorem 6.1) gives us a Galois correspondence (Theorem 7.1) between subfields, subgroups and blocks. This interplay between fields and permutation group

theoretic structures is crucial for our upper bounds. Apart from the permutation group theory we require, for our conditional results of Chapter 8 and 9, an effective version of the Chebotarev density theorem proved assuming the generalised Riemann hypothesis (Section 8.1).

In this chapter we give a brief description of the Galois theory and algebraic number theory required for our results in Chapters 7, 8 and 9. In Section 6.1 we describe the required Galois theory and in Section 6.3 some algebraic number theory. In Section 6.4 we explain some fundamental algorithmic results that we require in this thesis. To measure the complexity of various algorithms we need a precise formulation of sizes of various algebraic entities. This is explained in Section 6.4. Finally, in Section 6.5, we prove some bounds that will be needed in analysing the complexity of various algorithms in this thesis.

## 6.1 Galois theory

We recall some basic facts from Galois theory required for this thesis. A detailed account is available in any standard text book on Galois theory or Algebra for example Lang [40, Chapter VI]. By  $\mathbb{Q}$ ,  $\mathbb{R}$  and  $\mathbb{C}$  we mean the field of rational, real and complex numbers respectively. The ring of integers will be denoted by  $\mathbb{Z}$ . For primes  $p$ ,  $\mathbb{F}_{p^r}$  denotes the unique finite field of  $p^r$  elements.

Let  $K$  be a field. A field  $L$  is said to be a *field extension* of  $K$ , denoted by  $L/K$ , if  $L \supseteq K$ . For a field extension  $L/K$ ,  $L$  is a vector space over  $K$  and its dimension, denoted by  $[L : K]$ , is the *degree* of  $L/K$ . An extension  $L/K$  is *finite* if its degree  $[L : K]$  is finite. If  $L/M$  and  $M/K$  are finite extensions then  $[L : K] = [L : M].[M : K]$ .

Let  $K$  be any field. By  $K[X]$  we mean the ring of polynomials in  $X$  with coefficients from  $K$ . The ring  $K[X]$  is a *unique factorisation domain*. A polynomial  $f(X) \in K[X]$  is *irreducible* if it has no nontrivial factor.

For a field  $K$  the smallest positive integer  $n$  such that  $n.1 = 0$ , if it exists, is called the characteristic of  $K$ . If no such integer exists then we say that  $K$  is of characteristic 0. For example the fields  $\mathbb{Q}$ ,  $\mathbb{R}$  and  $\mathbb{C}$  are of characteristic 0 where as the field  $\mathbb{F}_{p^r}$  is of characteristic  $p$ . For any field  $K$ , the characteristic is either 0 or a prime  $p$ . If  $L/K$  is an extension then the characteristic of  $L$  is same as the characteristic of  $K$ .

Let  $L/K$  be an extension. Then  $\alpha \in L$  is *algebraic* over  $K$  if there is an  $f(X) \in K[X]$  such that  $f(\alpha) = 0$ . For  $\alpha$  algebraic over  $K$ , the *minimal polynomial* of  $\alpha$  over  $K$  is the unique monic polynomial  $\mu_\alpha[K](X)$  of least

degree in  $K[X]$  for which  $\alpha$  is a root. When  $K$  is clear from the context, we simply write  $\mu_\alpha(X)$  instead of  $\mu_\alpha[K](X)$ . Elements  $\alpha, \beta \in L$  are *conjugates* over  $K$  if they have the same minimal polynomial over  $K$ .

Let  $L/K$  be an extension and let  $\alpha \in L$  then  $K(\alpha)$  is the smallest subfield of  $L$  containing  $K$  and  $\alpha$ . If  $\alpha$  is algebraic over  $K$  and if  $\mu_\alpha(X)$  is the minimal polynomial of  $\alpha$  over  $K$  then  $K(\alpha)$  is isomorphic to  $K[X]/\mu_\alpha(X)$ , the ring of polynomials over  $K$  modulo  $\mu_\alpha(x)$ . If  $L/K$  is a finite extension then by the primitive element theorem [40, Theorem 4.6, Chapter V] there is an  $\alpha \in L$  such that  $L = K(\alpha)$ . Such an element  $\alpha$  is called a *primitive element* of  $L$ . A *primitive polynomial* of an extension  $L/K$  is the minimal polynomial of some primitive element of  $L/K$ . Thus if  $T(X)$  is a primitive polynomial of  $L/K$  then the field  $L$  is isomorphic to  $K[X]/T(X)$ .

The *splitting field*  $K_f$  of  $f \in K[X]$  is the smallest extension of  $K$  containing all the roots of  $f$ . An extension  $L/K$  is *normal* if for all irreducible polynomials  $f(X) \in K[X]$ , either  $f(X)$  splits completely into linear factors or has no root in  $L$ . Any finite normal extension over  $K$  is the splitting field of a polynomial in  $K[X]$ . Let  $L/K$  be any extension. By *normal closure* of  $L$  over  $K$  we mean the smallest normal extension of  $K$  that contains  $L$ . For a finite extension  $L/K$ , let  $T(X)$  be any primitive polynomial. The normal closure of  $L$  over  $K$  is the splitting field over  $K$  of  $T(X)$ .

An extension  $L/K$  is *separable* if for all irreducible polynomials  $f(X) \in K[X]$  there are no multiple roots in  $L$ . In particular all characteristic 0 fields are separable and so are all finite fields. A normal and separable extension  $L/K$  is called a *Galois extension*.

A field  $K$  is *algebraically closed* if every polynomial in  $K[X]$  splits over  $K$ . For example the field of complex numbers  $\mathbb{C}$  is algebraically closed. Let  $K$  be any field. The *algebraic closure* of  $K$ , denoted by  $\overline{K}$ , is the smallest field containing  $K$  that is algebraically closed. For every field there is a unique algebraic closure up to isomorphism.

An *automorphism* of a field  $L$  is a field isomorphism  $\sigma : L \rightarrow L$ . The *Galois group*  $\text{Gal}(L/K)$  of a field extension  $L/K$  is the subgroup of automorphisms of  $L$  that leaves  $K$  fixed, i.e. for all  $\alpha \in K$ ,  $\sigma(\alpha) = \alpha$ . The Galois group of a polynomial  $f \in K[X]$  is  $\text{Gal}(K_f/K)$ . Let  $f(X)$  be a polynomial over  $K$  of degree  $n$ . If  $\alpha$  is a root of  $f(X)$  and  $\sigma \in \text{Gal}(K_f/K)$  then  $\sigma(\alpha)$  is also a root of  $f(X)$ . Each  $\sigma \in \text{Gal}(K_f/K)$  is thus completely determined by  $\sigma(\alpha_i)$ ,  $1 \leq i \leq n$ , where the  $\alpha_1, \dots, \alpha_n$  are the roots of  $f$ . Thus the Galois group of a polynomial  $f(X)$  can be seen as a permutation group on the set of roots of  $f(X)$  and hence has order at most  $n!$ .

For a subgroup  $G$  of automorphisms of  $L$ , the *fixed field*  $\text{Fix}(L, G)$  is the largest subfield  $K$  of  $L$  such that every element of  $G$  restricted to  $K$

gives the identity automorphism. We now state the fundamental theorem of Galois theory [40, Theorem 1.1, Chapter VI] which, given a finite Galois extension  $L/K$  with Galois group  $G$ , gives a *Galois correspondence* between subgroups of  $G$  and subfields of  $L$  containing  $K$ .

**Theorem 6.1.** *Let  $L/K$  be a Galois extension with Galois group  $G$ . There is a one-to-one correspondence between subfields  $E$  of  $L$  containing  $K$  and subgroups  $H$  of  $G$ , given by  $E \rightleftharpoons \text{Fix}(L, H)$ . The Galois group  $\text{Gal}(L/E)$  is  $H$  and  $E/K$  is a Galois extension if and only if  $H$  is a normal subgroup of  $G$ . If  $H$  is a normal subgroup of  $G$  and  $E = \text{Fix}(L, H)$  then  $\text{Gal}(E/K)$  is the quotient group  $G/H$ .*

## 6.2 Finite Fields

A finite field is a field of finite cardinality. An example for a finite field is  $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ , the field of integers modulo a prime  $p$  with addition and multiplication defined modulo  $p$ . For any prime  $p$  and an integer  $r$  there is a unique field of cardinality  $p^r$  which we denote by  $\mathbb{F}_{p^r}$ . We have  $[\mathbb{F}_{p^r} : \mathbb{F}_p] = r$  and  $\mathbb{F}_{p^r}$  is the splitting field of  $f(X)$  for any irreducible polynomial  $f(X) \in \mathbb{F}_p[X]$  of degree  $r$ . For integers  $n$  and  $r$ ,  $\mathbb{F}_{p^n}$  is an extension of  $\mathbb{F}_{p^r}$  if and only if  $r$  divides  $n$  in which case the degree  $[\mathbb{F}_{p^n} : \mathbb{F}_{p^r}]$  is given by  $\frac{n}{r}$ . Also in this case  $\mathbb{F}_{p^n}/\mathbb{F}_{p^r}$  is a Galois extension.

Consider the algebraic closure  $\overline{\mathbb{F}}_p$  of  $\mathbb{F}_p$ . The map  $\sigma : a \mapsto a^p$  is an automorphism of  $\overline{\mathbb{F}}_p$  that is identity on  $\mathbb{F}_p$ . The automorphism  $\sigma$  is called the *Frobenius* automorphism. The Galois group  $\text{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_{p^r})$  is a cyclic group of order  $\frac{n}{r}$  and is generated by  $\sigma^r$ . In terms of the Frobenius we can give a different characterisation of the field  $\mathbb{F}_{p^r}$ . The field  $\mathbb{F}_{p^r}$  is the fixed field of  $\overline{\mathbb{F}}_p$  under the group of automorphisms generated by  $\sigma^r$ . Equivalently,  $\mathbb{F}_{p^r}$  is the set of roots of the polynomial  $X^{p^r} - X$  in  $\overline{\mathbb{F}}_p$ .

Let  $f(X)$  be any polynomial in  $\mathbb{F}_q$ ,  $q$  a power of prime  $p$ . Let  $f(X)$  factorise as  $f_1 \dots f_r$  over  $\mathbb{F}_q$  and let  $d_i$  denote the degrees of  $f_i$ . The splitting field of  $f$  is  $\mathbb{F}_{q^m}$  where  $m$  is the least common multiple of the integers  $d_1, \dots, d_r$ .

## 6.3 Algebraic numbers and number fields

We now recall some algebraic number theory. A detailed presentation is available in any standard textbook on algebraic number theory like for example the one due to Neukirch [55].

*Algebraic numbers* are roots of polynomials over  $\mathbb{Q}$  and *algebraic integers* are roots of monic polynomials in  $\mathbb{Z}[X]$ . The set of rational algebraic integers, i.e. algebraic integers in  $\mathbb{Q}$ , is exactly  $\mathbb{Z}$ . For an algebraic number  $\alpha$  there is an integer  $m \in \mathbb{Z}$  such that  $m\alpha$  is an algebraic integer. A *number field* is a finite extension of  $\mathbb{Q}$ .

Let  $\alpha$  be an algebraic number and let  $K$  be the number field  $\mathbb{Q}(\alpha)$ . Since  $\mathbb{C}$  is algebraically closed and contains  $\mathbb{Q}$ ,  $K$  can be seen as a subfield of  $\mathbb{C}$ , i.e. there is an isomorphism from  $K$  to a subfield of  $\mathbb{C}$ . Such an isomorphism from  $K$  to  $\mathbb{C}$  is called an embedding of  $K$ . If  $K$  is of degree  $n$  then there exists  $n$  distinct embeddings of  $K$  into  $\mathbb{C}$ . An embedding  $\sigma$  of  $K$  is a *real embedding* if the image of  $K$  under  $\sigma$  is contained in  $\mathbb{R}$ , otherwise it is a *complex embedding*. The *height* of  $\alpha$ , denoted by  $H(\alpha)$ , is  $\max\{|\sigma(\alpha)|^{c_\sigma}\}$ , where  $\sigma$  varies over all embeddings of  $K$  and  $c_\sigma$  is either 1 or 2 depending on whether  $\sigma$  is a real or complex embedding. Let  $\mu_\alpha(X) \in \mathbb{Q}[X]$  be the minimal polynomial of  $\alpha$  then  $H(\alpha)$  is  $\max\{|\eta|^{c_\eta}\}$  where  $\eta$  runs over all roots of  $\mu_\alpha$  in  $\mathbb{C}$  and  $c_\eta$  is 1 if  $\eta$  is a real root and 2 otherwise. If  $\alpha'$  is a conjugate of  $\alpha$  then  $H(\alpha) = H(\alpha')$ . The height of an algebraic number is a measure of its size. Using Cauchy-Schwartz the following inequalities can be derived.

**Lemma 6.2.** *For any two algebraic numbers  $\alpha$  and  $\beta$ :*

1.  $H(\alpha + \beta) \leq H(\alpha) + H(\beta)$ .
2.  $H(\alpha\beta) \leq H(\alpha)H(\beta)$ .

### 6.3.1 Ring of Algebraic Integers

Let  $K$  be a number field of degree  $n$  and let  $\mathbb{O}_K$  denote the *ring of algebraic integers* of  $K$ . There exist  $\omega_1, \dots, \omega_n \in \mathbb{O}_K$  such that  $\mathbb{O}_K = \mathbb{Z}\omega_1 + \dots + \mathbb{Z}\omega_n$ . Such a set of elements in  $\mathbb{O}_K$  is a *basis* for  $\mathbb{O}_K$ . If  $\omega_1, \dots, \omega_n$  is a basis for  $\mathbb{O}_K$  then  $K = \mathbb{Q}\omega_1 + \dots + \mathbb{Q}\omega_n$ , i.e. the set  $\{\omega_1, \dots, \omega_n\}$  is a basis of  $K$  as a vector space over  $\mathbb{Q}$ . For two bases  $\theta_1, \dots, \theta_n$  and  $\omega_1, \dots, \omega_n$  of  $\mathbb{O}_K$  there is a unimodular matrix  $A = (a_{ij})$  such that  $\omega_i = \sum a_{ij}\theta_j$  for all  $1 \leq i \leq n$ .

Let  $K$  be a number field of degree  $n$ . Recall that  $K$  has  $n$  distinct embeddings  $\sigma_1, \dots, \sigma_n$  into  $\mathbb{C}$ . Let  $\omega_1, \dots, \omega_n$  be a basis for  $\mathbb{O}_K$ . Then the *discriminant*  $d_K$  of  $K$  is the positive integer  $|\det(\sigma_j(\omega_i))|^2$ . The discriminant is independent of the basis chosen for  $\mathbb{O}_K$ .

An ideal  $\mathfrak{a}$  of  $\mathbb{O}_K$  is an additive subgroup of  $\mathbb{O}_K$  such that for every  $\alpha \in \mathbb{O}_K$  and  $\beta \in \mathfrak{a}$   $\alpha\beta \in \mathfrak{a}$ . Let  $\mathfrak{a}$  be an ideal of  $\mathbb{O}_K$ . For an algebraic integer  $\alpha \in \mathbb{O}_K$ , the set  $\alpha\mathbb{O}_K = \{\alpha\beta | \beta \in \mathbb{O}_K\}$  is an ideal. Such ideals are called *principal ideals*. Often we will denote the principal ideal  $\alpha\mathbb{O}_K$  as  $\alpha$ .

A principal ideal domain is a ring where all ideals are principal. An example for a principal ideal domain is  $\mathbb{Z}$ .

We define the sum and product of ideals of  $\mathbb{O}_K$ . For ideals  $\mathfrak{a}$  and  $\mathfrak{b}$  of  $\mathbb{O}_K$ , by  $\mathfrak{a} + \mathfrak{b}$  we mean  $\{\alpha + \beta : \alpha \in \mathfrak{a}, \beta \in \mathfrak{b}\}$ . Similarly by  $\mathfrak{a}\mathfrak{b}$  we mean  $\{\sum_i \alpha_i \beta_i : \alpha_i \in \mathfrak{a}, \beta_i \in \mathfrak{b}\}$ . Furthermore  $\mathfrak{a} + \mathfrak{b}$  is the smallest ideal that contains  $\mathfrak{a}$  and  $\mathfrak{b}$  and  $\mathfrak{a}\mathfrak{b}$  is the ideal  $\mathfrak{a} \cap \mathfrak{b}$ .

We say that  $\mathfrak{a}$  divides  $\mathfrak{b}$ , denoted by  $\mathfrak{a} \mid \mathfrak{b}$ , if  $\mathfrak{a} \supseteq \mathfrak{b}$ . Unlike  $\mathbb{Z}$ , for number fields  $K$ ,  $\mathbb{O}_K$  need not be a unique factorisation domain (for example in the ring  $\mathbb{Z}[\sqrt{-5}]$ , 21 has two factorisations [55, Chapter I, §3]). However ideals of  $\mathbb{O}_K$  have the unique factorisation property, i.e. any ideal  $\mathfrak{a}$  has a unique factorisation into prime ideals as  $\mathfrak{a} = \mathfrak{p}_1^{a_1} \dots \mathfrak{p}_r^{a_r}$ , where  $a_i$  is the highest power  $k$  such that  $\mathfrak{p}_i^k$  divides  $\mathfrak{a}$  ( $\mathbb{O}_K$  is a *Dedekind domain*).

For any ideal  $\mathfrak{a}$ , the ring  $\mathbb{O}_K/\mathfrak{a}$  is a finite ring. The *norm* of  $\mathfrak{a}$ , denoted by  $N(\mathfrak{a})$ , is the number of elements in  $\mathbb{O}_K/\mathfrak{a}$ . Consider a number field  $K$  of degree  $n$ . Let  $\sigma_1, \dots, \sigma_n$  be the  $n$  distinct embeddings of  $K$  into  $\mathbb{C}$ . For any  $\alpha \in \mathbb{O}_K$ , the norm of the principal ideal  $\alpha\mathbb{O}_K$ , which we denote by  $N(\alpha)$ , is equal to the product  $\prod_i \sigma_i(\alpha)$ .

Let  $p \in \mathbb{Z}$  be any prime. For a number field  $K$ , the principal ideal  $p\mathbb{O}_K$ , which we denote by  $p$ , need not be a prime ideal. Knowing how the principal ideal  $p$  factorise is important and Kummer-Dedekind theorem is algorithmically useful for this purpose (see [22, Theorem 4.8.13] for a proof).

**Theorem 6.3** (Kummer-Dedekind). *Let  $K = \mathbb{Q}(\theta)$ , where  $\theta$  is an algebraic integer with minimal polynomial  $T(X) \in \mathbb{Z}[X]$ . Let  $p \in \mathbb{Z}$  be a prime that does not divide the index  $[\mathbb{O}_K : \mathbb{Z}[\theta]]$ . Suppose  $T = T_1^{e_1} \dots T_k^{e_k} \pmod{p}$  is the factorisation of  $T$  over  $\mathbb{F}_p$  into its irreducible factors. Then  $p\mathbb{O}_K$  factors into prime ideals as  $p\mathbb{O}_K = \mathfrak{p}_1^{e_1} \dots \mathfrak{p}_k^{e_k}$ . Moreover the prime ideals  $\mathfrak{p}_i$  are given by  $\mathfrak{p}_i = p\mathbb{O}_K + T_i(\theta)\mathbb{O}_K$  and  $\mathbb{O}_K/\mathfrak{p}_i \cong \mathbb{Z}[\theta]/(p, T_i(\theta))$ .*

## 6.4 Basic algorithms

In this section we give an overview of the algorithmic results required for this thesis. For a detailed presentation of various algorithmic aspects of algebraic number theory we refer the reader to the textbook of Cohen [22]. The algorithms we describe take various algebraic entities like algebraic numbers and number fields as inputs. We need to encode these algebraic entities over a finite alphabet  $\Sigma$  typically  $\{0, 1\}$ . The complexity of these algorithms are measured in terms of the size of these encodings. Our first goal is to make this precise.

### 6.4.1 Encoding algebraic entities

For integers  $c$  we use the standard binary encoding. The size of an integer  $c$  is therefore  $\lceil \lg c \rceil$ . A rational number  $r$  is given by a pair of coprime integers  $(a, b)$  such that  $r = \frac{a}{b}$ . Thus,  $\text{size}(r) = \text{size}(a) + \text{size}(b)$ . Elements of the finite field  $\mathbb{F}_p$ , for prime  $p$ , will be represented as integers in between 0 and  $p$ . Hence an element of  $\mathbb{F}_p$  is of size  $\lg p$ .

The fields that we encounter in this thesis are either finite fields or number fields. Recall that any field  $K$  is a vector space over the associated base field which is either  $\mathbb{Q}$ , if the characteristic is 0, or  $\mathbb{F}_p$ , if the characteristic is  $p$ . We follow the approach of Lenstra [43, 44] for encoding fields. Here we describe how number fields are presented. A similar approach can be taken for finite fields for which we refer to the article of Lenstra [43].

There are two algorithmically equivalent ways of presenting a number field  $K$ , (1) by explicit data and (2) by presenting a primitive polynomial for  $K$ . Let  $K$  be a number field of degree  $n$ . By *explicit data* we mean a linearly independent basis  $e_1, \dots, e_n$  for  $K$  as a vector space over  $\mathbb{Q}$  together with  $n^3$  rationals  $\{c_{ijk}\}_{1 \leq i, j, k \leq n}$  such that  $e_i e_j = \sum_k c_{ijk} e_k$ . In addition by multiplying each  $e_i$ 's by suitable rational integers we assume, with out loss of generality, that  $e_i$ 's are algebraic integers. Thus the field can be presented by giving the list  $\{c_{ijk}\}_{1 \leq i, j, k \leq n}$  and by size of  $K$  we mean  $\sum \text{size}(c_{ijk})$ .

Any  $\alpha \in K$  can be expressed uniquely as a summation  $\alpha = \sum a_i e_i$ ,  $a_i \in \mathbb{Q}$ . By  $\text{size}(\alpha)$  we mean  $\sum \text{size}(a_i)$ . A polynomial of degree  $d$  over  $K$  is presented by giving the ordered list of all its  $d$  coefficients and hence for  $f(X) = a_0 + \dots + a_d X^d$  in  $K[X]$  by  $\text{size}(f)$  we mean  $\sum \text{size}(a_i)$ .

Recall that  $K = \mathbb{Q}(X)/\mu(X)$  for some primitive polynomial  $\mu(X)$  over  $\mathbb{Q}$ . Thus a number field  $K$  can be presented by giving a primitive polynomial  $\mu(X) \in \mathbb{Q}[X]$ . If  $\mu(X) = c_0 + c_1 X + \dots + c_{n-1} X^{n-1} + X^n$  then by  $\text{size}(K)$  we mean  $\sum \text{size}(c_i)$ . As before we can ensure that  $\mu(X)$  is a monic polynomial with coefficients from  $\mathbb{Z}$ .

A primitive polynomial  $\mu(X)$  for  $K$  directly gives explicit data for  $K$ : choose  $e_i$  to be  $X^{i-1} \pmod{\mu(X)}$ . Conversely we show that given explicit data, one can compute a primitive polynomial. We first prove the following lemma.

**Lemma 6.4.** *Let  $K$  be a number field of degree  $n$  presented via explicit data  $\{c_{ijk}\}_{1 \leq i, j, k \leq n}$ . Let  $e_1, \dots, e_n$  denote the corresponding basis for  $K$ . Given an algebraic number  $\alpha = \sum_{i=1}^n a_i e_i$  there is an algorithm to compute the minimal polynomial of  $\alpha$  that runs in time bounded by a polynomial in  $\text{size}(K)$  and  $\sum \text{size}(a_i)$*

*Proof.* Recall that  $K$  is a vector space over  $\mathbb{Q}$  with a basis  $\{e_i\}_{i=1}^n$  and any algebraic number  $\alpha$  in  $K$  is a vector  $\sum a_i e_i$ . The degree  $d$  of  $\alpha$  is the largest  $i$  such that the set  $\{1, \dots, \alpha^{i-1}\}$  is a linearly independent set of vectors. It follows that  $-\alpha^d$  can be written as a linear combination  $-\alpha^d = c_0 + \dots + c_{d-1} \alpha^{d-1}$ . Using the explicit data we can compute the vectors  $\alpha^i = \sum_j a_{ij} e_j$  in time polynomial in  $\text{size}(K)$  and  $\sum \text{size}(a_i)$ . Furthermore in polynomial time we can compute  $d$  and the rationals  $\{c_i : 0 \leq i < d\}$  as it involves solving linear equations over  $\mathbb{Q}$ . The minimal polynomial of  $\alpha$  is therefore  $c_0 + c_1 X + \dots + c_{d-1} X^{d-1} + X^d$ .  $\square$

We require the following effective version of primitive element theorem (see Section 6.10 of van der Waerden's book [71]).

**Lemma 6.5.** *Let  $\alpha$  and  $\beta$  be algebraic numbers of degrees  $m$  and  $n$  respectively. Let  $\{\alpha_i\}_{i=1}^m$  and  $\{\beta_j\}_{j=1}^n$  be their  $\mathbb{Q}$ -conjugates. Let  $c$  be any integer such that  $\alpha_i + c\beta_j \neq \alpha_r + c\beta_s$  for all  $(i, j) \neq (r, s)$ . Then  $\alpha + c\beta$  is a primitive element of  $\mathbb{Q}(\alpha, \beta)$ .*

Consider the set  $A = \{\frac{\alpha_i - \alpha_r}{\beta_s - \beta_j} \mid s \neq j\}$  of  $\binom{m}{2} \cdot \binom{n}{2} + 1$  algebraic numbers. It follows from Lemma 6.5 that if  $c \notin A$  then  $\alpha + c\beta$  is a primitive element of  $\mathbb{Q}(\alpha, \beta)$ . Therefore there exists an integer  $c$ ,  $1 \leq c \leq m^2 n^2 + 1$  such that  $\alpha + c\beta$  is primitive. We summarise this in the following proposition.

**Proposition 6.6.** *Let  $\alpha$  and  $\beta$  be algebraic numbers of degree  $m$  and  $n$  respectively. There exists an integer  $c \in \{1, \dots, m^2 n^2 + 1\}$  such that  $\alpha + c\beta$  is a primitive element of  $\mathbb{Q}(\alpha, \beta)$ .*

Computing the primitive polynomial for  $K$  is now straight forward. Let  $K$  be presented via explicit data  $\{c_{ijk}\}_{1 \leq i, j, k \leq n}$  and let  $e_1, \dots, e_n$  be the corresponding basis. We compute constants  $1 \leq c_i \leq n^4 + 1$ ,  $1 \leq i \leq n$ , such that  $\sum_{i=1}^n c_i e_i$  is a primitive element for  $K$ .

Let  $K_r$  denote the field  $\mathbb{Q}(e_1, \dots, e_r)$ . We compute the primitive element  $\gamma_i$  of  $K_i$  inductively. To begin with  $\gamma_1 = e_1$ . Assume that we have computed  $\gamma_{i-1}$ . We choose an integer  $c_i$  from the set  $\{1, \dots, n^4 + 1\}$  such that the minimal polynomial of  $\gamma_{i-1} + c_i e_i$  is of maximal degree. The algebraic number  $\gamma_i = \gamma_{i-1} + c_i e_i$  is a primitive element for  $K_i$ . Having computed  $\gamma_n$  we can compute the minimal polynomial for  $\gamma_n$  using Lemma 6.4. This gives a primitive polynomial for  $K$ .

We have thus proved that presenting number fields via explicit data or via a primitive polynomial are polynomial time equivalent. The size of  $K$  in each of these presentation might differ but only up to a polynomial factor. Thus we can assume without loss of generality either of the two presentation.

### 6.4.2 Factoring polynomials and related problems

Recall that for a field  $K$ , the ring  $K[X]$  is a unique factorisation domain. Polynomials  $f(X)$  over  $\mathbb{Q}$  can be factored into irreducible factors in polynomial time using the celebrated Lenstra-Lenstra-Lovász [42] algorithm. A key step in this algorithm is lattice basis reduction. A. K. Lenstra [41] generalised this basis reduction to give a polynomial time algorithm for factoring polynomials over number fields. Using norms of polynomials, Landau [38] gave a polynomial time reduction from factoring over  $K$  to factoring over  $\mathbb{Q}$ . We summarise these results in the following theorem.

**Theorem 6.7.** *Given a number field  $K$  and a polynomial  $f(X)$  in  $K[X]$  there is an algorithm that computes the irreducible factors of  $f(X)$  in time bounded by a polynomial in  $\text{size}(f)$  and  $\text{size}(K)$ .*

Let  $K$  be a finite field of characteristic  $p$ . Berlekamp [16] gave a deterministic polynomial time algorithm for factoring polynomials over  $K$  for small primes  $p$ . However for large characteristic only randomised algorithms are known. Given a polynomial  $f(X) \in \mathbb{F}_q$ , there are randomised algorithms that run in time polynomial in  $\text{size}(f)$  and  $\lg q$  [17, 21] for factoring  $f(X)$ . We summarise these results in the following theorem.

**Theorem 6.8.** *Given a polynomial  $f(X)$  over the finite field  $K$  of characteristic  $p$  there is a deterministic algorithm that runs in time polynomial in  $\text{size}(f)$  and  $p$  for factoring  $f(X)$ . Given a polynomial  $f(X) \in \mathbb{F}_q$  there is a randomised algorithm that runs in time polynomial in  $\text{size}(f)$  and  $\lg q$  to factor  $f(X)$ .*

Even though factoring polynomials over finite fields do not have efficient deterministic algorithms, there are efficient deterministic irreducibility tests. More generally given a polynomial  $f(X) \in \mathbb{F}_q[X]$  and an integer  $d$  there is a polynomial time deterministic algorithm to compute the product of all irreducible factors of  $f(X)$  of degree  $d$  (see Section 14.2 of [73]). In particular, for a give number  $d$  one can compute the number of irreducible factor of  $f(X)$  of degree  $d$  and thus we have an efficient irreducibility test. We summarise this result in the following theorem.

**Theorem 6.9.** *Given a polynomial  $f(X) \in \mathbb{F}_q[X]$  of degree  $n$  and an integer  $d \leq n$ , there is a deterministic algorithm that runs in time polynomial in  $\text{size}(f)$  and  $\lg q$  that computes the product of all the irreducible factors of  $f(X)$  of degree  $d$ . In particular there is a deterministic algorithm that runs in time polynomial in  $\text{size}(f)$  and  $\lg q$  that computes for each  $d$  the number of irreducible factors of  $f(X)$  of degree  $d$ .*

### 6.4.3 Algorithms for Galois group computation

Let  $L/K$  be a field extension. Recall that the Galois group  $\text{Gal}(L/K)$  is the group of automorphisms of  $L$  that are identity when restricted to  $K$ . In this section we describe known algorithms for computing the Galois group of a polynomial and related problems.

Firstly from a computational point of view if  $K$  is a finite field then the problem is trivial as the Galois group is generated by an appropriate power of the Frobenius (see Section 6.2). Let  $q = p^r$ . Given a polynomial  $f(X)$  over a finite field  $\mathbb{F}_q$  recall that if  $d_1, \dots, d_k$  are the set of degrees of irreducible factors of  $f(X)$  then the Galois group of  $f(X)$  is a cyclic group of order  $m$  where  $m$  is the least common multiple of  $d_1, \dots, d_k$  and is generated by the  $r$ th power of the Frobenius. By Theorem 6.9 we can compute the degrees  $d_1, \dots, d_k$  in time polynomial in size  $(f)$  and  $\lg q$ .

For polynomials  $f(X)$  over number fields the best known algorithm for computing the Galois group is due to Landau [37]. Given a polynomial  $f(X) \in K[X]$  Landau's algorithm computes the Galois group  $\text{Gal}(K_f/K)$  in time polynomial in size  $(f)$  and  $[K_f : K]$ . Since  $[K_f : K]$  could be as large as  $n!$ , this is an exponential time algorithm. We give a brief sketch of this algorithm.

Let  $K$  be any number field and let  $L/K$  be an extension, not necessarily Galois. There exists a primitive element  $\alpha$  such that  $L = K(\alpha)$ . Let  $\mu(X) \in K[X]$  be the minimal polynomial of  $\alpha$  over  $K$ . Using Theorem 6.7 we first factorise  $\mu(X)$  over  $L$ . Any root of  $\mu(X)$  can be expressed as a polynomial in  $\alpha$ . Let  $P_1(\alpha), \dots, P_r(\alpha)$ ,  $P_i(X) \in K[X]$ , be the distinct roots of  $\mu(X)$  over  $L$  then the Galois group  $\text{Gal}(L/K)$  consists of exactly  $r$  elements given by the linear maps  $\alpha \mapsto P_i(\alpha)$ . Thus the Galois group  $L/K$  can be computed in time polynomial in size  $(L)$ . Thus the problem of computing the Galois group amounts to computing a primitive polynomial for the splitting field of  $f(X)$ .

Given a polynomial  $f(X) \in K[X]$  of degree  $n$ . Let  $\alpha_1, \dots, \alpha_n$  be the roots of  $f(X)$ . Using Theorem 6.7 we compute the fields  $L_i = K(\alpha_1, \dots, \alpha_i)$  inductively. Start with  $L_0 = K$ . Assume that we have already computed the explicit data for  $L_i$ . If  $f(X)$  splits completely over  $L_i$  we stop otherwise let  $g(X)$  be an irreducible factor of  $f(X)$  of degree greater than 1 over  $L_i$  which we obtain using Theorem 6.7. Then the field  $L_{i+1} = L_i[X]/g(X)$  contains at least one more root of  $f(X)$  than  $L_i$  and the explicit data for  $L_{i+1}$  can be computed. Having computed the splitting field  $L$  of  $f$  we can compute the Galois group  $\text{Gal}(L/K)$  as described above. We summarise the above discussion in the following theorem.

**Theorem 6.10** (Landau). *Given a polynomial  $f(X)$  over a number field  $K$  and a positive integer  $N$ . There is a deterministic algorithm running in time bounded by a polynomial in  $N$  and  $\text{size}(f)$  that checks whether the splitting field of  $f$  is of degree less than  $N$  and if yes computes the entire multiplication table for  $\text{Gal}(K_f/K)$ . In particular given a number fields  $L/K$  and an integer  $N$ , there is an algorithm running in time polynomial in  $\text{size}(L)$ ,  $\text{size}(K)$  and  $N$  that decides whether the normal closure  $\tilde{L}$  of  $L$  is of degree less than  $N$  over  $K$  and if yes computes the explicit data for  $\tilde{L}$ .*

The algorithm of Theorem 6.10 outputs the entire multiplication table of the Galois group of  $f(X)$ . There is a much more succinct way of presenting the Galois group. Recall that the Galois group of  $f(X)$  is completely specified by its action on the roots of  $f(X)$ . If  $n$  is the degree of  $f(X)$ , by suitably naming the roots of  $f(X)$  the Galois group of  $f(X)$  can be seen as a subgroup of  $S_n$ . In other words there is a faithful representation of  $\text{Gal}(K_f/K)$  as a permutation group over a cardinality  $n$  set  $\Omega$ . Recall that subgroups of  $S_n$  can be presented succinctly via a generator set. Moreover several natural algorithmic tasks for a permutation group  $G$  given a generator set for it can be accomplished efficiently. For example it is possible to determine if  $G$  is solvable in polynomial time, or to determine a composition series for  $G$  in polynomial time among several other tasks (a survey of important results is available in the article of Luks [49]). Thus determining the Galois group by its action on the roots of  $f$  is a reasonable way of describing the output.

For polynomials  $f(X)$  with small Galois group Theorem 6.10 gives an efficient algorithm for Galois group computation. For example if  $f(X)$  is irreducible and  $f(X)$  splits in  $K[X]/f(X)$  then we have a polynomial time algorithm for computing the Galois group of  $f(X)$ . In particular Theorem 6.10 gives a polynomial time algorithm to compute the Galois group of an irreducible polynomial  $f(X)$  with abelian Galois group. This is because an abelian transitive subgroup of  $S_n$  is of size  $n$ .

**Proposition 6.11.** *Let  $G$  be a transitive abelian permutation group on  $\Omega$  then for any  $\alpha \in \Omega$   $G_\alpha = 1$  and  $\#G = \#\Omega$ .*

*Proof.* Fix any  $\alpha \in \Omega$ . Since  $G$  is transitive for any  $\beta \in \Omega$  there is a  $g_\beta \in G$  such that  $\alpha^{g_\beta} = \beta$ . The group  $G_\beta$  is given by  $g_\beta^{-1}G_\alpha g_\beta$  and since  $G$  is abelian is equal to  $G_\alpha$ . This implies any element that fixes  $\alpha$  fixes all elements  $\beta$  pointwise and hence is identity. Therefore  $G_\alpha = 1$ . By Orbit-Stabiliser formula (Theorem 3.8) we have  $\#G = \#\Omega \cdot \#G_\alpha = \#\Omega$ .  $\square$

Proposition 6.11 and Theorem 6.10 can be used to give a polynomial time algorithm to test whether the Galois group of a polynomial  $f(X) \in K[X]$  is

abelian [37]. Given a polynomial  $f(X)$  we first compute all its irreducible factors over  $K[X]$ . For each irreducible factor  $g(X)$  we compute the Galois group  $\text{Gal}(K_g/K)$  using Theorem 6.10. If  $\text{Gal}(K_g/K)$  is too large, i.e.  $\text{Gal}(K_g/K)$  is of order greater than the degree of  $g(X)$  then clearly it is not abelian (Proposition 6.11). Having computed the Galois group  $\text{Gal}(K_g/K)$  explicitly we can check whether it is abelian. The Galois group of  $f(X)$  is abelian if the Galois groups of each of its irreducible factor is abelian.

**Theorem 6.12** (Landau). *Let  $f(X) \in K[X]$  be any polynomial. There is an algorithm that runs in time polynomial in  $\text{size}(f)$  and  $\text{size}(K)$  that checks whether the Galois group of  $f(X)$  is abelian. If in addition the polynomial  $f(X)$  is irreducible, there is an algorithm that runs in time polynomial in  $\text{size}(f)$  and  $\text{size}(K)$  that computes the Galois group of  $f$ .*

Even though the Galois group of an irreducible polynomial  $f(X) \in \mathbb{Q}[X]$  with abelian Galois group can be computed efficiently, there are no efficient algorithms when  $f(X)$  is reducible. In fact even when  $f(X)$  is a product of quadratic polynomials nothing better than the exponential algorithm is known ([44, Problem 3.4]). In Chapter 9, assuming the generalised Riemann hypothesis, we give a randomised polynomial time algorithm for this problem.

We now consider another important task, computing the fixed field of a field  $L$  given a set of automorphism of  $L$ . Given an automorphism  $\sigma$  of  $L$ , the fixed field of  $L$  under automorphisms generated by  $\sigma$  is the kernel of the map  $\sigma - 1$ . Let  $L$  be any field of degree  $n$  presented via explicit data  $\{c_{ijk}\}_{1 \leq i,j,k \leq n}$ . Let  $e_1, \dots, e_n$  be the corresponding basis for  $L$ . Then any automorphism  $\sigma$  is a linear map on  $L$  as a vector space over  $\mathbb{Q}$ . Having fixed a basis  $e_1, \dots, e_n$ , each  $\sigma$  can be represented by an  $n \times n$  matrix  $A_\sigma$  over  $\mathbb{Q}$ . The subspace of solutions for the linear equation  $A_\sigma \mathbf{x} = \mathbf{x}$  is exactly the fixed field of  $L$  under  $\sigma$ . A basis for this field can be computed in polynomial time and thus we have its explicit data.

To find the fixed field of  $L$  under the automorphisms generated by  $S = \{\sigma_1, \dots, \sigma_r\}$ , consider the fixed field  $L_i$  of  $L$  under the automorphisms generated by  $\{\sigma_1, \dots, \sigma_i\}$ . Starting from  $L_0 = L$  we compute the fields  $L_i$  inductively. Having computed a basis for  $L_{i-1}$ , we compute  $L_i$  by computing a basis of the kernel of  $\sigma_i - 1$  over  $L_{i-1}$  as described above. This inductive algorithm is evidently polynomial time. Thus we have the following theorem.

**Theorem 6.13.** *Given a field  $L$  via explicit data  $\{c_{ijk}\}_{1 \leq i,j,k \leq n}$  and a set  $S$  of automorphisms of  $L$ . There is a deterministic algorithm running in*

time polynomial in  $\#S$  and  $\text{size}(L)$  to compute the fixed field of  $L$  under the group generated by  $S$ .

## 6.5 Some useful bounds

In this section we prove certain bounds that will be useful in analysing the algorithms of this thesis. Given a number field  $K$  of degree  $n$  presented via explicit data. Our first goal is to give a bound on the height of algebraic numbers of  $K$  in terms of its size.

**Lemma 6.14.** *Let  $K$  be a number field of degree  $n$  presented via explicit data  $\{c_{ijk}\}_{1 \leq i,j,k \leq n}$ . Let  $e_1, \dots, e_n$  be the corresponding basis for  $K$ . Then for each  $i$ ,  $H(e_i) \leq n \cdot 2^{\text{size}(K)}$ . It follows that for any  $\alpha$  in  $K$ ,  $H(\alpha) \leq n^2 \cdot 2^{\text{size}(\alpha) + \text{size}(K)}$ .*

*Proof.* Let  $1 \leq i \leq n$  be such that  $H(e_i)$  is maximum. We have  $e_i e_i = \sum_{k=1}^n c_{iik} e_k$ . Each  $c_{iik}$  is less than  $2^{\text{size}(K)}$ . Hence it follows that  $H(e_i)^2 \leq n H(e_i) 2^{\text{size}(K)}$  (Lemma 6.2) and  $H(e_i) \leq n \cdot 2^{\text{size}(K)}$ .

Let  $\alpha = \sum_{k=1}^n a_k e_k$ . Recall that  $\text{size}(\alpha) = \sum \text{size}(a_k)$  and hence  $a_k \leq 2^{\text{size}(\alpha)}$ . Therefore  $H(\alpha) \leq n \cdot 2^{\text{size}(\alpha)} \cdot H(e_i) = n^2 \cdot 2^{\text{size}(\alpha) + \text{size}(K)}$ .  $\square$

Conversely, in many cases we would like to bound the sizes of algebraic numbers given a bound on its height. The following lemma serves this purpose.

**Lemma 6.15.** *Let  $K$  be a number field of degree  $n$  and  $\alpha \in \mathbb{O}_K$ . Then  $\text{size}(\alpha) \leq n^3(\lg H(\alpha) + \text{size}(K))$ .*

*Proof.* Let  $\{c_{ijk}\}_{1 \leq i,j,k \leq n}$  be the explicit data for  $K$  and let  $e_1, \dots, e_n$  be the corresponding basis. Let  $\sigma_1, \dots, \sigma_n$  be the  $n$  distinct embeddings of  $K$  into  $\mathbb{C}$  and let  $e_{ij} = \sigma_i(e_j)$ . Consider the algebraic integer  $\alpha = \sum_j c_j e_j$ . If  $\alpha_i = \sigma_i(\alpha) = \sum_j c_j e_{ij}$  then  $|\alpha_i| \leq H(\alpha)$ .

Consider the system of linear equations  $\sum c_j e_{ij} = \alpha_i$ ,  $1 \leq i \leq n$ . By Cramer's rule, we have  $c_j = \frac{\det(A_j)}{\det(A)}$  where  $A$  is the  $n \times n$  matrix  $(e_{ij})_{1 \leq i,j \leq n}$  and  $A_j$  is the matrix obtained by replacing the  $j^{\text{th}}$  column of  $A$  by  $\alpha_i$ 's. The number  $\det(A)^2$  is a symmetric function on the algebraic integers  $e_{ij}$ 's. It follows that  $\det(A)^2$  is in  $\mathbb{Z}$  and therefore  $|c_j| \leq \det(A_j)$  for each  $i$ .

For an  $n \times n$  matrix  $M$ , the determinant of  $M$  is bounded by  $n^n \lambda^n$  where  $\lambda$  is the largest entry of  $M$ . Therefore  $|\det(A_i)| \leq n^n \cdot \lambda^n$  where  $\lambda =$

$\max(\mathbf{H}(e_i), \mathbf{H}(\alpha))$  ( $\mathbf{H}(e_{ij}) = \mathbf{H}(e_i)$ ). We have thus proved that  $\text{size}(c_i) \leq n \cdot (\lg n + \lg \mathbf{H}(\alpha) + \lg \mathbf{H}(e_i)) \leq n \cdot (\lg \mathbf{H}(\alpha) + \text{size}(K) + 2 \lg n)$ . We then obtain

$$\text{size}(\alpha) = \sum_{i=0}^{n-1} \text{size}(c_i) \leq n^3(\lg \mathbf{H}(\alpha) + \text{size}(K)).$$

□

We now prove a bound on the size of the minimal polynomial of an algebraic integer  $\alpha$  in terms of its height.

**Proposition 6.16.** *Let  $\alpha$  be an algebraic integer of degree  $n$  and let  $\mu_\alpha(X)$  be the minimal polynomial of it over  $\mathbb{Q}$ . Then  $\text{size}(\mu_\alpha(X)) \leq n^2(1 + \lg \mathbf{H}(\alpha))$ .*

*Proof.* Let  $\alpha_1, \dots, \alpha_n$  be the conjugates of  $\alpha$  then the minimal polynomial is give by  $\mu(X) = \sum_{i=0}^n s_i X^i$  where  $s_i$  is the  $i$ th symmetric function over  $\alpha_1, \dots, \alpha_n$ , i.e.  $s_i$  is the sum of all possible products of elements in  $\{\alpha_1, \dots, \alpha_n\}$  taken  $i$  at a time. Since  $\mathbf{H}(\alpha_i) = \mathbf{H}(\alpha)$  we have  $|s_i| \leq \binom{n}{i} \mathbf{H}(\alpha)^i$ . Therefore

$$\text{size}(\mu_\alpha(X)) \leq \sum_i \lg \left[ \binom{n}{i} \mathbf{H}(\alpha)^i \right] \leq n^2(1 + \lg \mathbf{H}(\alpha)).$$

□

Conversely, given and algebraic number  $\alpha$ , we often need a bound on  $\mathbf{H}(\alpha)$  in terms of the size of its minimal polynomial  $\mu_\alpha$  over  $\mathbb{Q}$ . For this purpose we state an inequality due to Landau [36] (a proof of this inequality is available in [73, Theorem 6.31]). Consider a polynomial  $f(X) = \sum a_i X^i \in \mathbb{C}[X]$ . Define  $\|f\|_2$  as  $\sqrt{\sum |a_i|^2}$ . We use the following inequality to bound the sizes of algebraic numbers.

**Lemma 6.17** (Landau). *Let  $f(X) = a_0 + \dots + a_d X^d \in \mathbb{C}[X]$  of degree  $d$ , and let  $\alpha_1, \dots, \alpha_d \in \mathbb{C}$  be its roots. Then,*

$$|a_d| \prod_{i=1}^d \max(1, |\alpha_i|) \leq \|f\|_2.$$

Let  $\alpha$  be an algebraic integer of degree  $n$  and let  $\mu_\alpha(X) = c_0 + c_1 X + \dots + c_n X^{n-1} + X^n$  be its minimal polynomial. For all  $i$ ,  $c_i \leq 2^{\text{size}(\mu_\alpha)}$  and therefore  $\|\mu_\alpha\|_2 \leq \sqrt{n} 2^{\text{size}(\mu_\alpha)}$ . Recall that  $\mathbf{H}(\alpha)$  is the maximum over  $|\eta|^{c_n}$

where  $\eta$  ranges over the roots of  $\mu_\alpha$  and  $c_\eta$  is either 1 or 2 depending on whether  $\eta$  is real or complex. Together with Landau's inequality we thus have the following bound.

**Proposition 6.18.** *Let  $\alpha$  be an algebraic integer with minimal polynomial  $\mu_\alpha$  over  $\mathbb{Q}$ . Then  $H(\alpha) \leq n \cdot 4^{\text{size}(\mu_\alpha)}$ .*

We now prove a bound on the discriminant of a number field. For a polynomial  $T(X) \in \mathbb{Q}[X]$  with roots  $\theta_1, \dots, \theta_n$  by *discriminant*, which we denote by  $d_T$  we mean the product  $\prod_{i \neq j} (\theta_i - \theta_j)$ . For an algebraic number  $\alpha$ , by the discriminant of  $\alpha$ , denoted by  $d_\alpha$ , we mean the discriminant of the minimal polynomial  $\mu_\alpha(X)$  of  $\alpha$  over  $\mathbb{Q}$ . Consider a number field  $K = \mathbb{Q}(\theta)$  where  $\theta$  is an algebraic integer. The discriminant  $d_K$  of  $K$  divides  $d_\theta$  and  $\frac{d_\theta}{d_K} = [\mathbb{O}_K : \mathbb{Z}[\theta]]^2$ . Therefore to bound  $d_K$  it is sufficient to give a bound on the discriminant  $d_\theta$ . We use of this to show the following bound on the discriminant.

**Theorem 6.19.** *Let  $f(X)$  be a monic polynomial over  $\mathbb{Z}$  of degree  $n$  with roots  $\alpha_1, \dots, \alpha_n$ . There exists an algebraic integer  $\theta = \sum c_i \alpha_i$ ,  $c_i \in \mathbb{Z}$  such that the  $\theta$  is a primitive element for the extension  $\mathbb{Q}_f/\mathbb{Q}$  and  $\lg d_\theta \leq (n!)^3 \cdot \text{size}(f)$ . As a result we have  $\lg d_{\mathbb{Q}_f} \leq (n!)^3 \cdot \text{size}(f)$ .*

*Proof.* Let  $N \leq n!$  be the degree of the splitting field  $K = \mathbb{Q}_f$ . Let  $\alpha_1, \dots, \alpha_n$  be the roots of  $f(X)$  then  $K = \mathbb{Q}(\alpha_1, \dots, \alpha_n)$ . By Proposition 6.6 there are integer constant  $1 \leq c_i \leq N^4 + 1$  such that  $\theta = \sum_i c_i \alpha_i$  is a primitive element of  $K$ . Since  $f(X)$  is a monic polynomial over  $\mathbb{Z}$  it follows that  $\alpha_i$ 's are algebraic integers and so is  $\theta$ .

By Landau's inequality we have  $|\alpha_i| \leq \sqrt{n} 2^{\text{size}(f)}$  and therefore we have  $H(\theta) \leq n \cdot N^4 4^{\text{size}(f)}$ . Therefore we have

$$\lg d_K \leq \lg d_\theta \leq N^2(1 + 2 \lg n + 4 \lg N + 2 \text{size}(f)).$$

Since  $N \leq n!$  we have the required bound. □

## 6.6 Discussion

Computing the Galois group of  $f(X)$  is hard both in theory and in practice. No polynomial time algorithm is known. Current computer algebra systems can compute typically the Galois group of polynomials of degree in the range 20 to 25. Apart from their mathematical significance many computational problems that arise in algebraic number theory have wide range of application especially in cryptography. Hence algorithms that run efficiently in

practice are of utmost importance. For a detailed presentation of algorithms from a practical point of view we refer to the book of Cohen [22].

However, in this thesis our focus is not directed on these issues. A polynomial time algorithm will be considered efficient although practical implementations might turn out to be too slow. In this sense our approach is similar to the approach of Lenstra in his survey article [44]. In the absence of efficient algorithms our attempt would be to give nontrivial complexity upper bounds.

As mentioned before, to prove nontrivial complexity theoretic results one often require novel techniques. A striking example is the recent AKS algorithm for primality testing [1]. Although the algorithm runs in polynomial time, as far as testing primality of large numbers in practical contexts like for example in cryptographic application, the randomised tests are still preferred. However the techniques developed could lead to solutions of other interesting questions.

Often complexity theoretic classification of natural problems have been fruitful in understanding the inherent intractability of these problems. For example showing hardness results for a problem say for NP in some sense shows that the problem is computationally hard. Even though computing Galois groups are hard in practice, no hardness results (in the complexity theoretic sense) is yet known. Showing such hardness results could be challenging and probably need considerable mathematical techniques.

Attempts to understand the complexity of natural problems have led to considerable progress in complexity theory as well. For example study of one-way functions led Valiant to define the complexity class UP [69]. Algebraic number theory and Galois theory is a rich source of natural computational problems and studying the complexity of these problems might lead to considerable progress in complexity theory as well. What makes these problems particularly attractive is the availability of powerful mathematical tools. Unfortunately even though the mathematics is fairly well understood virtually nothing is known about the computational complexity of many of the fundamental problems in this area. On one hand even for polynomials with abelian Galois group no efficient algorithms are known unconditionally. On the other hand there is no hardness result known despite the fact that the best known algorithm for computing the Galois group is exponential time.

## Chapter 7

# Testing nilpotence of Galois group

Given a polynomial  $f(X)$  over  $\mathbb{Q}$  in this chapter we study the following two problems (1) checking whether the Galois group is nilpotent and (2) checking whether the Galois group is in  $\Gamma_d$ . As mentioned before knowing certain properties of the Galois group of a polynomial  $f(X)$  gives information about its roots. For example the seminal work of Galois shows that a polynomial  $f(X)$  is solvable by radicals if and only if its Galois group is solvable. However algorithmically this does not give a satisfactory answer as computing the Galois group is hard. Landau and Miller [39] achieved a remarkable breakthrough by giving a polynomial time algorithm for checking solvability.

First, we show that given a polynomial  $f(X) \in \mathbb{Q}[X]$ , we can test whether the Galois group of  $f(X)$  is nilpotent in polynomial time. Recall that a group  $G$  is nilpotent if all its Sylow subgroups are normal. Even though every nilpotent group is solvable, the Landau-Miller solvability test does not give a nilpotence test. This is because knowing the composition factors of a group  $G$  alone is not enough to decide whether  $G$  is nilpotent.

We generalise the Landau-Miller test and show that  $\Gamma_d$ -testing for constant  $d$  is in polynomial time. Recall that a group  $G$  is in  $\Gamma_d$  if there is a composition series  $G = G_0 \triangleright \dots \triangleright G_t = 1$  such that  $G_i/G_{i+1}$  is either abelian or is isomorphic to a subgroup of  $S_d$ .

An important idea used in both these tests is the Galois correspondence between blocks, fields and groups that we now explain. Let  $f(X)$  be an irreducible polynomial in  $\mathbb{Q}[X]$  and let  $G$  be the Galois group of  $f(X)$ . Since  $f(X)$  is irreducible,  $G$  is a transitive permutation group on  $\Omega$ , the set

of roots of  $f(X)$ . For a block  $\Delta$  recall that  $G_\Delta$  is the subgroup of  $G$  that setwise stabilises  $\Delta$ . Let  $\mathbb{Q}_\Delta$  denote the fixed field  $\text{Fix}(\mathbb{Q}_f, G_\Delta)$ .

**Theorem 7.1.** *Let  $f(X) \in \mathbb{Q}[X]$  be an irreducible polynomial with Galois group  $G$ . Let  $\Omega$  be the roots of  $f(X)$  and let  $\alpha \in \Omega$  be any particular root. There is a one-to-one correspondence between  $G$ -blocks containing  $\alpha$ , subgroups of  $G$  containing  $G_\alpha$  and subfields between  $\mathbb{Q}(\alpha)$  and  $\mathbb{Q}$  given by*

$$\Delta \rightleftharpoons G_\Delta \rightleftharpoons \mathbb{Q}_\Delta.$$

*Furthermore if  $\{\alpha\} = \Delta_0 \subseteq \dots \subseteq \Delta_m = \Omega$  is an increasing chain of blocks then  $\mathbb{Q}(\alpha) = \mathbb{Q}_{\Delta_0} \supseteq \dots \supseteq \mathbb{Q}_{\Delta_m} = \mathbb{Q}$  is a decreasing tower of number fields between  $\mathbb{Q}(\alpha)$  and  $\mathbb{Q}$ .*

The first is the Galois correspondence between  $G$ -blocks and subgroups of  $G$  containing  $G_\alpha$  (Theorem 3.11) and the second correspondence is via the fundamental theorem of Galois theory (Theorem 6.1). The crucial observation of Landau and Miller is that even though the Galois group  $G$  is unknown, the field  $\mathbb{Q}_\Delta$  can be computed in polynomial time. Knowing the structure of the fields  $\mathbb{Q}_\Delta$  gives us valuable information about the groups  $G_\Delta$ . Consider a permutation group  $G$  on the set  $\Omega$ . Let  $\Delta \subseteq \Sigma$  be two  $G$ -blocks recall that  $\mathcal{B}(\Sigma/\Delta)$  denotes the set of blocks  $\{\Delta^g | g \in G, \Delta^g \subseteq \Sigma\}$ . The group  $G(\Sigma/\Delta)$  is the subgroup of  $G_\Sigma$  that fixes all the blocks in  $\mathcal{B}(\Sigma/\Delta)$  and  $G^\Delta = G(\Omega/\Delta)$ .

**Proposition 7.2.** *Let  $\Delta \subseteq \Sigma$  be two  $G$  blocks then.*

1. *The normal closure of the field  $\mathbb{Q}_\Delta$  over  $\mathbb{Q}_\Sigma$  is exactly the fixed field  $\text{Fix}(\mathbb{Q}_f, G(\Sigma/\Delta))$ . In particular the normal closure of  $\mathbb{Q}_\Delta$  is the fixed field  $\text{Fix}(\mathbb{Q}_f, G^\Delta)$ .*
2. *The index of  $G$ -blocks  $[\Sigma : \Delta]$  is equal to the degree  $[\mathbb{Q}_\Delta : \mathbb{Q}_\Sigma]$ .*

*Proof.* Recall that the normal closure of  $\mathbb{Q}_\Delta$  over  $\mathbb{Q}_\Sigma$  is the smallest field containing  $\mathbb{Q}_\Delta$  that is normal over  $\mathbb{Q}_\Sigma$ . The field  $\mathbb{Q}_f$  is a Galois extension of  $\mathbb{Q}_\Sigma$  with Galois group  $G_\Sigma$ . Since  $\mathbb{Q}_\Delta$  is contained in  $\mathbb{Q}_f$  it follows that the normal closure of  $\mathbb{Q}_\Delta$  over  $\mathbb{Q}_\Sigma$  is contained in  $\mathbb{Q}_f$ . Let  $L \subseteq \mathbb{Q}_f$  be any normal extension of  $\mathbb{Q}_\Sigma$  containing  $\mathbb{Q}_\Delta$ . By the fundamental theorem of Galois theory (Theorem 6.1),  $L$  is the fixed field  $\text{Fix}(\mathbb{Q}_f, H)$  for some subgroup  $H$  of  $G_\Sigma$  that is normal in  $G_\Sigma$ . The group  $G(\Sigma/\Delta)$  is the largest subgroup of  $G_\Sigma$  that is normal in  $G_\Sigma$  (Theorem 3.12). Therefore  $\text{Fix}(\mathbb{Q}_f, G(\Sigma/\Delta))$  is the smallest normal extension of  $\mathbb{Q}_\Sigma$  that contains  $\mathbb{Q}_\Delta$  and is thus the normal closure of  $\mathbb{Q}_\Delta$  over  $\mathbb{Q}_\Sigma$ . Let  $\Sigma = \Omega$  then we have  $\mathbb{Q}_\Sigma = \mathbb{Q}_\Omega = \mathbb{Q}$ .

Hence the normal closure of  $\mathbb{Q}_\Delta$  is  $\text{Fix}(\mathbb{Q}_f, G^\Delta)$ . This completes the proof of part 1.

For  $G$ -blocks  $\Delta \subseteq \Sigma$ , by the Galois correspondence of blocks (Theorem 3.11), the index  $[\Sigma : \Delta] = [G_\Sigma : G_\Delta]$ . Consider any  $G$ -block  $\Psi$ . The extension  $\mathbb{Q}_f/\mathbb{Q}_\Psi$  is Galois with Galois group  $G_\Psi$ . Therefore  $[\mathbb{Q}_f : \mathbb{Q}_\Psi] = \#G_\Psi$ . It follows that  $[G_\Sigma : G_\Delta] = \frac{[\mathbb{Q}_f : \mathbb{Q}_\Sigma]}{[\mathbb{Q}_f : \mathbb{Q}_\Delta]} = [\mathbb{Q}_\Delta : \mathbb{Q}_\Sigma]$ . This proves part 2.  $\square$

Proposition 7.2 will play an important role in our algorithms for nilpotence and  $\Gamma_d$  testing. We give polynomial time algorithm for computing the fields  $\mathbb{Q}_\Delta$  in Section 7.1. In Section 7.2 we study the block structure of transitive nilpotent permutation groups. Using these properties we give a nilpotence test. Finally the  $\Gamma_d$ -test is given in section 7.3.

## 7.1 Computing the fields $\mathbb{Q}_\Delta$

The goal of this section is to prove the following theorem that plays an important role in both the property testing algorithms we are going to describe.

**Theorem 7.3.** *Let  $f(X)$  be an irreducible polynomial over  $\mathbb{Q}$  with  $\Omega$  as its set of roots. Let  $G \leq \text{Sym}(\Omega)$  be its Galois group. Let  $\Delta$  be any  $G$ -block of  $\Omega$  such that  $\alpha \in \Delta$  for some  $\alpha \in \Omega$ . There is an algorithm that given the field  $\mathbb{Q}_\Delta$  as a subfield of  $\mathbb{Q}(\alpha)$ , runs in time polynomial in  $\text{size}(f)$  and  $\text{size}(\mathbb{Q}_\Delta)$  and computes the field  $\mathbb{Q}_\Sigma$  for all  $G$ -blocks  $\Sigma$  such that  $\Delta$  is a maximal  $G$ -subblock of  $\Sigma$ . Moreover  $\text{size}(\mathbb{Q}_\Sigma)$  is at most a polynomial in  $\text{size}(f)$  and is independent of the size of the presentation of  $\mathbb{Q}_\Delta$ .*

Although stated differently, this algorithm is due Landau and Miller [39] and is used in their polynomial-time solvability test. Through a sequence of lemmas we prove this theorem in the rest of this section.

For a  $G$ -block  $\Delta$  let  $T_\Delta(X)$  be the polynomial defined by

$$T_\Delta(X) = \prod_{\eta \in \Delta} (X - \eta).$$

**Proposition 7.4.** *If  $T_\Delta(X) = \delta_0 + \delta_1 X + \dots + \delta_{r-1} X^{r-1} + X^r$  then field  $\mathbb{Q}_\Delta$  is the field  $\mathbb{Q}(\delta_0, \dots, \delta_{r-1})$ .*

*Proof.* For any automorphism  $\sigma \in G$  we have  $\sigma(T_\Delta) = T_{\Delta^\sigma}$ . Therefore if  $\sigma$  is in  $G_\Delta$  then  $\sigma(T_\Delta) = T_\Delta$ . Let  $T_\Delta(X) = \delta_0 + \delta_1 X + \dots + \delta_{r-1} X^{r-1} + X^r$ .

Comparing the coefficients of  $X^i$  we have  $\sigma(\delta_i) = \delta_i$  for all  $0 \leq i < r$ . Conversely if for some  $\sigma \in G$ , if  $\sigma(\delta_i) = \delta_i$ ,  $0 \leq i < r$ , then  $\sigma(T_\Delta) = T_\Delta$  and hence  $\sigma \in G_\Delta$ . Thus we have the following proposition.  $\square$

In view of Proposition 7.4, to compute  $\mathbb{Q}_\Delta$  it is sufficient to compute the polynomial  $T_\Delta(X)$ . The algebraic integers  $\delta_i$ 's are symmetric functions on the roots of  $f(X)$  in  $\Delta$  and hence using Lemma 6.15 and Proposition 6.18,  $\text{size}(\delta_i)$  is bounded by a polynomial in  $\text{size}(f)$ . Having computed the polynomial  $T_\Delta$ , one can compute the field  $\mathbb{Q}_\Delta$  in time polynomial in  $\text{size}(f)$ .

We prove the following important lemma on the irreducible factors of  $f(X)$  over  $\mathbb{Q}_\Delta$ .

**Lemma 7.5.** *Let  $\Delta$  be a  $G$ -block containing  $\alpha$ . There is a one-to-one correspondence between irreducible factors of  $f(X)$  over  $\mathbb{Q}_\Delta$  and orbits of  $G_\Delta$  given by*

$$\Omega' = \prod_{\eta \in \Omega'} (X - \eta), \quad \Omega' \text{ a } G\text{-orbit.}$$

*Proof.* Let  $g(X)$  be an irreducible factor of  $f(X)$  over  $\mathbb{Q}_\Delta$ . Then  $G_\Delta = \text{Gal}(\mathbb{Q}_f/\mathbb{Q}_\Delta)$  acts transitively on the roots of  $g(X)$ . Hence for any two roots  $\eta$  and  $\eta'$  of  $g(X)$  there is an element  $\sigma \in G_\Delta$  such that  $\sigma(\eta) = \eta'$ . Therefore  $\eta$  and  $\eta'$  belong to the same  $G_\Delta$ -orbit. Conversely if  $\eta$  and  $\eta'$  belong to the same  $G_\Delta$  orbit then there is a  $\sigma \in G_\Delta$  such that  $\sigma(\eta) = \eta'$  and they are  $\mathbb{Q}_\Delta$ -conjugates. This is possible if and only if  $\eta$  and  $\eta'$  are the roots of the same irreducible factor  $g$  of  $f(X)$  over  $\mathbb{Q}_\Delta$ .  $\square$

The above lemma has the following important corollary.

**Lemma 7.6.** *Let  $\Delta$  be any  $G$ -block containing  $\alpha$ . The polynomial  $T_\Delta$  is the irreducible factor of  $f$  over  $\mathbb{Q}_\Delta$  which has  $\alpha$  as its root. Let  $\Sigma$  be any  $G$ -block such that  $\Sigma \supseteq \Delta$ . If  $g$  is an irreducible factor of  $f$  over  $\mathbb{Q}_\Delta$  then  $\Sigma$  contains a root of  $g$  if and only if it contains all the roots of  $g$ .*

*Proof.* In the correspondence of Lemma 7.5,  $T_\Delta$  corresponds to the orbit of  $\alpha$  under  $G_\Delta$ . Hence  $T_\Delta$  is the factor of  $f(X)$  that has  $\alpha$  as a root.

Let  $\Delta \subseteq \Sigma$  be any two  $G$ -blocks and let  $g(X)$  be an irreducible factor of  $f(X)$  over  $\mathbb{Q}_\Delta$ . Suppose that  $\Sigma$  contains a root  $\eta$  of  $g$ . Any other root  $\eta'$  of  $g$  is in the same  $G_\Delta$  orbit, i.e.  $\eta' \in \eta^{G_\Delta}$ . However since  $\Delta \subseteq \Sigma$  we have  $G_\Delta \leq G_\Sigma$ . Hence  $\eta' \in \eta^{G_\Sigma} = \Sigma$ .  $\square$

The above theorem gives a polynomial time algorithm to identify the polynomial  $T_\Delta(X)$ . Recall that  $\mathbb{Q}_\Delta$  is a subfield of  $\mathbb{Q}(\alpha)$ . The polynomial

$T_\Delta(X)$  is that irreducible factor  $g(X)$  of  $f(X)$  for which  $g(\alpha) = 0$ . We now prove an important lemma from which the proof of Theorem 7.3 is more or less direct.

**Lemma 7.7.** *Let  $\Delta$  be a  $G$ -block containing  $\alpha$ . Given the field  $\mathbb{Q}_\Delta$  as a subfield of  $\mathbb{Q}(\alpha)$  and an irreducible factor  $g$  of  $f$  over  $\mathbb{Q}_\Delta$ , we can compute in polynomial time  $T_\Sigma$  as a polynomial in  $\mathbb{Q}(\alpha)[Y]$ , where  $\Sigma$  is the smallest  $G$ -block containing  $\Delta$  and the roots of  $g$ .*

*Proof.* Let the factorisation of  $f$  over  $\mathbb{Q}_\Delta$  be  $f = g_0 \dots g_r$ , where  $g_0 = T_\Delta$  and  $g = g_1$ . Denote the set of roots of  $g_i$  by  $\Phi_i$ , for each  $i$ . Then by Lemma 7.5,  $\Phi_i$ 's are the orbits of  $G_\Delta$  and the polynomial  $T_\Sigma(X)$  is precisely the product of  $g_i$  such that  $\Phi_i \subseteq \Sigma$ . Let  $\beta$  be any root of  $g(X)$ , and  $\sigma \in \text{Gal}(\mathbb{Q}_f/\mathbb{Q})$  be an automorphism that maps  $\alpha$  to  $\beta$ . The map  $\sigma$  is in fact an isomorphism between the fields  $\mathbb{Q}(\alpha)$  and  $\mathbb{Q}(\beta)$ . Let  $\Sigma$  be the smallest  $G$ -block containing  $\Delta$  and  $\Phi_1$ . It follows from the Galois correspondence of blocks (Theorem 3.11) that  $G_\Sigma$  is generated by  $G_\Delta \cup \{\sigma\}$ . If we knew the permutation  $\sigma$  and the orbits  $\Phi_i$  explicitly then the following transitive closure procedure would give us  $\Sigma$ .

```

S ← {Δ, Φ1}
repeat
  S' ← {Φσ | Φ ∈ S}
  foreach orbit Φj do
    if Φj ∩ Φσ ≠ ∅ for some Φσ ∈ S' then S ← S ∪ {Φj}
  end
until S is unchanged;
Output ∪{Φ | Φ ∈ S}

```

**Algorithm 9:** Computing  $\Sigma$

Our goal is to “simulate” Algorithm 9. The key idea is that the orbit  $\Phi_i$  corresponds to the irreducible factor  $g_i$  of  $f(X)$  over  $\mathbb{Q}_\Delta$  and testing whether  $\Phi_j \cap \Phi_i^\sigma \neq \emptyset$  (step 1 of the Algorithm 9) amounts to checking whether the g.c.d of  $g_j$  and  $g_i^\sigma$  is nontrivial. We give a polynomial time algorithm for computing the g.c.d of  $g_j$  and  $g_i^\sigma$ .

First, we compute the extension field  $\mathbb{Q}(\alpha, \beta)$ . We factor the polynomial  $g(X)$  over the field  $\mathbb{Q}(\alpha)$  into irreducible factors. Let  $h$  be any irreducible factor of  $g$  over  $\mathbb{Q}(\alpha)$  then  $\mathbb{Q}(\alpha, \beta) = \mathbb{Q}(\alpha)[X]/h(X)$ . Since  $[\mathbb{Q}(\alpha, \beta) : \mathbb{Q}] \leq n^2$  and the heights  $H(\alpha) = H(\beta)$  is bounded by  $2^{O(\text{size}(f))}$  (Proposition 6.18), we can compute in polynomial time the explicit data of  $\mathbb{Q}(\alpha, \beta)$ . Furthermore, in polynomial time we compute a primitive element  $\gamma = \alpha + c\beta$ ,

$1 \leq c \leq n^8 + 1$ , of the field  $\mathbb{Q}(\alpha, \beta)$  and polynomials  $a(X)$  and  $b(X)$  in  $\mathbb{Q}[X]$  such that  $\alpha = a(\gamma)$  and  $\beta = b(\gamma)$ .

Any irreducible factors  $g_i(X)$  can be written as a bivariate polynomial  $g_i(X, \alpha)$ . Hence symbolically  $g_i^\sigma(X)$  is the bivariate polynomial  $g_i(X, \beta)$ . In  $g_i(X, \alpha)$  and  $g_i(X, \beta)$  we replace  $\alpha$  and  $\beta$  by  $a(\gamma)$  and  $b(\gamma)$  respectively to get the polynomials  $g_i(X)$  and  $g_i^\sigma(X)$  as polynomials of over  $\mathbb{Q}(\gamma) = \mathbb{Q}(\alpha, \beta)$ . Having computed the polynomials  $g_i^\sigma$  and  $g_j$  as polynomials over the same field  $\mathbb{Q}(\gamma)$ , one can compute their g.c.d in polynomial time. The complete algorithm to compute the polynomial  $T_\Sigma$  is given below (Algorithm 10). Clearly Algorithm 10 runs in time bounded by a polynomial in the input size. The correctness of the algorithm follows from the correctness of Algorithm 9.

```

S ← {TΔ, g}
repeat
  S' ← {giσ | gi ∈ S}.
  foreach factor gj do
    if gcd(gj, hσ) is nontrivial for some hσ ∈ S' then
      S ← S ∪ {gj}
  end
until S is unchanged;
Output TΣ = ∏gi ∈ S gi

```

**Algorithm 10:** Computing  $T_\Sigma$

□

We now complete the proof of Theorem 7.3. By Proposition 7.4, it suffices to compute the set  $\mathcal{S}$  of polynomials  $T_\Sigma$  such that  $\Sigma$  is a minimal  $G$ -block properly containing  $\Delta$ . Let  $f(X)$  factor as  $f(X) = g_0 \dots g_r$  over  $\mathbb{Q}_\Delta$  with  $g_0 = T_\Delta$ .

Let  $\Sigma_i$  be the smallest  $G$ -block containing  $\Delta$  and all the roots of  $g_i$ . For any  $G$ -block  $\Sigma$  such that  $\Delta$  is a maximal  $G$ -subblock of  $\Sigma$ , there is an  $i$ ,  $1 \leq i \leq r$  such that  $\Sigma = \Sigma_i$ . Using Lemma 7.7 we compute  $T_{\Sigma_i}$ 's for each  $1 \leq i \leq r$ . The  $G$ -block  $\Sigma_j \subseteq \Sigma_i$  if and only if  $T_{\Sigma_j}$  divides  $T_{\Sigma_i}$  and hence  $\Sigma_i$  is a minimal  $G$ -block properly containing  $\Delta$  if and only if  $T_{\Sigma_i}$  is not divisible by  $T_{\Sigma_j}$  for all  $j \neq i$ . The set  $\mathcal{S}$  is the collection of all the polynomials  $T_{\Sigma_i}$  such that for all  $j \neq i$ ,  $T_{\Sigma_j} \nmid T_{\Sigma_i}$ . Clearly computing  $\mathcal{S}$  is in polynomial time.

Having computed the set  $\mathcal{S}$  we compute the fields  $\mathbb{Q}_{\Sigma_i}$  for all polynomials  $T_{\Sigma_i}(X) \in \mathcal{S}$ . Recall that  $\mathbb{Q}_{\Sigma_i}$  is obtained by adjoining the coefficients of the polynomial  $T_{\Sigma_i}$  each of which are symmetric functions of roots of  $f(X)$

in  $\Sigma_i$ . Thus although computing  $\mathbb{Q}_{\Sigma_i}$  takes time proportional to  $\text{size}(f)$  and  $\text{size}(\mathbb{Q}_\Delta)$ , the size of the explicit data computed for the field  $\mathbb{Q}_{\Sigma_i}$  is polynomial in  $\text{size}(f)$  and is independent of the size of presentation of  $\mathbb{Q}_\Delta$ . This completes the proof of Theorem 7.3.

*Remark 7.8.* That the size of the computed presentation of  $\mathbb{Q}_\Sigma$  is bounded by a polynomial in  $\text{size}(f)$  and is independent on the size of  $\mathbb{Q}_\Delta$  is important because Theorem 7.3 will be used repeatedly in our algorithms to compute a tower of fields  $\mathbb{Q}_{\Delta_0} \supset \dots \supset \mathbb{Q}_{\Delta_m}$  for a maximal chain of  $G$ -blocks  $\Delta_0 \subset \dots \subset \Delta_m$ . The length  $m$  of such a chain of  $G$ -blocks could be as large as  $\lg n$  where  $n$  is the degree of  $f$ . If the size of  $\mathbb{Q}_{\Delta_i}$  depended on the size of presentation of  $\mathbb{Q}_{\Delta_{i-1}}$  then the presentation of  $\mathbb{Q}_{\Delta_m}$  could be as large as  $n^{\lg n}$ .

## 7.2 Nilpotence testing for Galois groups

In this section we give a polynomial time algorithm for testing whether the Galois group of a given polynomial is nilpotent. We give a characterisation of transitive nilpotent groups which can be tested in polynomial time. Recall that a finite group  $G$  is *nilpotent* if and only if every Sylow subgroup of  $G$  is normal (see Lemma 3.5 for other equivalent definitions). For a nilpotent group  $G$  and a prime  $p$  that divides  $\#G$ , there is a unique  $p$ -Sylow subgroup which we denote in this section by  $G_p$ . In fact  $G_p$  is the set of all element of  $G$  that has order a power of  $p$ . Moreover any subgroup  $H$  of  $G$  is also nilpotent and the  $p$ -Sylow subgroup of  $H$  is  $G_p \cap H$ . If  $\{p_1, \dots, p_k\}$  are the set of prime factors of  $\#G$  then  $G = G_{p_1} \times \dots \times G_{p_k}$ .

**Lemma 7.9.** *Let  $G$  be a transitive nilpotent permutation group on  $\Omega$  then*

1. *For all primes  $p$ ,  $p$  divides  $\#G$  if and only if  $p$  divides  $\#\Omega$ .*
2. *For any prime  $p \mid \#G$  and  $\alpha \in \Omega$  there is a block  $\Sigma_p^\alpha$  containing  $\alpha$  such that  $\#\Sigma_p^\alpha$  is the highest power of  $p$  that divides  $\#\Omega$ .*
3. *Let  $\Delta$  be any  $G$ -block containing  $\alpha$  such that  $\#\Delta = p^l$  for some prime  $p$  dividing  $\#G$ . Then  $\Delta \subseteq \Sigma_p^\alpha$ . Also for all  $q$  different from  $p$  the  $q$ -Sylow subgroup of  $G_\Delta$  is same as the  $q$ -Sylow subgroup of  $G_\alpha$ , i.e.  $G_q \cap G_\Delta = G_q \cap G_\alpha$ .*

*Proof.* As  $G$  is transitive on  $\Omega$ ,  $\#\Omega$  divides  $\#G$  by Orbit-Stabiliser formula (Theorem 3.8). Hence, each prime factor of  $\#\Omega$  divides  $\#G$ . Conversely let

$p$  be a prime factor of  $\#G$ . For  $\alpha \in \Omega$ , the set  $\Sigma_p^\alpha = \alpha^{G_p}$  is a nontrivial  $G$ -block as  $G_p$  is a normal subgroup of  $G$  (Lemma 3.13). Since  $G_p$  is transitive on  $\Sigma_p^\alpha$ , it follows from the Orbit-Stabiliser formula that  $\#\Sigma_p^\alpha$  divides  $\#G_p$ . Hence  $\#\Sigma_p^\alpha$  is  $p^l$  for some  $l > 0$ . Since  $p$  divides the cardinality of a  $G$ -block  $\Sigma_p^\alpha$ ,  $p$  must divide  $\#\Omega$ . This proves part 1.

Next, we prove (2). From the Galois correspondence of  $G$ -blocks (Theorem 3.11) we have  $[\Omega : \Sigma_p^\alpha] = [G : G_{\Sigma_p^\alpha}]$ . The prime  $p$  does not divide  $[G : G_p]$  as  $G_p$  is the  $p$ -Sylow subgroup of  $G$ . Therefore  $p$  does not divide  $[G : G_{\Sigma_p^\alpha}]$  either as  $G_p$  is a subgroup of  $G_{\Sigma_p^\alpha}$ . Hence  $p$  is not a factor of  $[\Omega : \Sigma_p^\alpha]$  and  $\#\Sigma_p^\alpha$  is the highest power of  $p$  that divides  $\#\Omega$ .

To prove part 3 notice that  $G_\Delta$  is a nilpotent group with the unique normal  $q$ -Sylow subgroup  $G_q \cap G_\Delta$ . Therefore we have  $G_\Delta = \prod_q (G_q \cap G_\Delta)$ . By the Galois correspondence (Theorem 3.11) of blocks we have

$$\#\Delta = [G_\Delta : G_\alpha] = \prod_q [G_q \cap G_\Delta : G_q \cap G_\alpha]. \quad (7.1)$$

Since  $G_q \cap G_\Delta$  is a  $q$ -group, the prime  $p$  divides the index  $[G_q \cap G_\Delta : G_q \cap G_\alpha]$  if and only if  $q = p$ . However, in Equation 7.1  $\#\Delta$  is a power of  $p$ . This is possible if and only if  $[G_q \cap G_\Delta : G_q \cap G_\alpha] = 1$  for all  $q \neq p$ . Thus  $G_q \cap G_\Delta = G_q \cap G_\alpha$  for all  $q$  different from  $p$ .

The group  $G_\Delta$  is therefore the product group  $G_p \cap G_\Delta \times \prod_{q \neq p} G_q \cap G_\alpha$ . Since the group  $G_{\Sigma_p^\alpha}$  contains both  $G_p$  and  $G_\alpha$  we have  $G_{\Sigma_p^\alpha} \geq G_\Delta$ . Thus by Galois correspondence of blocks (Theorem 3.11),  $\Delta$  is a  $G$ -subblock of  $\Sigma_p^\alpha$ .  $\square$

Nilpotent groups behave almost like  $p$ -groups. Let  $G$  be a transitive nilpotent permutation group on  $\Omega$  and let  $p$  be a prime dividing  $\#G$ . We prove that as far as  $G$ -blocks contained in  $\Sigma_p^\alpha$  are concerned,  $G$  behaves like  $G_p$ . The following lemma makes this precise.

**Lemma 7.10.** *Let  $G$  be a transitive nilpotent permutation group acting on  $\Omega$ . Let  $p$  be any prime that divides  $\#G$  and let  $G_p$  be the corresponding  $p$ -Sylow subgroup. Consider any element  $\alpha \in \Omega$  and let  $\Sigma_p^\alpha$  be the  $G$ -block  $\alpha^{G_p}$ . A set  $\Delta \subseteq \Sigma_p^\alpha$  is a  $G$ -block if and only if  $\Delta$  is a  $G_p$ -block under the transitive action of  $G_p$  on  $\Sigma_p^\alpha$ .*

*Proof.* Clearly any  $G$ -block contained in  $\Sigma_p^\alpha$  is a  $G_p$ -block as  $G$  contains  $G_p$ . Conversely consider a  $G_p$ -block  $\Delta$  of  $\Sigma_p^\alpha$ . The group  $G_p \cap G_\Delta$  contains  $G_p \cap G_\alpha$ . To see this consider the transitive action of  $G_p$  restricted to  $\Sigma_p^\alpha$ . The restriction action is a homomorphism  $\psi : G_p \rightarrow \text{Sym}(\Sigma_p^\alpha)$ . Let  $H$

denote the image  $\psi(G_p) = G_p|_{\Sigma_p^\alpha}$ . The groups  $G_p \cap G_\Delta$  and  $G_p \cap G_\alpha$  are the pullbacks  $\psi^{-1}(H_\Delta)$  and  $\psi^{-1}(H_\alpha)$  respectively. Since the subset  $\Delta$  is a  $H$ -block of  $\Sigma_p^\alpha$  and contains  $\alpha$ ,  $H_\Delta \geq H_\alpha$ . Therefore  $G_p \cap G_\Delta \geq G_p \cap G_\alpha$ .

Consider the group  $G' = (G_p \cap G_\Delta) \times \prod_{q \neq p} G_q \cap G_\alpha$ . The group  $G_\alpha$  is nilpotent and hence  $G_\alpha = (G_p \cap G_\alpha) \times \prod_{q \neq p} G_q \cap G_\alpha$ . Since  $G_p \cap G_\Delta \geq G_p \cap G_\alpha$  we have  $G' \geq G_\alpha$ . Therefore by the Galois correspondence of blocks (Theorem 3.11) we have  $\Delta = \alpha^{G'}$  is a  $G$ -block between  $\{\alpha\}$  and  $\Sigma_p^\alpha$ .  $\square$

We now study the structure of blocks of a  $p$ -group. We state the following result due to Luks [46, Lemma 1.1].

**Lemma 7.11** (Luks). *Let  $G$  be a  $p$ -group acting transitively on  $\Omega$  and let  $\Delta$  be a maximal  $G$ -block. Then the index  $[\Omega : \Delta]$  is exactly  $p$  and  $G_\Delta = G(\Omega/\Delta) = G^\Delta$  is a normal group of index  $p$  in  $G$ .*

*Proof.* Let  $\Delta$  be a maximal  $G$ -block. By Galois correspondence of blocks we have  $[\Omega : \Delta] = [G : G_\Delta]$ . Suppose that  $[G : G_\Delta] = p^l$  for  $l \geq 1$ . The group  $G$  being a  $p$ -group, it follows that there is a subnormal series  $G = G_0 \triangleright G_1 \dots \triangleright G_l = G_\Delta$  such that  $[G_i : G_{i+1}] = p$  [30, Theorem 4.3.2]. Let  $\alpha$  be any element of  $\Delta$ . Since  $G_\Delta \geq G_\alpha$ ,  $(G_i)_\alpha = G_\alpha$ . Therefore by Orbit-Stabiliser formula  $\frac{\#\alpha^G}{\#\alpha^{G_1}} = \frac{\#G}{\#G_1} = [G : G_1] = p$ . However  $G_1$  is a normal subgroup of  $G$  and  $\alpha^{G_1} \neq \Omega$ . Therefore  $\alpha^{G_1}$  is the maximal block  $\Delta$  and  $G_\Delta = G_1$ .

Recall that  $G^\Delta = G(\Omega/\Delta)$  is the largest normal subgroup of  $G = G_\Omega$  contained in  $G_\Delta$  (Theorem 3.12). However  $G_\Delta = G_1$  itself is normal. Hence  $G_\Delta = G^\Delta$ .  $\square$

Applying Lemma 7.11 repeatedly we have the following lemma.

**Lemma 7.12.** *Let  $G$  be a transitive  $p$ -group acting on  $\Omega$  and  $\alpha \in \Omega$ . Let  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_t = \Omega$  be any maximal chain of  $G$ -blocks. Then*

1.  $[\Delta_{i+1} : \Delta_i] = p$  for all  $0 \leq i < t$ .
2.  $G(\Delta_{i+1}/\Delta_i) = G_{\Delta_i}$ .
3. The group  $G_{\Delta_i}$  is a normal subgroup of  $G_{\Delta_{i+1}}$  and the quotient group  $G_{\Delta_{i+1}}/G_{\Delta_i}$  is cyclic of order  $p$ .

*In particular any minimal  $G$ -block is of cardinality  $p$ .*

We now prove the following important property of transitive nilpotent permutation groups.

**Lemma 7.13.** *Let  $G$  be a transitive nilpotent permutation group on  $\Omega$ . Let  $p$  be any prime dividing  $\#G$ . Let  $\Delta$  be any  $G$ -block such that  $\#\Delta = p^l$  for some integer  $l \geq 0$ . Let  $m$  be the highest power of  $p$  that divides  $\#\Omega$ . If  $l < m$  then we have*

1. *There exists a  $G$ -block  $\Sigma$  such that  $\Delta$  is a maximal  $G$ -subblock of  $\Sigma$  and  $[\Sigma : \Delta] = p$ .*
2. *For all  $G$ -blocks  $\Sigma$  such that  $\Delta$  is a maximal  $G$ -subblock of  $\Sigma$  and  $[\Sigma : \Delta] = p$ ,  $G_\Delta$  is a normal subgroup of  $G_\Sigma$ .*

*Proof.* Let  $\Sigma_p^\alpha$  as before denote the  $G$ -blocks  $\alpha^{G_p}$ . Since  $\#\Delta$  is a power of  $p$  it follows that  $\Delta$  is a  $G$ -subblock of  $\Sigma_p^\alpha$  (Lemma 7.9). The subset  $\Delta$  is a  $G_p$ -block on the transitive action of  $G_p$  on  $\Sigma_p^\alpha$  (Lemma 7.10). Consider the action of the  $p$ -group  $G_p$  on  $\Sigma_p^\alpha$ . If  $l < m$  there is a  $G_p$ -block  $\Sigma$  such that  $\Sigma_p^\alpha \supseteq \Sigma \supset \Delta$  and  $[\Sigma : \Delta] = p$ . By Lemma 7.10 it follows that  $\Sigma$  is a  $G$ -block contained in  $\Sigma_p^\alpha$ . This proves part 1.

Let  $\alpha$  be any element of  $\Delta$ . It follows from Lemma 7.9 that for all  $q \neq p$  the  $q$ -Sylow subgroup of  $G_\Sigma$  and  $G_\Delta$  are both  $G_q \cap G_\alpha$ . Let  $\widehat{G}_p$  be the product group  $\prod_{q \neq p} G_q$ . The groups  $G_\Sigma$  and  $G_\Delta$  are  $(G_p \cap G_\Sigma) \times (\widehat{G}_p \cap G_\alpha)$  and  $(G_p \cap G_\Delta) \times (\widehat{G}_p \cap G_\alpha)$  respectively. Moreover the groups  $G_p \cap G_\Sigma$  and  $G_p \cap G_\Delta$  are  $p$ -groups with index  $[G_p \cap G_\Sigma : G_p \cap G_\Delta] = [G_\Sigma : G_\Delta] = [\Sigma : \Delta] = p$ . Therefore  $G_p \cap G_\Delta$  is a normal subgroup of  $G_p \cap G_\Sigma$ . As a result  $G_\Delta = (G_p \cap G_\Delta) \times (\widehat{G}_p \cap G_\alpha)$  is a normal subgroup of  $G_\Sigma = (G_p \cap G_\Sigma) \times (\widehat{G}_p \cap G_\alpha)$  and the quotient group  $\frac{G_\Sigma}{G_\Delta} = \frac{G_p \cap G_\Sigma}{G_p \cap G_\Delta}$  is isomorphic to  $\mathbb{F}_p$ . □

We give the following characterisation of transitive nilpotent groups.

**Theorem 7.14.** *Let  $G$  be a transitive permutation group on  $\Omega$  then the following are equivalent.*

1.  *$G$  is nilpotent.*
2. *For all primes  $p$  dividing  $\#G$ ,  $p$  divides  $\#\Omega$  and there exists a maximal chain of  $G$ -block  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m$  such that*
  - (a)  *$m$  is the highest power of  $p$  dividing  $\#\Omega$ .*
  - (b)  *$G_{\Delta_i}$  is a normal subgroup of  $G_{\Delta_{i+1}}$ .*
  - (c)  *$[\Delta_{i+1} : \Delta_i] = p$  for all  $0 \leq i < m$ .*
  - (d)  *$p \nmid [G : G^{\Delta_m}]$ .*

*Proof.* If  $G$  is nilpotent then condition 2 holds. The required maximal chain of  $G$ -blocks is any maximal chain between  $\{\alpha\}$  and  $\Sigma_p^\alpha$ . We now prove the converse.

Consider any prime  $p$  dividing  $\#G$ . The prime  $p$  divides  $\#\Omega$  and let  $m > 0$  be the highest power of  $p$  dividing  $\#\Omega$ . Let  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m$  be a maximal chain of  $G$ -blocks satisfying the conditions 2a–2d. We prove that  $G^{\Delta_m}$  is the unique  $p$ -Sylow subgroup for  $G$ .

Recall that  $G(\Delta_{i+1}/\Delta_i)$  is the largest subgroup of  $G_{\Delta_i}$  that is normal in  $G_{\Delta_{i+1}}$  (Theorem 3.12). However since  $G_{\Delta_i}$  itself is a normal subgroup of  $G_{\Delta_{i+1}}$  it follows that  $G_{\Delta_i} = G(\Delta_{i+1}/\Delta_i)$ . Moreover  $[G_{\Delta_{i+1}} : G_{\Delta_i}] = [\Delta_{i+1} : \Delta_i] = p$  and therefore  $[G_{\Delta_{i+1}} : G(\Delta_{i+1}/\Delta_i)] = p$ .

The group  $\frac{G^{\Delta_{i+1}}}{G^{\Delta_i}}$  is a subgroup of the  $l_i$ -fold product of  $\frac{G_{\Delta_{i+1}}}{G(\Delta_{i+1}/\Delta_i)}$  (Theorem 3.12). Hence  $\frac{G^{\Delta_{i+1}}}{G^{\Delta_i}}$  is of order  $p^l$  for some  $l$ . As a result we have

$$\#G^{\Delta_m} = [G^{\Delta_m} : G^{\Delta_{m-1}}] \dots [G^{\Delta_1} : G^{\Delta_0}] = \text{a power of } p.$$

The group  $G^{\Delta_m}$  is thus a  $p$ -group. Furthermore  $p \nmid [G : G^{\Delta_m}]$  (condition 2d). Therefore  $G^{\Delta_m}$  is a  $p$ -Sylow subgroup of  $G$ . Moreover the group  $G^{\Delta_m} = G(\Omega/\Delta_m)$  is also a normal subgroup of  $G = G_\Omega$  (part 1 of Theorem 3.12). Thus we have shown that for every prime  $p$  that divides  $\#G$  the  $p$ -Sylow subgroup is normal. This proves that  $G$  is nilpotent.  $\square$

### 7.2.1 The nilpotence test

Given  $f(X) \in \mathbb{Q}[X]$  we want to test if the Galois group of  $f(X)$  is nilpotent. If  $f$  is reducible then the Galois group of  $f$  is nilpotent if and only if the Galois group of each of its irreducible factors is nilpotent. This is because nilpotent groups are closed under products and subgroups. Since in polynomial time one can factor polynomials over  $\mathbb{Q}$  (Theorem 6.7), without loss of generality we assume that  $f(X)$  is irreducible. Let  $G$  be the Galois group of  $f(X)$  considered as a subgroup of  $\text{Sym}(\Omega)$ , where  $\Omega$  is the set of roots of  $f(X)$ . Since  $f$  is irreducible,  $G$  is transitive on  $\Omega$ .

We describe the main idea behind the algorithm. It follows from Theorem 7.14 that  $G$  is nilpotent if and only if for all primes  $p$  that divide the order of  $G$ , there is a maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m$  satisfying the conditions of part 2 of Theorem 7.14. We do not have access to the sets  $\Delta_i$  and the groups  $G_{\Delta_i}$ . However we prove that conditions in part 2 of Theorem 7.14 can be verified once the fields  $\mathbb{Q}(\alpha) = \mathbb{Q}_{\Delta_0} \supset \dots \supset \mathbb{Q}_{\Delta_m}$  are known. Recall that for a  $G$ -block  $\Delta$ ,  $\mathbb{Q}_\Delta$  is the fixed field of the splitting

field  $\mathbb{Q}_f$  under the automorphisms of  $G_\Delta$ . Algorithm 11 is the complete algorithm.

**Input:** A polynomial  $f(X) \in \mathbb{Q}[X]$  of degree  $n$ .

**Result:** *Accepts*  $f$  if Galois group of  $f(X)$  is nilpotent, *Rejects* otherwise.

Verify whether  $f(X)$  is solvable.

- 1 Compute the set  $P$  of all the prime factors of  $\#\text{Gal}(f)$ .  
Let  $G$  denote the Galois group of  $f$  thought of as a permutation group on  $\Omega$ , the set of roots of  $f$ .
  - 2 **foreach**  $p \in P$  **do**
  - 3     **if**  $p$  does not divide  $n$  **then Reject.**  
Let  $m$  be the highest power of  $p$  dividing  $n$ .  
 $\mathbb{Q}_{\Delta_0} \leftarrow \mathbb{Q}[X]/f(X)$ .
  - 4     **for**  $i \leftarrow 1$  **to**  $m$  **do**  
      Using Theorem 7.3 compute the set of fields  
$$\mathcal{F} = \{\mathbb{Q}_\Sigma \mid \Delta \text{ is a maximal } G\text{-block of } \Sigma\}.$$
  
      Let  $\mathbb{Q}_\Sigma$  be any field of  $\mathcal{F}$  such that  $[\mathbb{Q}_\Sigma : \mathbb{Q}_{\Delta_{i-1}}] = p$ . If no such field exists then **Reject.**
  - 6       **if**  $\mathbb{Q}_\Sigma/\mathbb{Q}_{\Delta_{i-1}}$  is not normal **then Reject.**  
      **else**  $\mathbb{Q}_{\Delta_i} \leftarrow \mathbb{Q}_\Sigma$ .
  - end**  
      Let  $\mu_{\Delta_m}(X)$  be the primitive polynomial for  $\mathbb{Q}_{\Delta_m}$ .
  - 7     **if**  $p$  divides  $\#\text{Gal}(\mu_{\Delta_m})$  **then Reject.**
  - end**
- Accept.**

**Algorithm 11:** Nilpotence test

Given a polynomial  $f(X)$  with solvable Galois group, as a by product of the Landau-Miller test [39], there is a polynomial time algorithm to compute the prime factors of  $\#\text{Gal}(f)$  (see also Theorem 7.23). Therefore the steps 1 and 7 of Algorithm 11 can be performed in polynomial time. All other steps can clearly be performed in polynomial time. This gives us the following proposition.

**Proposition 7.15.** *Algorithm 11 runs in time polynomial in size( $f$ ).*

We now argue the correctness of the algorithm in the following two propositions.

**Proposition 7.16.** *Algorithm 11 accepts  $f(X)$  if the Galois group of  $f$  is nilpotent.*

*Proof.* Let  $G$  be the Galois group of  $f(X)$  and let  $p$  be any prime that divides  $\#G$ . Let  $G_p$  be the  $p$ -Sylow subgroup of  $G$  and let  $\Sigma_p^\alpha$  be the  $G$ -block  $\alpha^{G_p}$ . The loop in step 4 in fact constructs the tower of fields  $\mathbb{Q}_{\Sigma_p^\alpha} = \mathbb{Q}_{\Delta_m} \subset \dots \subset \mathbb{Q}_{\Delta_0} = \mathbb{Q}(\alpha)$  for a maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m = \Sigma_p^\alpha$ . Lemma 7.13 guarantees that the step 5 will never fail.

The extension  $\mathbb{Q}_{\Delta_i}/\mathbb{Q}_{\Delta_{i-1}}$  is normal because  $G_{\Delta_{i-1}}$  is a normal subgroup of  $G_{\Delta_i}$ . Let  $K$  be the normal closure of  $\mathbb{Q}_{\Delta_m}$  then it follows from Proposition 7.2 that  $\text{Gal}(\mathbb{Q}_f/K)$  is  $G^{\Delta_m}$ . The Galois group of  $\mu_{\Delta_m}(X)$  is the quotient group  $\frac{G}{G^{\Delta_m}}$ . Since  $G^{\Delta_m} = G^{\Sigma_p^\alpha} = G_p$ ,  $p$  does not divide the order of the Galois group of  $\mu_{\Delta_m}(X)$ .

Thus no step in the loop 4 will reject the input if the Galois group of  $f$  is nilpotent. This completes the proof.  $\square$

We now prove the converse.

**Proposition 7.17.** *If Algorithm 11 accepts then the Galois group of  $f(X)$  is nilpotent.*

*Proof.* Let  $\Omega$  be the roots of  $f(X)$  and let  $G$  be the Galois group of  $f(X)$  as a permutation group on  $\Omega$ . Since the algorithm has accepted  $f(X)$  we have the following conditions of the Galois group  $G$  of  $f(X)$ .

1. Every prime  $p$  that divides  $\#G$  also divides  $n = \#\Omega$ . This is verified in step 3.
2. For any prime  $p$  dividing  $\#G$  let  $m$  be the highest power of  $p$  dividing  $n$ . There is a maximal chain  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m$  of  $G$ -blocks such that for all  $0 \leq i < m$ 
  - (a)  $G_{\Delta_i}$  is a normal subgroup of  $G_{\Delta_{i+1}}$ . We verified this in step 6 by checking that the extension  $\mathbb{Q}_{\Delta_{i+1}}/\mathbb{Q}_{\Delta_i}$  is a normal.
  - (b)  $[\Delta_{i+1} : \Delta_i] = p$ . This is because  $[\Delta_{i+1} : \Delta_i] = [\mathbb{Q}_{\Delta_{i+1}} : \mathbb{Q}_{\Delta_i}] = p$  (Proposition 7.2).
  - (c) The prime  $p$  does not divide  $[G : G^{\Delta_m}]$ . As argued before the Galois group of the polynomial  $\mu_{\Delta_m}$ , a primitive polynomial of  $\mathbb{Q}_{\Delta_m}$ , is the Galois group  $\frac{G}{G^{\Delta_m}}$ . Thus in step 7 we have verified that  $p$  does not divide  $\#\frac{G}{G^{\Delta_m}} = [G : G^{\Delta_m}]$ .

Hence from Theorem 7.14,  $G$  is nilpotent.  $\square$

Combining Propositions 7.15, 7.16 and 7.17 we have the main theorem of this section.

**Theorem 7.18.** *Given a polynomial  $f(X) \in \mathbb{Q}[X]$ , there is an algorithm that runs in time polynomial in size  $(f)$  that decides whether the Galois group of  $f$  is nilpotent.*

### 7.3 $\Gamma_d$ -testing for Galois groups

In this section we show that the technique underlying the Landau-Miller solvability test can be adapted to efficiently solve a more general problem, the problem of testing whether the Galois group of a polynomial  $f(X) \in \mathbb{Q}[X]$  is in  $\Gamma_d$  for constant  $d$ . Recall that a group  $G$  is in  $\Gamma_d$  if there is a composition series  $G = G_0 \triangleright \dots \triangleright G_t = \{1\}$  such that  $G_i/G_{i+1}$  is either abelian or isomorphic to a subgroup of  $S_d$ . Given a polynomial  $f(X)$  over  $\mathbb{Q}$  of degree  $n$ , we give an algorithm that runs in time polynomial in size  $(f)$  and  $n^d$  to check whether the Galois group of  $f$  is in  $\Gamma_d$ . For constant  $d$  this yields a polynomial time  $\Gamma_d$ -test. As a byproduct of our polynomial time  $\Gamma_d$ -testing, we obtain a polynomial time algorithm to compute the prime factors of  $\#\text{Gal}(f)$  for any polynomial  $f$  with Galois group in  $\Gamma_d$ . Note that for  $d < 5$ ,  $\Gamma_d$  is the class of solvable groups and hence our result is a generalisation of the result of Landau-Miller [39].

We are given a polynomial  $f(X)$  over  $\mathbb{Q}$ . Since the class  $\Gamma_d$  is closed under subgroups and quotients and products, without loss of generality assume that  $f(X)$  is irreducible of degree  $n$ . For describing the  $\Gamma_d$  test we fix the following notation for the rest of this section. Let  $G$  be the Galois group of  $f$ . Consider the faithful action of  $G$  as a permutation group on  $\Omega$ , the set of roots of  $f$ . Let  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m = \Omega$  be any maximal chain of  $G$ -blocks. Recall that for all  $0 \leq i < m$  the group  $G(\Delta_{i+1}/\Delta_i)$  is a normal subgroup of  $G_{\Delta_{i+1}}$  (Theorem 3.12). We have the following proposition.

**Proposition 7.19.** *The group  $G$  is in  $\Gamma_d$  if and only if the quotient groups  $\frac{G_{\Delta_{i+1}}}{G(\Delta_{i+1}/\Delta_i)}$ ,  $0 \leq i < m$ , are all in  $\Gamma_d$ .*

*Proof.* The series  $G = G^{\Delta_t} \triangleright \dots \triangleright G^{\Delta_0} = 1$  gives a normal series for  $G$ . Hence  $G$  is in  $\Gamma_d$  if and only if for each  $0 \leq i < m$  the quotient  $\frac{G^{\Delta_{i+1}}}{G^{\Delta_i}}$  is in  $\Gamma_d$ . Consider the subgroups  $G_{\Delta_{i+1}}$  and  $G(\Delta_{i+1}/\Delta_i)$  of  $G$ . If  $G$  is in  $\Gamma_d$  so are  $G_{\Delta_{i+1}}$  and  $G(\Delta_{i+1}/\Delta_i)$  and hence their quotient  $\frac{G_{\Delta_{i+1}}}{G(\Delta_{i+1}/\Delta_i)}$  (Proposition 3.3). On the other hand,  $\frac{G^{\Delta_{i+1}}}{G^{\Delta_i}}$  is isomorphic to a subgroup

of  $\left(\frac{G_{\Delta_{i+1}}}{G(\Delta_{i+1}/\Delta_i)}\right)^l$  for some  $l$  (Theorem 3.12) and therefore  $\frac{G_{\Delta_{i+1}}}{G_{\Delta_i}}$  is in  $\Gamma_d$  if  $\frac{G_{\Delta_{i+1}}}{G(\Delta_{i+1}/\Delta_i)}$  is in  $\Gamma_d$ . Hence  $G$  is in  $\Gamma_d$  if and only if for each  $0 \leq i < m$  the quotient group  $\frac{G_{\Delta_{i+1}}}{G(\Delta_{i+1}/\Delta_i)}$  is in  $\Gamma_d$ .  $\square$

We have no access to the groups  $G_{\Delta_{i+1}}$  and  $G(\Delta_{i+1}/\Delta_i)$ . However using Theorem 7.3 we can compute the field  $K_i = \mathbb{Q}_{\Delta_i}$ ,  $0 \leq i \leq m$ , for some maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m = \Omega$ . Let  $L_i$  be the normal closure of  $K_{i-1}$  over  $K_i$ . Using Proposition 7.2 and 7.19 we have the following proposition.

**Proposition 7.20.** *The Galois group  $G$  is in  $\Gamma_d$  if and only the Galois groups  $\text{Gal}(L_i/K_i)$ ,  $1 \leq i \leq m$ , is in  $\Gamma_d$ . Furthermore, if  $G$  is in  $\Gamma_d$  then the degree  $[L_i : \mathbb{Q}] = n^{O(d)}$ .*

*Proof.* The field  $L_i$  is the fixed field  $\text{Fix}(\mathbb{Q}_f, G(\Delta_i/\Delta_{i-1}))$  (Proposition 7.2) and hence the Galois group  $\text{Gal}(\mathbb{Q}_f/L_i)$  is  $G(\Delta_i/\Delta_{i-1})$ . Moreover the Galois group of  $\mathbb{Q}_f/\mathbb{Q}_{\Delta_i}$  is  $G_{\Delta_i}$  and hence by the fundamental theorem of Galois theory (Theorem 6.1), the Galois group  $\text{Gal}(L_i/K_i)$  is the quotient group  $\frac{G_{\Delta_i}}{G(\Delta_i/\Delta_{i-1})}$ . It then follows from Proposition 7.19 that  $G$  is in  $\Gamma_d$  if and only if each of the Galois groups  $\text{Gal}(L_i/K_i)$  is in  $\Gamma_d$ .

The block  $\Delta_{i-1}$  is a maximal  $G$ -subblock of  $\Delta_i$ . Recall that the group  $\frac{G_{\Delta_i}}{G(\Delta_i/\Delta_{i-1})}$  acts faithfully as a primitive permutation group on the set of  $G$ -blocks  $\mathcal{B}(\Delta_i/\Delta_{i-1})$  (Theorem 3.12). Moreover if  $G$  is in  $\Gamma_d$  then so is  $\frac{G_{\Delta_i}}{G(\Delta_i/\Delta_{i-1})}$  and hence by the Babai-Cameron-Pálffy bound (Theorem 3.10) we have

$$[L_i : K_i] = \#\text{Gal}(L_i/K_i) = \#\frac{G_{\Delta_i}}{G(\Delta_i/\Delta_{i-1})} \leq [\Delta_i : \Delta_{i-1}]^{O(d)} \leq n^{O(d)}$$

Therefore  $[L_i : \mathbb{Q}] \leq n^{O(d)}$ .  $\square$

The above proposition in particular implies that if  $G$  is in  $\Gamma_d$  then the fields  $L_i$  can be computed in time polynomial in  $\text{size}(f)$  and  $n^d$ . To see this note that we have computed the explicit data of the fields  $K_i$  and  $K_{i-1}$  which are of size at most a polynomial in  $\text{size}(f)$ . Since the degree of the normal closure  $L_i$  of  $K_{i-1}$  over  $K_i$  is bounded by a polynomial in  $n^d$ , we can use Landau's algorithm (Theorem 6.10) to compute the field  $L_i$ . Thus we have the following proposition.

**Proposition 7.21.** *If the Galois group  $G$  is in  $\Gamma_d$  then there is an algorithm that runs in time polynomial in  $\text{size}(f)$  and  $n^d$  to compute the fields  $L_i$ .*

We now describe the polynomial time algorithm for  $\Gamma_d$ -testing. The algorithm first computes the fields  $K_i$  in time polynomial in size  $(f)$ . Let  $b(n)$  be the bound on the size of primitive subgroups of  $S_n$  that are in  $\Gamma_d$ . By the Babai-Cameron-Pálffy bound we have  $b(n) = n^{O(d)}$ . For each  $i$ , using Landau's algorithm (Theorem 6.10), checks whether the degree  $[L_i : K_i]$  is at most  $b(n)$  and if yes computes it. If any of the degrees  $[L_i : K_i]$  is greater than  $b(n)$  then clearly  $G$  is not in  $\Gamma_d$ .

Having computed the fields  $L_i$  and  $K_i$ , the Galois groups  $\text{Gal}(L_i/K_i)$  are explicitly computed using Landau's algorithm. In time bounded by a polynomial in  $n^d$  we verify whether each of the groups  $\text{Gal}(L_i/K_i)$  is in  $\Gamma_d$  (this is sufficient because of Proposition 7.20). Algorithm 12 is the complete algorithm.

**Input:** An irreducible polynomial  $f(X)$  of degree  $n$  over  $\mathbb{Q}$ .

**Result:** *Accept* if the Galois group of  $f(X)$  is in  $\Gamma_d$ , *Reject* otherwise. Let  $G$  be the Galois group of  $f(X)$  as a permutation group on  $\Omega$ , the roots of  $f(X)$ .

Using Theorem 7.3 compute the fields  $K_i = \mathbb{Q}_{\Delta_i}$  for a maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m = \Omega$ .

**foreach**  $1 \leq i \leq m$  **do**

**if**  $[L_i : K_i] > b(n)$  **then Reject.**

**else if**  $\text{Gal}(L_i/K_i)$  *is not in*  $\Gamma_d$  **then Reject.**

**end**

**Accept.**

**Algorithm 12:**  $\Gamma_d$ -testing

The main theorem of this section follows.

**Theorem 7.22.** *Given a polynomial  $f(X) \in \mathbb{Q}[X]$ , there is an algorithm running in time polynomial in size  $(f)$  and  $n^{O(d)}$  that decides whether the Galois group of  $f$  is in  $\Gamma_d$ .*

For any  $1 \leq i \leq m$ , we have  $\#G = [\mathbb{Q}_f : \mathbb{Q}] = [\mathbb{Q}_f : L_i] \cdot [L_i : K_i] \cdot [K_i : \mathbb{Q}]$ . Therefore any prime factor of  $[L_i : K_i]$  divides  $\#G$ . Conversely  $G^{\Delta_i}/G^{\Delta_{i-1}}$  is a subgroup of  $l_i$ -fold product of  $\frac{G_{\Delta_i}}{G(\Delta_i/\Delta_{i-1})}$  (Theorem 3.12) for some integer  $l_i \geq 0$ . However by Proposition 7.20  $\frac{G_{\Delta_i}}{G(\Delta_i/\Delta_{i-1})} = \text{Gal}(L_i/K_i)$ . It follows that any prime factor of  $\#G$  is a prime factor of  $[L_i : K_i]$  for some  $1 \leq i \leq m$ . Therefore the set of primes dividing  $\#G$  is exactly the set  $\{p \mid p \text{ prime and } \exists 1 \leq i \leq m \text{ } p \text{ divides } [L_i : K_i]\}$ . If the Galois group  $G$  is in  $\Gamma_d$ , in time polynomial in size  $(f)$  and  $n^d$  we can compute the fields  $L_i$  and  $K_i$ . As a result we have the following theorem.

**Theorem 7.23.** *Given  $f(X) \in \mathbb{Q}[X]$  with Galois group in  $\Gamma_d$  there is an algorithm running in time polynomial in size( $f$ ) and  $n^d$  that computes all the prime factors of  $\#\text{Gal}(f)$ .*

## 7.4 Discussion

We saw that even though computing the Galois group of a polynomial is hard, certain properties of Galois groups can be efficiently tested. Landau and Miller showed that solvability is one such property. We have added nilpotence testing and  $\Gamma_d$  testing to this list. A group being solvable is in some sense a “local property”. The solvability of a group  $G$  can be established by looking at the composition series of  $G$ . The composition series considered for  $G$  was  $G = G^{\Delta_m} \triangleright \dots \triangleright G^{\Delta_0} = 1$  for a maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m = \Omega$ . The two-way Galois correspondence of Theorem 7.1 and Theorem 3.12 ensured that it was sufficient to compute the fields  $\{\mathbb{Q}_{\Delta_i}\}_{0 \leq i \leq m}$ , to infer the solvability of  $G^{\Delta_i}/G^{\Delta_{i-1}}$  and hence  $G$ . Nilpotence testing cannot be inferred from the composition series. However Theorem 7.14 together with Theorem 7.1 ensured that the nilpotence of  $G$  can be tested once the tower of fields  $\{\mathbb{Q}_{\Delta_i}\}_{0 \leq i \leq m}$  for a suitable maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subset \dots \subset \Delta_m$  is computed. In this context an interesting open problem is to test whether the Galois group of a polynomial is supersolvable. A group  $G$  is *supersolvable* if there is a *normal series*  $G = G_0 \triangleright \dots \triangleright G_t = 1$  such that each of the quotient group  $G_i/G_{i+1}$  is cyclic. (see Chapter 10 of Hall’s book [30]). Super solvable groups are a proper subclass of solvable groups and contain nilpotent groups. However, it is not clear whether the Landau-Miller solvability test or our nilpotence test can be adapted to an efficient supersolvability test. Even conditional results, for example assuming the generalised Riemann hypothesis, would be interesting.

What about nilpotence and  $\Gamma_d$  testing  $\text{Gal}(f)$  for polynomials  $f(X)$  over a number field  $K$ ? It is not difficult to see that our algorithms can be generalised. This is because our test require only certain basic algorithms like factoring of univariate polynomials and gcd computations and efficient algorithms for these basic tasks over arbitrary number field are known.

## Chapter 8

# Chebotarev density theorem and Order finding

In this chapter we study the problem of finding the order of the Galois group of a degree  $n$  polynomial  $f(X) \in \mathbb{Q}[X]$  [7]. There is a polynomial time Turing reduction from computing the order to computing the Galois group because given a permutation group  $G \leq S_n$  via its generators, the order of  $G$  can be computed in time polynomial in  $n$  (Theorem 3.9). In this chapter we show some conditional results. Assuming the generalised Riemann hypothesis we show better upper bounds for computing the order than the direct exponential time algorithm that follows from Landau's algorithm (Theorem 6.10).

Assuming the generalised Riemann hypothesis, we prove that there is a polynomial time deterministic algorithm that makes one query to a #P oracle to compute the order of the Galois group  $\text{Gal}(f)$ . In particular, this shows that the order can be computed in PSPACE, which was not known before. Recall that computing the Galois group of a polynomial is not known to be in PSPACE as nothing better than the EXP upper bound is known. From the above result, by an application of Stockmeyer's result on approximating #P functions, we prove that there is a randomised algorithm with an NP oracle to approximate the order of the Galois group of  $f(X)$ .

Our next result is on computing the order of polynomials with Galois group in  $\Gamma_d$ . We give a polynomial time reduction from exact computation to approximate computation of order of  $\text{Gal}(f)$  for polynomials  $f(X)$  with Galois group in  $\Gamma_d$ . Therefore assuming the generalised Riemann hypothesis, there is a randomised algorithm with NP-oracle for computing the order of  $\text{Gal}(f)$  exactly for polynomials  $f(X)$  with Galois group in  $\Gamma_d$ .

We can assume that the given polynomial  $f(X)$  is a monic polynomial over  $\mathbb{Z}$ . Otherwise by clearing denominator we can assume that  $f(X) = a_0 + \dots + a_n X^n$ ,  $a_i \in \mathbb{Z}$ . Consider the polynomial  $g(X) = a_0 a_n^n + \dots + a_i a_n^{n-i} X + \dots + X^n$ . Clearly  $g(X)$  is a monic polynomial over  $\mathbb{Z}$ . Moreover  $g(a_n X) = a_n^n f(X)$ . Hence every root of  $g(X)$  is of the form  $a_n \alpha$  where  $\alpha$  is a root of  $f(X)$ . Therefore  $\mathbb{Q}_g = \mathbb{Q}_f$ . Given  $f(X)$  we can compute  $g(X)$  in polynomial time and hence from now on, with out loss of generality, we will assume that the input polynomial  $f(X)$  is a monic polynomial of  $\mathbb{Z}$ .

The main idea underlying these results is the following: For a positive integer  $x$  let  $S^f(x)$  denote the number of primes  $p \leq x$  such that  $f(X) \pmod{p}$  splits completely over  $\mathbb{F}_p$ . It follows from the Chebotarev density theorem, which we describe in Section 8.1, that  $S^f(x)$  is asymptotically  $\frac{x}{\#G \ln x}$ . Thus for large enough  $x$ ,  $\#G$  is close to  $\frac{x}{S^f(x) \ln x}$ . We prove that the function  $x \mapsto S^f(x)$  is a  $\#P$  function. The polynomial time algorithm makes a query to and  $\#P$  oracle and computes  $S^f(x)$ . The effective version of Chebotarev density theorem guarantees that the order  $\#G$  is then the nearest integer to  $\frac{x}{S^f(x) \ln x}$ . We now describe the Chebotarev density theorem which plays a crucial role in our complexity theoretic results.

## 8.1 Chebotarev density theorem

Let  $K$  be any number field and  $L/K$  be an extension of  $K$ . Recall that the ring of integers of  $L$ ,  $\mathbb{O}_L$ , is a Dedekind domain and ideals of  $\mathbb{O}_L$  has the unique factorisation property. Let  $\mathfrak{p}$  be a prime ideal of  $\mathbb{O}_K$ . The ideal  $\mathfrak{p}\mathbb{O}_L$ , which will also be denoted by  $\mathfrak{p}$ , need not be a prime ideal of  $\mathbb{O}_L$ . Let  $\mathfrak{p}$  factorise as  $\mathfrak{p} = \mathfrak{P}_1^{e_1} \dots \mathfrak{P}_g^{e_g}$  over  $\mathbb{O}_L$ . If  $L/K$  is a Galois extension then all the exponent  $e_i$  are the same, i.e.  $e_1 = \dots = e_g = e$ . A prime ideal  $\mathfrak{p}$  of  $\mathbb{O}_K$  is *ramified* over the extension  $L/K$  if  $e > 1$  and *unramified* otherwise.

We now consider Galois extensions  $L/K$ . Let  $G$  be the Galois group of  $L/K$ . Consider a prime  $\mathfrak{p}$  of  $\mathbb{O}_K$  that is unramified in  $L$ . Let  $\mathfrak{P}$  be any prime ideal of  $\mathbb{O}_L$  that divides  $\mathfrak{p}$ . Since  $\mathbb{O}_L$  and  $\mathbb{O}_K$  are Dedekind domains it follows that  $\mathbb{O}_L/\mathfrak{P}$  and  $\mathbb{O}_K/\mathfrak{p}$  are finite fields of cardinality  $N(\mathfrak{P})$  and  $N(\mathfrak{p})$  respectively. Furthermore the field  $\mathbb{O}_L/\mathfrak{P}$  is an extension of  $\mathbb{O}_K/\mathfrak{p}$  and the corresponding Frobenius element is given by  $\alpha \pmod{\mathfrak{P}} \mapsto \alpha^{N(\mathfrak{p})} \pmod{\mathfrak{P}}$ . For  $\mathfrak{P} | \mathfrak{p}$  there is an element  $\left(\frac{L/K}{\mathfrak{P}}\right)$  of  $\text{Gal}(L/K)$  such that

$$\left(\frac{L/K}{\mathfrak{P}}\right) \alpha = \alpha^{N(\mathfrak{p})} \pmod{\mathfrak{P}},$$

for all  $\alpha$  in  $\mathbb{O}_L$ . This element is called the *Frobenius* element associated with

$\mathfrak{P}$  as its action modulo  $\mathfrak{P}$  matches with the Frobenius element of the finite field extension  $\frac{\mathbb{O}_L}{\mathfrak{P}}/\frac{\mathbb{O}_K}{\mathfrak{p}}$ .

Let  $\mathfrak{P}_1, \dots, \mathfrak{P}_g$  be the primes of  $\mathbb{O}_L$  that divide  $\mathfrak{p}$ . The Galois group  $\text{Gal}(L/K)$  fixes the ideal  $\mathfrak{p}$  and act transitively on the set  $\{\mathfrak{P}_1, \dots, \mathfrak{P}_g\}$ . In particular, if  $\sigma \in \text{Gal}(L/K)$  maps  $\mathfrak{P}_1$  to  $\mathfrak{P}_2$  then  $\left(\frac{L/K}{\mathfrak{P}_2}\right) = \sigma\left(\frac{L/K}{\mathfrak{P}_1}\right)\sigma^{-1}$ . Thus  $\left(\frac{L/K}{\mathfrak{P}_i}\right)$  are all conjugates in  $\text{Gal}(L/K)$  and the subset  $\text{Frob}_{L/K}(\mathfrak{p})$  of  $\text{Gal}(L/K)$  defined by

$$\text{Frob}_{L/K}(\mathfrak{p}) = \left\{ \left( \frac{L/K}{\mathfrak{P}} \right) : \mathfrak{P}|\mathfrak{p} \right\}$$

is a conjugacy class of  $\text{Gal}(L/K)$ . Let  $C$  be any conjugacy class of  $G$  and let  $\pi_C(x)$  denote the function

$$\pi_C(x) = \# \{ \mathfrak{p} : \text{Frob}_L(\mathfrak{p}) = C \text{ and } N(\mathfrak{p}) \leq x \}.$$

A remarkable result on the asymptotic value of  $\pi_C(x)$  is the *Chebotarev density theorem* which states that  $\pi_C(x) \sim \frac{\#C}{\#G} \frac{x}{\ln x}$ . To apply this result in a complexity-theoretic setting we need the following effective version of the Chebotarev density theorem due to Lagarias and Odlyzko proved assuming the generalised Riemann Hypothesis [35].

**Theorem 8.1** (Lagarias and Odlyzko). *Let  $L/K$  be a Galois extension and  $C$  be any conjugacy class of  $\text{Gal}(L/K)$ . Assuming the generalised Riemann hypothesis we have the following bound for  $\pi_C(x)$ :*

$$\left| \pi_C(x) - \frac{\#C}{\#G} \frac{x}{\ln x} \right| \leq O(\sqrt{x} \cdot \ln x \cdot \ln d_L + \#C\sqrt{x}).$$

An unramified prime ideal  $\mathfrak{p}$  of  $K$  is said to be *completely split* if the number of prime ideals  $\mathfrak{P}$  of  $L$  that divide  $\mathfrak{p}$  is  $[L : K]$ . In this case  $\text{Frob}_L(\mathfrak{p})$  is the singleton conjugacy class containing the identity element. The number of completely split primes  $\mathfrak{p}$  such that  $N(\mathfrak{p}) \leq x$  is denoted by  $\pi_1(x)$ . A direct consequence of Theorem 8.1 is the following.

**Proposition 8.2.** *Assuming the generalised Riemann Hypothesis we have*

$$\left| \pi_1(x) - \frac{1}{\#G} \frac{x}{\ln x} \right| \leq O(\sqrt{x} \cdot \ln x \cdot \ln d_L).$$

We are given a monic polynomial  $f(X)$  over  $\mathbb{Z}$ . For an integer  $x$ , using the Chebotarev density theorem, we estimate the number of primes  $p \leq x$  for which  $f(X) \pmod{p}$  splits completely over  $\mathbb{F}_p$ .

**Theorem 8.3.** *Given a monic polynomial  $f(X)$  over  $\mathbb{Z}$  with Galois group  $G$  let  $S^f(x)$  denote the number of primes  $p \leq x$  such that  $f(X) \pmod{p}$  splits completely over  $\mathbb{F}_p$ . Assuming generalised Riemann hypothesis we have*

$$\left| S^f(x) - \frac{1}{\#G} \frac{x}{\ln x} \right| \leq O(\sqrt{x} \cdot \ln x \cdot (n!)^3 \cdot \text{size}(f)).$$

*Proof.* Let  $\mathcal{S}^f(x)$  denote the set of all primes  $p$  such that  $f(X) \pmod{p}$  splits completely over  $\mathbb{F}_p$ . Then  $S^f(x) = \#\mathcal{S}^f(x)$ . Let  $L$  be the splitting field  $\mathbb{Q}_f$ . Roots of  $f(X)$  are algebraic integers and hence are contained in  $\mathbb{O}_L$ . Consider any prime  $\mathfrak{p}$  of  $\mathbb{O}_L$  and let  $p$  be the prime in  $\mathbb{Z}$  such that  $\mathfrak{p} \cap \mathbb{Z} = p\mathbb{Z}$ . Then for any root  $\alpha \in \mathbb{O}_L$  of  $f(X)$ ,  $\alpha \pmod{\mathfrak{p}}$  is a root of  $f(X) \pmod{p}$  in the finite field  $\mathbb{O}_L/\mathfrak{p}$ . Therefore  $\mathbb{O}_L/\mathfrak{p}$  is the splitting field of  $f(X) \pmod{p}$ . If  $p$  is unramified and splits completely over  $L$  then  $\mathbb{O}_L/\mathfrak{p} = \mathbb{F}_p$  for all  $\mathfrak{p} \mid p$  and hence  $f(X)$  splits completely over  $\mathbb{F}_p$ . Therefore all unramified primes  $p \leq x$  that split completely over  $L/\mathbb{Q}$  are contained in the set  $\mathcal{S}^f(x)$ .

We now prove that the number of primes  $p$  in  $\mathcal{S}^f(x)$  that are not completely split are  $\leq (n!)^3 \cdot \text{size}(f)$ . Let  $\alpha_1, \dots, \alpha_n$  denote the roots of  $f(X)$ . Then there exists an algebraic integer  $\theta = \sum c_i \alpha_i$  such that  $\theta$  is a primitive element of  $L$  and  $\lg d_\theta \leq (n!)^3 \text{size}(f)$  (Theorem 6.19). Let  $\mu_\theta(X)$  be the minimal polynomial of  $\theta$ . Since  $\theta = \sum c_i \alpha_i$ , for any  $p$  if  $f(X)$  splits completely over  $\mathbb{F}_p$  then so does  $\mu_\theta(X)$ . If in addition  $p$  does not divide the discriminant  $d_\theta$  then  $\mu_\theta(X)$  splits completely into distinct linear terms. It follows from the Kummer-Dedekind theorem (Theorem 6.3) that  $p$  is unramified and splits completely over  $L/\mathbb{Q}$ . Therefore the primes  $p \in \mathcal{S}^f(x)$  that are not completely split divide the discriminant  $d_\theta$ . The number of primes that divide  $d_\theta$  is bounded by  $\lg d_\theta \leq (n!)^3 \cdot \text{size}(f)$  (Theorem 6.19). Hence the number of primes in  $\mathcal{S}^f(x)$  that are not completely split over  $L/\mathbb{Q}$  is less than  $(n!)^3 \cdot \text{size}(f)$ .

We have thus proved that  $\pi_1(x) \leq \#\mathcal{S}^f(x) = S^f(x) \leq \pi_1(x) + \lg d_\theta$ . Also  $d_L \leq d_\theta$ . Thus

$$\begin{aligned} \left| S^f(x) - \frac{1}{\#G} \frac{x}{\ln x} \right| &\leq \left| S^f(x) - \pi_1(x) \right| + \left| \pi_1(x) - \frac{1}{\#G} \frac{x}{\ln x} \right| \\ &\leq O(\sqrt{x} \cdot \ln x \cdot (n!)^3 \cdot \text{size}(f)) \quad (\text{Proposition 8.2}). \end{aligned}$$

□

## 8.2 Computing the order of the Galois group

In this section we prove our first result on order computation. We are given a monic polynomial  $f(X)$  over  $\mathbb{Z}$ . As in the previous section let  $S^f(x)$  denote the number primes  $p \leq x$  such that  $f(X)$  splits completely over  $\mathbb{F}_p$ .

**Proposition 8.4.** *Assuming generalised Riemann hypothesis there exists a constant  $c$  such that for  $x \geq c \cdot (n!)^{10} \text{size}(f)^{2k}$*

$$\left| \#G - \frac{1}{S^f(x)} \frac{x}{\ln x} \right| \leq \frac{1}{n! \cdot \text{size}(f)^{k-1}}.$$

Therefore if  $x \geq c(n!)^{10} \text{size}(f)^2$  and  $n \geq 2$ , the integer closest to  $\frac{1}{S^f(x)} \frac{x}{\ln x}$  is  $\#G$ .

*Proof.* Let  $N(x) = \frac{1}{S^f(x)} \frac{x}{\ln x}$ . From Theorem 8.3 we have

$$(1 - \varepsilon(x)) \frac{1}{\#G} \frac{x}{\ln x} \leq S^f(x) \leq (1 + \varepsilon(x)) \frac{1}{\#G} \frac{x}{\ln x}$$

where  $\varepsilon(x)$  is  $O\left(\frac{\ln^2 x \cdot n!^3 \cdot \text{size}(f)}{\sqrt{x}}\right)$ . Therefore  $(1 - \varepsilon(x))N(x) \leq \#G \leq (1 + \varepsilon(x))N(x)$ . It follows that  $N(x) \leq \frac{\#G}{1 - \varepsilon(x)} \leq \frac{n!}{1 - \varepsilon(x)}$ . For  $x = \Omega(n!^6 \cdot \text{size}(f)^2)$ ,  $\frac{1}{1 - \varepsilon(x)} \leq 1 + 2\varepsilon(x)$ . Therefore

$$\left| \#G - \frac{1}{S^f(x)} \frac{x}{\ln x} \right| = |\#G - N(x)| \leq n! \varepsilon(x) (1 + 2\varepsilon(x)).$$

There is a constant  $c$  such that for  $x \geq c \cdot n!^{10} \text{size}(f)^{2k}$ ,  $\varepsilon(x) \leq \frac{1}{4n!^2 \cdot \text{size}(f)^{k-1}}$ .

It follows that for  $x \geq c \cdot n!^{10} \text{size}(f)^{2k}$ ,  $\left| \#G - \frac{1}{S^f(x)} \frac{x}{\ln x} \right|$  is bounded by  $\frac{1}{n! \cdot \text{size}(f)^{k-1}}$ .  $\square$

Consider the machine  $M$  that on input  $\langle f(X), x \rangle$  guesses a prime  $p \leq x$  and checks whether  $f(X)$  splits over  $\mathbb{F}_p$ . Since in time polynomial in  $\text{size}(f)$  and  $\text{size}(p)$  one can verify whether  $f(X)$  completely splits over  $\mathbb{F}_p$  (Theorem 6.9),  $M$  is an NP machine. The function  $S^f(x)$  is the number of accepting paths of  $M$  on input  $\langle f(X), x \rangle$  and therefore is in  $\#P$ .

**Proposition 8.5.** *The function  $\langle f, x \rangle \mapsto S^f(x)$  is in  $\#P$ .*

We now give the  $\text{FP}^{\#\text{P}}$  machine  $M$  to compute the order of the Galois group. Given the polynomial  $f(X)$  the machine  $M$  makes a single query to the  $\#\text{P}$  function of Proposition 8.5 and computes  $S^f(x)$  for  $x = c \cdot (n!)^{10} \cdot \text{size}(f)^2$ , where  $c$  is the constant of Proposition 8.4. Having computed  $S^f(x)$  the machine  $M$  in polynomial time finds the integer  $N$  closest to  $\frac{1}{S^f(x)} \frac{x}{\ln x}$ . It follows from Proposition 8.4 that  $N$  is the order of  $\text{Gal}(f)$ . Thus we have the following theorem.

**Theorem 8.6.** *Given a polynomial  $f(X)$  over  $\mathbb{Q}$ , assuming the generalised Riemann hypothesis there is a polynomial time deterministic algorithm with a  $\#\text{P}$  oracle that computes the order of  $\text{Gal}(\mathbb{Q}_f/\mathbb{Q})$ .*

For an arbitrary function in  $\#\text{P}$ , Stockmeyer proved the following theorem [64].

**Theorem 8.7** (Stockmeyer). *For every function  $F$  in  $\#\text{P}$  and any fixed constant  $c$  there is a randomised polynomial time algorithm with NP oracle that on input string  $x$  of length  $n$  computes a value  $N_x$  such that*

$$\left(1 - \frac{1}{n^c}\right) N_x \leq F(x) \leq \left(1 + \frac{1}{n^c}\right) N_x.$$

Using the above theorem we show that there is a randomised polynomial time algorithm with NP oracle to approximate the order of the Galois group.

**Theorem 8.8.** *Given a polynomial  $f(X)$  over  $\mathbb{Q}$  there is a randomised algorithm with an NP oracle that runs in time polynomial in  $\text{size}(f)$  and approximates the order of the Galois group of  $f$  with a error of at most  $\frac{1}{\text{size}(f)^{O(1)}}$ .*

*Proof.* Since  $S^f(x)$  is in  $\#\text{P}$ , using the randomised procedure of Theorem 8.7, for any constant  $k$ , we can compute a  $\frac{1}{\text{size}(f)^k}$ -approximation  $\tilde{S}^f(x)$  of  $S^f(x)$ , i.e. compute  $\tilde{S}^f(x)$  such that  $(1 - \varepsilon)\tilde{S}^f(x) \leq S^f(x) \leq (1 + \varepsilon)\tilde{S}^f(x)$  where  $\varepsilon = \frac{1}{\text{size}(f)^k}$ . Therefore we have

$$\frac{1 - \varepsilon}{S^f(x)} \leq \frac{1}{\tilde{S}^f(x)} \leq \frac{1 + \varepsilon}{S^f(x)}.$$

By Proposition 8.4 there is a constant  $c$  such that for  $x \geq c \cdot n!^{10} \text{size}(f)^{2(k+1)}$   $\left| \frac{1}{S^f(x)} \frac{x}{\ln x} - \#G \right|$  is bounded by  $\frac{1}{n! \cdot \text{size}(f)^k}$ . Choosing  $x = c \cdot n!^{10} \text{size}(f)^{2(k+1)}$

we have

$$\left. \begin{aligned} \left| \frac{1}{S^f(x)} \frac{x}{\ln x} - \#G \right| &\leq \left| \frac{1}{S^f(x)} \frac{x}{\ln x} - \#G \right| + \varepsilon \cdot \frac{1}{S^f(x)} \frac{x}{\ln x} \\ &\leq \frac{2}{n! \cdot \text{size}(f)^k} + \#G \frac{1}{\text{size}(f)^k} \\ &\leq \#G \frac{2}{\text{size}(f)^k}. \end{aligned} \right\} \quad (8.1)$$

The above inequality proves that the integer closest to  $\frac{1}{S^f(x)} \frac{x}{\ln x}$  is a  $\frac{2}{\text{size}(f)^k}$ -approximation of  $\#G$ .

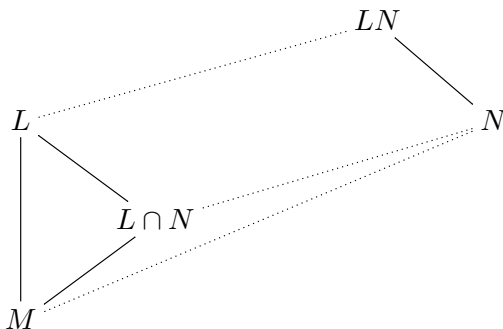
We now give the randomised algorithm with NP-oracle to compute an  $\frac{1}{\text{size}(f)^k}$  approximation of  $\#G$ . For  $x = c \cdot n!^{10} \text{size}(f)^{2(k+1)}$ , using Theorem 8.7, the algorithm first computes the approximation  $\tilde{S}^f(x)$  of the #P function  $S^f(x)$ . Then compute the integer  $N$  closest to  $\frac{1}{\tilde{S}^f(x)} \frac{x}{\ln x}$ . It follows from inequality 8.1 that  $N$  is a  $\frac{2}{\text{size}(f)^k}$ -approximation of  $\#G$ .  $\square$

### 8.3 Computing the order of Galois groups in $\Gamma_d$

Given a polynomial  $f(X)$  over  $\mathbb{Q}$  with  $\text{Gal}(\mathbb{Q}_f/\mathbb{Q})$  in  $\Gamma_d$ , in this section we show that  $\#\text{Gal}(\mathbb{Q}_f/\mathbb{Q})$  can be computed by a randomised polynomial-time algorithm with access to an NP oracle. The algorithm can be seen as a polynomial time Turing reduction from exact order finding to approximate order finding. The result then follows from Theorem 8.8.

First we state an important Lemma from Lang's book [40, Chapter VI, Theorem 1.12].

**Lemma 8.9.** *Let  $L/M$  be a Galois extension and let  $N$  be any field that contains  $M$ . Then  $LN/N$  is Galois and  $\text{Gal}(LN/N) \cong \text{Gal}(L/L \cap N)$ . Moreover the map that sends  $\tau \in \text{Gal}(LN/N)$  to its restriction on  $L$  is an isomorphism between the Galois groups  $\text{Gal}(LN/N)$  and  $\text{Gal}(L/L \cap N)$ .*



Using Lemma 8.9 we prove the following theorem on simple Galois extensions, i.e. Galois extensions  $L/K$  such that  $\text{Gal}(L/K)$  is simple.

**Theorem 8.10.** *Let  $L/M/N$  be finite extensions such that  $L/M$  is a simple Galois extension. Let  $E$  be a finite Galois extension of  $N$  containing  $M$  and let  $K$  be the normal closure of  $EL$  over  $N$ . Then  $[K : E] = [L : M]^l$  for some integer  $l \geq 0$ .*

*Proof.* Let  $L_1, \dots, L_r$  be the conjugate fields of  $L$  over  $N$ . Fix  $r$  automorphisms  $\{\sigma_i\}_{1 \leq i \leq r}$  of  $\text{Gal}(\overline{N}/N)$  such that  $L_i = \sigma_i(L)$  and let  $M_i = \sigma_i(M)$ .

First we prove that the Galois group  $\text{Gal}(K/E)$  embeds into the product group  $\prod_{i=1}^r \text{Gal}(L_i/M_i)$ . Let  $G_i$  be the Galois group  $\text{Gal}(L_i/M_i)$ . For any  $\tau \in \text{Gal}(K/E)$  let  $\tau_i$  denote the element of  $G_i$  obtained by restricting the action of  $\tau$  on  $L_i$ . The homomorphism  $\psi$  that maps  $\tau$  to  $\langle \tau_1, \dots, \tau_r \rangle$  is an embedding from  $\text{Gal}(K/E)$  to  $\prod_{i=1}^r G_i$ . This is because  $K$  is the field  $EL_1 \dots L_r$  and hence for any  $\tau \in \text{Gal}(K/E)$  if  $\tau_i$  fixes  $L_i$  for all  $1 \leq i \leq r$  then it fixes  $K$  as well.

Having proved that  $\text{Gal}(K/E)$  embeds into the product group  $\prod G_i$  we now prove that the degree  $[K : E]$  is a power of  $[L : M]$ . We dispose of the case when  $E$  contains one of the fields  $L_i$ . Since  $E$  is Galois over  $N$ , if  $E$  contains  $L_i$ , it contains all the other conjugate fields  $L_j$  and hence  $K = E$ . Therefore when  $E$  contains one of the  $L_i$ ,  $[K : E] = 1 = [L : M]^0$ .

We now consider the case when  $E$  contains none of the fields  $L_1, \dots, L_r$ . We prove that in this case the projection map  $\tau \mapsto \tau_i$  from  $\text{Gal}(K/E)$  to  $G_i$  is onto. It is sufficient to show that for all  $\sigma \in G_i = \text{Gal}(L_i/M_i)$  there is an automorphism  $\tau$  in  $\text{Gal}(K/E)$  such that  $\tau_i = \sigma$ ,  $1 \leq i \leq r$ .

Since  $K/E$  is Galois, every element  $\tau \in \text{Gal}(EL_i/E)$  can be extended to an element  $\tilde{\tau} \in \text{Gal}(K/E)$  such that  $\tilde{\tau}$  restricted to  $EL_i$  is  $\tau$  ([40, Theorem 2.8, Chapter V]). Therefore, it is sufficient to prove that for any element  $\sigma \in \text{Gal}(L_i/M_i)$  there is an element  $\tau_i \in \text{Gal}(EL_i/E)$  such that  $\tau_i$  restricted to  $L_i$  is  $\sigma$ .

Consider the extension  $EL_i/E$ . Since  $E$  is Galois and contains  $M$ ,  $E \supseteq M_i$ . By Lemma 8.9,  $\text{Gal}(EL_i/E)$  is isomorphic to  $\text{Gal}(L_i/L_i \cap E)$  via the map that send an automorphism  $\text{Gal}(EL_i/E)$  to its restriction on  $L_i$ . The extensions  $L_i/M_i$  and  $E/M_i$  are Galois and hence  $L_i \cap E/M_i$  is also Galois. Therefore  $\text{Gal}(L_i/L_i \cap E)$  is a normal subgroup of  $\text{Gal}(L_i/M_i)$  (Theorem 6.1). But  $\text{Gal}(L_i/M_i)$  is simple and  $L_i \cap E \neq L_i$ . Therefore  $L_i \cap E = M_i$  and hence  $\text{Gal}(EL_i/E) \cong \text{Gal}(L_i/M_i)$ .

For any  $\sigma \in \text{Gal}(L_i/M_i)$ , there is an element  $\sigma_i$  in  $\text{Gal}(EL_i/E)$  such that  $\sigma_i$  restricted to  $L_i$  is  $\sigma$ . Let  $\tau \in \text{Gal}(K/E)$  be any automorphism such

that  $\tau$  restricted to  $EL_i$  is  $\sigma_i$ . Then  $\tau_i = \sigma$ . As a result we have  $\text{Gal}(K/E)$  embeds *onto* the product group  $\prod_{i=1}^r \text{Gal}(L_i/M_i)$  via the map  $\tau \mapsto \tau_i$ .

If  $L_i/M_i$  is a simple abelian extension then  $\text{Gal}(L_i/M_i) \cong \mathbb{F}_p$  and therefore  $\text{Gal}(K/E)$  is isomorphic to a vector space over  $\mathbb{F}_p$ . Hence  $[K : E] = \#\text{Gal}(K/E) = p^l = [L : M]^l$  for some  $l$ . Otherwise if  $L_i/M_i$  is a nonabelian simple extension then using Scott's Lemma (Lemma 3.6),  $\text{Gal}(K/E)$  is a product of diagonal subgroups of  $\text{Gal}(L_i/M_i)$  and hence  $[K : E] = [L : M]^l$ .  $\square$

Given a polynomial  $f(X)$  over  $\mathbb{Q}$  of degree  $n$ . If the Galois group of  $f$  is in  $\Gamma_d$  then we show that the order of the Galois group of  $f$  can be computed by a randomised algorithm with an NP oracle. We first give a sketch of the algorithm here and defer the detailed description to Algorithm 13. For simplicity we assume that  $f(X)$  is irreducible. Algorithm 13 handles reducible  $f(X)$  as well.

For a number field  $K$  we denote the normal closure of  $K$  over  $\mathbb{Q}$  by  $\tilde{K}$ . Let  $G$  be the Galois group of  $f(X)$  thought of as a permutation group over  $\Omega$  the set of roots of  $f(X)$ . For a  $G$ -block  $\Delta$  recall that  $\mathbb{Q}_\Delta$  denotes the fixed field  $\text{Fix}(\mathbb{Q}_f, G_\Delta)$ . Using Theorem 7.3 repeatedly we can compute the number fields  $K_i = \mathbb{Q}_{\Delta_i}$  for maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subseteq \dots \subseteq \Delta_m = \Omega$ . Recall that the normal closure  $\tilde{K}_m$  and  $\tilde{K}_0$  are respectively  $\mathbb{Q}$  and  $\mathbb{Q}_f$  and  $\#\text{Gal}(f) = [\mathbb{Q}_f : \mathbb{Q}]$ . For  $i$  decreasing from  $m$  to 0 we compute the degree  $[\tilde{K}_i : \mathbb{Q}]$  inductively. To begin with the degree  $[\tilde{K}_m : \mathbb{Q}] = 1$ . Assuming we have computed the degree  $[\tilde{K}_i : \mathbb{Q}]$  we show how the degree  $[\tilde{K}_{i-1} : \mathbb{Q}]$  can be computed. Let  $L_i$  denote the normal closure of  $K_{i-1}$  over  $K_i$ . Recall that  $\tilde{L}_i = \tilde{K}_{i-1}$  and hence it is sufficient to compute the degree  $[\tilde{L}_i : \mathbb{Q}]$ . Recall that  $L_i/K_i$  is a Galois extension with small ( $\leq O(n^d)$ ) Galois group (Proposition 7.20). Hence we can compute the Galois group  $H = \text{Gal}(L_i/K_i)$  using Landau's algorithm. Furthermore in time polynomial in  $n^d$  we compute a composition series  $H = H_0 \triangleright \dots \triangleright H_t = 1$  for  $H$  where each of the quotient group  $H_i/H_{i+1}$  is simple. Let  $F_i$  denote the fixed field  $\text{Fix}(L_i, H_i)$ . Consider the tower of extensions  $\tilde{K}_i = \tilde{F}_0 \subseteq \dots \subseteq \tilde{F}_t = \tilde{K}_{i-1}$ . We have the following proposition

**Proposition 8.11.** *The extension  $F_{j+1}/F_j$  is a simple Galois extension and the degree  $[\tilde{F}_{j+1} : \tilde{F}_j]$  is a power of the degree  $[F_{j+1} : F_j]$ .*

*Proof.* The group  $H_{j+1}$  is a normal subgroup of  $H_j$  such that  $H_j/H_{j+1}$  is simple. Hence by fundamental theorem of Galois theory, the extension  $F_{j+1}/F_j$  is a simple Galois extension. Using Theorem 8.10 for the field

extensions  $F_{j+1}/F_j/\mathbb{Q}$ , the degree  $[\tilde{F}_{j+1} : \tilde{F}_j]$  is a power of the degree  $[F_{j+1} : F_j]$ .  $\square$

We compute the degree  $[\tilde{F}_j : \mathbb{Q}]$  inductively for increasing  $j$ . To begin with  $[\tilde{F}_0 : \mathbb{Q}] = [\tilde{K}_i : \mathbb{Q}]$  which we have already computed. Assume that we already know the degree  $[\tilde{F}_j : \tilde{K}_i]$ . We can compute a primitive polynomial  $h(X)$  of  $F_{j+1}$  over  $\mathbb{Q}$  in time polynomial in size  $(f)$  and  $n^d$ . Using Theorem 8.8 for a suitable small  $\varepsilon$  (say  $\varepsilon = 0.1$ ) we compute an approximation  $A$  of the degree  $[\tilde{F}_{j+1} : \mathbb{Q}]$  such that  $(1 - \varepsilon)A \leq [\tilde{F}_{j+1} : \mathbb{Q}] \leq (1 + \varepsilon)A$ . We have already computed the degrees  $[\tilde{F}_j : \tilde{K}_i]$  and  $[\tilde{K}_i : \mathbb{Q}]$  and therefore can compute  $[F_i : \mathbb{Q}] = [\tilde{F}_j : \tilde{K}_i][\tilde{K}_i : \mathbb{Q}]$ . Therefore  $A' = \frac{A}{[\tilde{F}_j : \mathbb{Q}]}$  gives an  $\varepsilon$ -approximation of  $[\tilde{F}_{j+1} : \tilde{F}_j]$ .

Let  $r$  denote the degree  $[F_{j+1} : F_j]$  which we have already computed. Then by Proposition 8.11,  $[\tilde{F}_{j+1} : \tilde{F}_j]$  is a power of  $r$ . Let  $r^l$  be the power of  $r$  that is closest to  $A'$ . Since  $A'$  is an  $\varepsilon$ -approximation of  $[\tilde{F}_{j+1} : \tilde{F}_j]$ , if  $\varepsilon < 0.1$  then  $[\tilde{F}_{j+1} : \tilde{F}_j] = r^l$ .

Having computed  $A'$ ,  $r$  and  $[\tilde{F}_j : \mathbb{Q}]$ , it is easy to find  $[\tilde{F}_{j+1} : \tilde{F}_j]$  and thus  $[\tilde{F}_{j+1} : \mathbb{Q}]$ . This completes our description of the algorithm. Algorithm 13 is a detailed presentation.

Algorithm 13 can be seen as a polynomial time Turing reduction from exact order finding to approximate order finding. The step 1 can be seen as an oracle query to a function that gives an approximation of the order of the Galois group. We thus have the following theorem.

**Theorem 8.12.** *For polynomials  $f(X)$  with Galois group in  $\Gamma_d$  there is a polynomial time (polynomial in size  $(f)$  and  $n^d$ ) Turing reduction from exact order finding to approximate order finding. Hence there is a randomised algorithm with an NP-oracle to compute the order of  $\text{Gal}(f)$  for polynomials  $f(X)$  with Galois group in  $\Gamma_d$ .*

## 8.4 Discussion

In this section we proved upper bounds on order finding for Galois group assuming the generalised Riemann hypothesis. We proved that computing the order of the Galois group of a polynomial  $f(X)$  is in  $\text{FP}^{\#\text{P}}$ . In addition if the Galois group of  $f(X)$  is in  $\Gamma_d$ , a fact that can be checked efficiently using Theorem 7.22, then the order of  $\text{Gal}(f)$  can be computed within the polynomial hierarchy. We can prove similar results for  $f(X) \in K[X]$  where  $K$  is given via explicit data.

**Input:** A polynomial  $f(X)$ .  
**Output:** The order of  $\text{Gal}(f)$ .  
**if**  $f(X)$  *is a constant polynomial* **then return** 1  
Let  $f$  factorise as  $gh$  where  $g$  is an irreducible polynomial over  $\mathbb{Q}$ .  
Let  $G$  be the Galois group  $\text{Gal}(\mathbb{Q}_g/\mathbb{Q})$ .  
Using Theorem 7.3 compute the fields  $K_i = \mathbb{Q}_{\Delta_i}$  for a maximal chain of  $G$ -blocks  $\{\alpha\} = \Delta_0 \subseteq \dots \subseteq \Delta_m = \Omega$ .  
Recursively compute  $N_m = [\mathbb{Q}_h : \mathbb{Q}]$ .  $N_i$  will denote the degree  $[\mathbb{Q}_h \tilde{K}_i : \mathbb{Q}]$ .  
**for**  $i \leftarrow m$  **downto** 0 **do**  
    Compute the normal closure  $L_i$  of  $K_{i-1}$  over  $K_i$ .  
    Compute  $H = \text{Gal}(L_i/K_i)$ .  
    Compute a composition series  $H = H_0 \triangleright \dots \triangleright H_t$ .  
    **for**  $j \leftarrow 0$  **to**  $t$  **do**  
         $F_j \leftarrow \text{Fix}(L_i, H_j)$   
        Compute the primitive polynomial  $f_j(X)$  of  $F_j$   
    **end**  
     $M_0 \leftarrow 1$ ,  $M_j$  will be the degree  $[\mathbb{Q}_h \tilde{F}_j : \mathbb{Q}_h]$ .  
    **for**  $j \leftarrow 1$  **to**  $t$  **do**  
        1     Compute a 0.1-approximation  $A$  of  $\#\text{Gal}(hf_j)$ .  
        Let  $r = [F_j : F_{j-1}]$ .  
        Compute the power  $r^l$  closest to  $\frac{A}{M_{j-1}N_i}$ .  
         $M_j \leftarrow M_{j-1} \cdot r^l$ .  
    **end**  
     $N_{i-1} \leftarrow N_i \cdot M_t$   
**end**  
**return**  $N_0$ .

**Algorithm 13:** Computing order of Galois group in  $\Gamma_d$ .

An interesting open problem is to give nontrivial upper bound unconditionally. Another interesting problem is to give better upper bounds for order finding for special polynomials, like for example polynomials with solvable Galois groups. One way to achieve this is to give better upper bounds for approximating the order of the Galois group. Certain #P-complete functions like #DNF can be approximated efficiently (Chapter 11 of the book by Motwani and Raghavan [54] gives a detailed presentation of such #P complete problems). It would be interesting to know whether the number of completely split primes less than a given number  $x$  can be approximated efficiently in which case we would have efficient order finding algorithm for polynomials with Galois group in  $\Gamma_d$ .

At present computing the Galois group looks harder than computing the order. It would be interesting to know for example whether the Galois group can be computed in PSPACE. Even conditional results will be interesting. For polynomials with solvable Galois group are there better upper bounds ?

## Chapter 9

# Computing Galois groups

In this chapter we give some upper bounds on computing the Galois group of certain special polynomials. Our first result is a randomised algorithm to compute the Galois group of polynomials with abelian Galois group [7]. This result makes use of the effective version of the Chebotarev density theorem and hence is conditional on the validity of the generalised Riemann hypothesis. We then consider polynomials  $f(X)$  that are product of polynomials  $\{f_i\}_{1 \leq i \leq m}$  having the following properties (1)  $\mathbb{Q}_{f_i} = \mathbb{Q}[X]/f_i(X)$  and (2)  $\text{Gal}(f_i)$  is simple and nonabelian. We show that in this case there is deterministic algorithm that runs in time polynomial in size  $(f)$  to compute the Galois group of  $f$ . This result is unconditional and Scott's Lemma plays a crucial role in the proof of this result. In particular, for this result the assumption that  $\text{Gal}(f_i)$  is nonabelian is crucial as Scott's lemma is not true for abelian simple groups.

Recall that if  $f(X)$  is irreducible and has abelian Galois group then  $\text{Gal}(f)$  can be computed in polynomial time using Landau's algorithm (Theorem 6.12). However, when  $f(X)$  is reducible with abelian Galois group, the Galois group can be exponentially large. Hence Landau's algorithm cannot be used directly. In fact even when the polynomial is a product of quadratic polynomial nothing better than the exponential time algorithm is known (cf. Lenstra [44]).

For polynomials  $f(X)$  with abelian Galois group we give a polynomial time almost uniform sampling algorithm for elements of  $\text{Gal}(\mathbb{Q}_f/\mathbb{Q})$ . It is easy to see that for a group  $G$  a random sample of  $O(\lg G)$  elements from  $G$  is a generator set with high probability.

## 9.1 Computing abelian Galois groups

Given a polynomial  $f(X)$  with abelian Galois group. Our task is to compute the Galois group  $G$  of  $f(X)$ . Let  $f = f_1, \dots, f_r$  be the factorisation of  $f$  into irreducible factors. Let  $G_i$  be the Galois group of  $f_i$ . Each of the groups  $G_i$  can be computed explicitly using Landau's algorithm. The group  $G$  is a subgroup of the product group  $\prod_{i=1}^r G_i$  and projects onto each  $G_i$ , i.e.  $G$  embeds into the product group  $\prod G_i$ . Hence any  $\sigma \in G$  can be considered as a tuple  $\sigma = \langle \sigma_1, \dots, \sigma_r \rangle$  where  $\sigma_i \in G_i$ .

There are two important properties of abelian extensions that we require. Firstly, each conjugacy class of  $G$  is a singleton set. Secondly, by factoring each of the irreducible factors  $f_i$  over  $\mathbb{F}_p$  we can recover the Frobenius element associated to  $p$  (Proposition 9.2).

Let  $L$  denote the splitting field  $\mathbb{Q}_f$ . Recall that for each prime  $p$  we can associate a conjugacy class  $\text{Frob}_{L/\mathbb{Q}}(p)$  (see Section 8.1). Since  $G$  is abelian the conjugacy class  $\text{Frob}_{L/\mathbb{Q}}(p)$  is a singleton set  $\{\sigma_p\}$ . We show that for a given  $\sigma \in G$  the probability that  $\sigma_p = \sigma$  for a random prime is close to  $\frac{1}{\#G}$ . This follows from the Chebotarev density theorem. Hence picking primes  $p$  at random and recovering the corresponding Frobenius gives us an almost uniform sampler for elements of  $G$ . A polynomial size sample will then generate  $G$ .

Let  $p$  be any prime. To recover the Frobenius  $\sigma_p$ , we recover the corresponding Frobenius'  $\sigma_{p,i}$  of  $G_i$ . Then  $\sigma_p = \langle \sigma_{p,1}, \dots, \sigma_{p,r} \rangle$ . The following important property of polynomials with abelian Galois group is useful in recovering the Frobenius element  $\sigma_p$ .

**Lemma 9.1.** *Let  $g \in \mathbb{Q}[X]$  be an irreducible polynomial of degree  $d$  with abelian Galois group. Let  $\theta$  be any root of  $g$  and let  $g(X) = \prod_{i=1}^d (X - A_i(\theta))$  be the factorisation of  $g$  over  $\mathbb{Q}(\theta)$  where  $A_i(X)$  are polynomial over  $\mathbb{Q}$ . For any  $\sigma \in \text{Gal}(\mathbb{Q}_g/\mathbb{Q})$  there is a unique index  $i$  such that  $\sigma$  maps  $\eta$  to  $A_i(\eta)$  for any root  $\eta$  (not necessarily  $\theta$ ) of  $g$ .*

*Proof.* Let  $G$  be the Galois group of  $g(X)$ . Since  $g$  is irreducible and  $G$  is abelian,  $\mathbb{Q}_g = \mathbb{Q}(\theta)$  and there is a unique automorphism  $\sigma_i$  that maps  $\theta$  to  $A_i(\theta)$ . The automorphisms  $\{\sigma_i\}_{i=1}^d$  constitutes the group  $G$ . Consider any root  $\eta$  of  $g$ . Since  $G$  is transitive there is a  $\tau \in G$  such that  $\tau(\theta) = \eta$ . Now  $\sigma_i(\eta) = \sigma_i\tau(\theta) = \tau\sigma_i(\theta)$  since  $G$  is abelian. Therefore  $\sigma_i(\eta) = \tau(A_i(\theta)) = A_i(\tau(\theta)) = A_i(\eta)$ . Therefore  $\sigma_i$  maps  $\eta$  to  $A_i(\eta)$ .  $\square$

We now show that given a prime  $p$  that does not divide the discriminant  $d_f$ , the automorphism  $\sigma_p$  can be recovered efficiently.

**Proposition 9.2.** *Given a prime  $p$  that does not divide  $d_f$ , there is a randomised algorithm running in time polynomial in  $\text{size}(f)$  and  $\lg p$  that computes the Frobenius  $\sigma_p$  as an  $r$ -tuple  $\langle \sigma_{p,1}, \dots, \sigma_{p,r} \rangle$  where  $\sigma_{p,i} \in G_i$  is the Frobenius element corresponding to  $p$  for the extension  $\mathbb{Q}_{f_i}/\mathbb{Q}$ .*

*Proof.* Fix a root  $\theta_i$  of  $f_i(X)$  over the extension  $\mathbb{Q}[X]/f_i(X)$ . Let  $f_i(X)$  factorise as

$$f_i(X) = \prod_{j=1}^{n_i} (X - A_{ij}(\theta_i)).$$

Compute the Galois group  $G_i$  of  $f_i$  using Landau's algorithm. Let  $\sigma_{ij}$  denote the unique automorphism of  $G_i$  that maps  $\theta_i$  to  $A_{ij}(\theta_i)$ . Our task is to identify which of these is  $\sigma_{p,i}$ .

For each  $i$  we find the splitting field  $\mathbb{F}_{q_i}$  of  $f_i$  over  $\mathbb{F}_p$ . Since  $f_i$  is irreducible over  $\mathbb{Q}$  the order of the Frobenius  $\sigma_{p,i}$  divides  $n_i$ , the degree of  $f_i$ . Therefore  $[\mathbb{F}_{q_i} : \mathbb{F}_p]$  divides  $n_i$  and hence the splitting field is a small extension (of degree less than the degree of  $f$ ) over  $\mathbb{F}_p$ . Let  $\alpha$  be any root of  $f_i(X)$  in  $\mathbb{F}_{q_i}$ . In polynomial time find the index  $j$  such that  $\alpha^p = \tilde{A}_{ij}(\alpha)$  where  $\tilde{A}_{ij}(X)$  is the polynomial  $A_{ij}(X) \bmod p$ . Since  $p \nmid d_f$  the index  $j$  is unique as there are no multiple roots for  $f_i(X)$  over  $\mathbb{F}_p$ . The Frobenius  $\sigma_{p,i} = \sigma_{ij}$ .

Having computed  $\sigma_{p,i}$  for all  $1 \leq i \leq r$  we have  $\sigma_p = \langle \sigma_{p,1}, \dots, \sigma_{p,r} \rangle$ .  $\square$

For our almost uniform sampler we study the distribution of  $\sigma_p$  for random primes  $p$ . We show that for a random prime  $p$ , the distribution of  $\sigma_p$  is almost uniform over  $G$ .

**Proposition 9.3.** *Let  $\sigma$  be any automorphism in  $\text{Gal}(\mathbb{Q}_f/\mathbb{Q})$ . Let  $P_\sigma(x)$  denote the probability that for an unramified prime  $p \leq x$  picked uniformly at random  $\sigma_p = \sigma$ . Assuming the generalised Riemann hypothesis, there exists a constant  $c$  independent of  $f(X)$  such that*

$$\frac{1}{\#G} \left(1 - \frac{1}{n!}\right) \leq P_\sigma(x) \leq \frac{1}{\#G} \left(1 + \frac{1}{n!}\right)$$

for all  $x \geq c \cdot (n!)^{10} \cdot \text{size}(f)^2$ .

*Proof.* Let  $L$  be the splitting field  $\mathbb{Q}_f$ . For an automorphism  $\sigma \in G$  let  $\pi_\sigma(x)$  denote the number of unramified primes  $p \leq x$  such that  $\sigma_p = \sigma$ . By the effective version Chebotarev density theorem (Theorem 8.1) we have  $\left| \pi_\sigma(x) - \frac{1}{\#G} \frac{x}{\ln x} \right| \leq O(\sqrt{x} \cdot \ln x \cdot \ln d_L)$ . Recall that  $d_L \leq (n!)^3 \text{size}(f)$ . Also

by the prime number theorem, the number of primes less than  $x$  is given by  $\pi(x) = \frac{x}{\ln x}$ . Therefore  $P_\sigma(x) = \frac{\pi_\sigma(x)}{\pi(x)}$ . It follows that

$$\left| P_\sigma(x) - \frac{1}{\#G} \right| \leq O\left(\frac{\ln x^2 \cdot n!^3 \text{size}(f)}{\sqrt{x}}\right).$$

Therefore there is a constant  $c$  such that for  $x \geq c \cdot (n!)^{10} \cdot \text{size}(f)^2$

$$\frac{1}{\#G} \left(1 - \frac{1}{n!}\right) \leq P_\sigma(x) \leq \frac{1}{\#G} \left(1 + \frac{1}{n!}\right).$$

□

Proposition 9.3 shows that picking random primes and computing  $\sigma_p$  gives an almost uniform sampling procedure. That  $\sigma_p$  can be computed given  $p$  follows from Proposition 9.2. The only missing result is to show that a polynomial sized sample generates  $G$  which we do now.

**Lemma 9.4.** *Let  $G$  be any group. Consider a sampling procedure that produces each element  $g \in G$  with probability at least  $\frac{1}{\lambda \#G}$ , for some  $\lambda > 1$ . A sample set of size  $4 \cdot \lambda \cdot \lg \#G$  where each element is obtained by running the sampling procedure independently will generate  $G$  with probability at least  $\frac{1}{4\lambda}$ .*

*Proof.* Let  $N = 4 \cdot \lambda \cdot \lg \#G$  and let  $g_1, \dots, g_N$  be the group elements sampled by running the procedure  $N$  times. Let  $G_0 = \{1\}$  and let  $G_i$  denote the group generated by  $\{g_1, \dots, g_i\}$ . Define the random variable  $X_i$  as follows.

$$X_i = \begin{cases} 1 & \text{if } G_{i-1} \neq G \text{ and } g_i \in G_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

We have  $\text{Prob}[X_i = 1 \mid G_{i-1} = G] = 0$ . If  $G_{i-1} \neq G$  then  $\#G_{i-1} \leq \frac{1}{2} \#G$ . Therefore the probability  $\text{Prob}[g_i \notin G_{i-1} \mid G_{i-1} \neq G]$  is at least  $\frac{1}{2\lambda}$ . We now compute the expectation of the variable  $X_i$ .

$$\begin{aligned} \mathbf{E}[X_i] &= \text{Prob}[X_i = 1] \\ &= \text{Prob}[G_{i-1} \neq G \text{ and } g_i \in G_{i-1}] \\ &= \text{Prob}[g_i \in G_{i-1} \mid G_{i-1} \neq G] \cdot \text{Prob}[G_{i-1} \neq G] \\ &= 1 - \text{Prob}[g_i \notin G_{i-1} \mid G_{i-1} \neq G] \\ &\leq 1 - \frac{1}{2\lambda}. \end{aligned}$$

Let  $X$  be the random variable  $\sum_{i=1}^N X_i$ . The random variable  $X$  is always positive with expectation  $\mathbf{E}[X] = \sum \mathbf{E}[X_i] \leq N \cdot (1 - \frac{1}{2\lambda})$ . By Markov's inequality  $\text{Prob}[X \geq t] \leq \frac{\mathbf{E}[X]}{t}$  for all  $t$ . Using  $t = N - \lg \#G$  we have

$$\begin{aligned} \text{Prob}[X \geq N - \lg \#G] &\leq \frac{1 - \frac{1}{2\lambda}}{1 - \frac{1}{4\lambda}} \\ &\leq 1 - \frac{1}{4\lambda}. \end{aligned}$$

Consider any sample  $g_1, \dots, g_N$  such that random variable  $X$  is less than  $N - \lg \#G$ . Assume that the random group  $G_N$  generated by  $g_1, \dots, g_N$  is different from  $G$ . Then  $G_i \neq G$  for all  $1 \leq i \leq n$ . As a result there are at least  $\lceil \lg \#G \rceil$  different indices  $i$  such that  $g_i \notin G_{i-1}$ . At each such  $i$ ,  $\#G_i \geq 2\#G_{i-1}$ . Hence  $\#G_N \geq G$ . But  $G_i$ 's are all subgroup of  $G$ . This contradicts the assumption that  $G_N \neq G$ . Therefore if  $X \leq N - \lg \#G - 1$  then  $G_N = G$ . Thus

$$\begin{aligned} \text{Prob}[G_N = G] &\geq \text{Prob}[X < N - \lg \#G] \\ &= 1 - \text{Prob}[X \geq N - \lg \#G] \\ &\geq \frac{1}{4\lambda}. \end{aligned}$$

□

We are ready to give a randomised algorithm to compute the Galois group of  $f(X)$ . The idea is to pick a prime  $p \leq x$  for some sufficiently large  $x$  at random and recover  $\sigma_p$  using Proposition 9.2. It follows from Proposition 9.3 that if  $x \geq c \cdot (n!)^{10} \cdot \text{size}(f)^2$ , an element  $\sigma$  will be obtained by this sampling procedure with probability at least  $\frac{1}{2\#G}$ . Therefore an  $8 \lg \#G \leq 8n^2$  sized sample set will generate  $G$  with probability at least  $\frac{1}{8}$ . Algorithm 14 is the detailed presentation.

We now prove the main result of this section.

**Theorem 9.5.** *Given a polynomial  $f(X)$  over  $\mathbb{Q}$  of degree  $n$  with abelian Galois group. Assuming the generalised Riemann hypothesis there is a randomised algorithm that runs in time polynomial in  $\text{size}(f)$  and outputs a strong generator set for Galois group of  $f$  with probability  $1 - \frac{1}{2^{n \cdot \text{size}(f)}}$*

*Proof.* Algorithm 14 gives a generator set of  $G$  with probability at least  $\frac{1}{8}$ . To improve the probability we run Algorithm 14 independently  $s$  times to get subsets  $A_1, \dots, A_s$  each of size  $8n^2$ . Since  $A_i$ 's are picked independently

**Input:** A polynomial  $f(X)$  over  $\mathbb{Q}$ .  
**Output:** Galois group of  $f(X)$ .  
Factorise  $f$  into irreducible factors  $f_1, \dots, f_r$ .  
Let  $S \leftarrow \emptyset$   
**for**  $i = 1$  to  $8n^2$  **do**  
    Pick a prime  $p \leq c \cdot (n!)^{10} \text{size}(f)^2$  at random.  
    Recover the  $\sigma_p$  using Proposition 9.2  
     $S \leftarrow S \cup \{\sigma_p\}$   
**end**  
**return**  $S$ .

**Algorithm 14:** Computing abelian Galois group

at random, the probability that none of  $A_i$ 's generate  $G$  is at most  $(\frac{7}{8})^s$ . Hence  $A = \cup_{i=1}^k A_k$  is a generating set for  $G$  with at least  $1 - (\frac{7}{8})^s$ . Choosing  $s = \frac{n \cdot \text{size}(f)}{\lg 8 - \lg 7}$  we have the desired result. We can reduce the size of the set  $A$  to  $n^2$  by computing a strong generator set for  $G$ .  $\square$

## 9.2 Computing simple Galois groups

We consider an interesting special case of nonabelian Galois groups computation for which we have a polynomial-time algorithm. Let  $f(X)$  be a polynomial such that  $f(X)$  factors as  $f = \prod_{i=1}^r f_i(X)$  over  $\mathbb{Q}$ . Suppose the Galois group of  $f_i(X)$  is small (of order bounded by a polynomial in  $\text{size}(f)$ ), simple and nonabelian. Then there is a polynomial time algorithm to compute the Galois group of  $f$ .

Firstly using Landau's algorithm the groups  $G_i = \text{Gal}(K_{f_i}/K)$  can be computed in time polynomial in  $\text{size}(f)$  as  $G_i$  is of size bounded by a polynomial in  $\text{size}(f)$ . The Galois group  $G = \text{Gal}(\mathbb{Q}_f/\mathbb{Q})$  is a subgroup of  $\prod_{i=1}^r G_i$ . Moreover since the splitting field  $\mathbb{Q}_f$  contains the splitting field  $\mathbb{Q}_{f_i}$ , the projection from  $G$  to  $G_i$  is onto. Each of the groups  $G_i$  is simple and non-abelian. Therefore, by Scott's Lemma (Lemma 3.6), there is a partition on the set  $\{1, \dots, r\}$  into subsets  $I_1, \dots, I_s$  such that  $G$  is given by

$$G = \prod_{k=1}^s \text{Diag} \left( \prod_{j \in I_k} G_j \right).$$

As in Chapter 5 we say that  $i$  and  $j$  are *linked* if  $G_i$  and  $G_j$  belong to the same partition. In this case  $G$  projected to  $G_i \times G_j$  is the diagonal group. This implies that  $i$  and  $j$  are linked if and only if the splitting fields  $f_i$  and

$f_j$  are the same. This give a polynomial time algorithm to check whether  $i$  and  $j$  are linked: Compute the explicit data for the splitting field  $L_i = \mathbb{Q}_{f_i}$  and factorise  $f_j(X)$  over  $L_i$ . The indices  $i$  and  $j$  are linked if and only if  $f_j(X)$  splits completely over  $L_i$

The partitions  $I_1, \dots, I_s$  are the equivalence classes of the equivalence relation  $\sim$  defined by  $i \sim j$  if  $i$  linked to  $j$ . Since  $i \sim j$  can be checked in time polynomial in size  $(f)$  the equivalence classes  $\{I_k\}_{1 \leq k \leq s}$  can be computed in polynomial time. Putting it all together we have the following theorem.

**Theorem 9.6.** *Let  $f(X) \in \mathbb{Q}[X]$  be a polynomial such that  $f = f_1 f_2 \dots f_r$  where each  $f_i$  has a non-abelian simple Galois group of size at most  $N$ . Then there is an algorithm that runs in time polynomial in size  $(f)$  and  $N$  to compute the Galois group of  $f(X)$ . In particular if  $N$  is bounded by a polynomial in size  $(f)$ , there is a polynomial time algorithm for finding the Galois group of  $f$ .*

*Proof.* First factorise the polynomial  $f$  into  $f_1, f_2, \dots, f_n$ . Compute the Galois groups  $G_i = \text{Gal}(\mathbb{Q}_{f_i}/\mathbb{Q})$  for each  $1 \leq i \leq n$  in time polynomial in size  $(f)$  and  $N$  using Landau's algorithm (Theorem 6.10). As described before equivalence classes  $\{I_k\}_{1 \leq k \leq s}$  can be computed in time polynomial in size  $(f)$  and  $N$ . For  $i$  and  $j$  that are linked, in order to compute the diagonal group, we need to find the right isomorphism between  $G_i$  and  $G_j$ . This can be computed by factoring  $f_j$  over  $\mathbb{Q}_{f_i}$ . We then output the group

$$G = \prod_{k=1}^t \text{Diag} \left( \prod_{j \in I_k} G_j \right),$$

which is the required Galois group. □

### 9.3 Discussion

As all our results on computational Galois theory, we can prove similar results for polynomials  $f(X)$  over a number field  $K$  given by explicit data. It still remains open whether there is a polynomial time deterministic algorithm to compute the Galois group of a polynomial with abelian Galois group. Even when  $f(X)$  is a product of quadratic polynomials we do not have polynomial time deterministic algorithm.

For polynomials  $f(X)$  with abelian Galois group, each conjugacy class of  $G$  was singleton. Also any prime  $p$  that does not divide the discriminant  $d_f$ , using Lemma 9.1 we could recover the Frobenius associated to the prime  $p$ .

These two properties gave us the uniform sampling procedure. However if the Galois group of  $f(X)$  is not abelian we do not have a method to recover the action of the Frobenius. By factoring  $f(X)$  over  $\mathbb{F}_p$  for different primes we get only the cyclic structure of element of  $\text{Gal}(f)$ .

Finally, in the absence of any good algorithms, it is of interest to prove hardness results for Galois group computations.

# Bibliography

- [1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.
- [2] Ákos Seress. *Permutation group algorithms*. Number 152 in Cambridge Tracts in Mathematics. Cambridge University Press, 2003.
- [3] Eric Allender. Arithmetic circuits and counting complexity classes. *Quaderni di Matematica series*, 2004. To appear.
- [4] Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8:99–126, 1999.
- [5] Eric Allender, K. Reinhardt, and S. Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.
- [6] V. Arvind and Piyush P Kurur. Graph Isomorphism is in SPP. In 43<sup>rd</sup> *Annual Symposium of Foundations of Computer Science*, pages 743–750. IEEE, November 2002.
- [7] V. Arvind and Piyush P Kurur. Upper bounds on the complexity of some Galois theory problems. In 14<sup>th</sup> *International Symposium on Algorithms and Computation, ISAAC*, volume 2906 of *Lecture Notes in Computer Science*, pages 716–725. Springer, 2003.
- [8] V. Arvind, Piyush P Kurur, and T. C. Vijayaraghavan. Bounded color multiplicity Graph Isomorphism is in the  $\#L$  hierarchy. In *20th Conference on Computational Complexity (CCC 2005)*, pages 13–27. IEEE, June 2005.
- [9] Lázló Babai. Monte carlo algorithms in graph isomorphism testing, 1979. Universitat de Montreal Tech. Report D.M.S 79-10.

- [10] Lázló Babai. Bounded round interactive proofs in finite groups. *SIAM Journal of Discrete Mathematics*, pages 88–111, 1992.
- [11] Lázló Babai, Peter J. Cameron, and P. P. Pálffy. On the order of primitive groups with restricted nonabelian composition factors. *Journal of Algebra*, 79:161–168, 1982.
- [12] Lázló Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 171–183, 1983.
- [13] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*, volume 11 of *ETACS monographs on theoretical computer science*. Springer-Verlag, Berlin, 1988.
- [14] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity II*, volume 22 of *ETACS monographs on theoretical computer science*. Springer-Verlag, Berlin, 1990.
- [15] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.
- [16] E. R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967.
- [17] E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, July 1970.
- [18] K S Booth. Isomorphism testing for graphs, semigroups and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7:273–279, 1978.
- [19] R. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs, May 1987.
- [20] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace-MOD classes. *Mathematical Systems Theory*, 25(3):223–237, 1992.
- [21] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, April 1981.

- [22] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, Berlin, 1993.
- [23] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [24] John D. Dixon and Brian Mortimer. *Permutation Groups*. Number 163 in Graduate texts in mathematics. Springer-Verlag, 1991.
- [25] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [26] Stephen A. Fenner, Lance Fortnow, and Stuart A. Kurtz. Gap-definable counting classes. In *Structure in Complexity Theory Conference*, pages 30–42, 1991.
- [27] Lance J. Fortnow and S. Homer. A short history of computational complexity. *The History of Mathematical Logic*, 2003.
- [28] Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *IEEE Symposium on Foundations of Computer Science*, pages 36–41, 1980.
- [29] M. Garey and D. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.
- [30] Marshall Hall Jr. *The Theory of Groups*. The Macmillan Company, New York, first edition, 1959.
- [31] Richard Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, pages 85–104, 1972.
- [32] Johannes Köbler, Uwe Schöning, and Jacobo Torán. Graph isomorphism is low for PP. *Computational Complexity*, 2(4):301–330, 1992.
- [33] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser, 1993.
- [34] Richard E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.
- [35] J. C. Lagarias and A. M. Odlyzko. Effective versions of the Chebotarev density theorem. In A. Fröhlich, editor, *Algebraic Number Fields*, pages 409–464. Academic Press, London, 1977.

- [36] E. Landau. Sur quelques théorèmes de M. Petrovitch relatifs aux zéros des fonctions analytiques. *Bulletin de la Société de France*, 33:251–261, 1905.
- [37] Susan Landau. Polynomial time algorithms for Galois groups. In John Fitch, editor, *EUROSAM 84 Proceedings of International Symposium on Symbolic and Algebraic Computation*, volume 174 of *Lecture Notes in Computer Sciences*, pages 225–236. Springer, July 1984.
- [38] Susan Landau. Factoring polynomials over algebraic number fields. *SIAM Journal of Computing*, 14:184–195, 1985.
- [39] Susan Landau and Gary. L. Miller. Solvability by radicals is in polynomial time. *Journal of Computer and System Sciences*, 30:179–208, 1985.
- [40] Serge Lang. *Algebra*. Addison-Wesley Publishing Company, Inc, third edition, 1999.
- [41] Arjen K. Lenstra. Factoring polynomials over algebraic number fields. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, pages 245–254, March 1983.
- [42] Arjen K. Lenstra, Hendrik W. Lenstra Jr. and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
- [43] Hendrik W. Lenstra Jr. Finding isomorphisms between finite fields. *Mathematics of Computation*, 56(193):329–347, January 1991.
- [44] Hendrik W. Lenstra Jr. Algorithms in algebraic number theory. *Bulletin of the American Mathematical Society*, 26(2):211–244, April 1992.
- [45] Leonid A. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.
- [46] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.
- [47] Eugene M. Luks. Parallel algorithms for permutation groups and graph isomorphism. In *Proceedings of the IEEE Foundations of Computer Science*, pages 292–302. IEEE Computer Society, 1986.

- [48] Eugene M. Luks. Lectures on polynomial-time computation in groups. Technical Report NU-CCS-90-16, Northeastern University, 1990. <http://www.cs.uoregon.edu/~luks/northeasterncourse.pdf>.
- [49] Eugene M. Luks. Permutation groups and polynomial time computations. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.
- [50] Eugene M. Luks and Pierre McKenzie. Parallel algorithms for solvable permutation groups. *Journal of Computer and System Sciences*, 37(1):39–62, 1988.
- [51] Meena Mahajan and V. Vinay. Determinant: Old algorithms, new insights. *SIAM journal on Discrete Mathematics*, 12(4):474–490, 1999.
- [52] R Mathon. A note on graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–132, 15 March 1979.
- [53] Gary L. Miller. Graph isomorphism, general remarks. *Journal of Computer and System Sciences*, 18(2):128–142, 1979.
- [54] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, first edition, 1997.
- [55] Jürgen Neukirch. *Algebraic Number Theory*. Springer-Verlag, 1992.
- [56] P. P. Pálffy. A polynomial bound for the orders of primitive solvable groups. *Journal of Algebra*, pages 127–137, July 1982.
- [57] Omer Reingold. Undirected ST-connectivity in logspace. In *Proceedings of the thirty-seventh annual ACM Symposium on Theory of Computing, STOC*, pages 376–385. ACM, 2005.
- [58] W L Ruzzo, J Simon, and M Tompa. Space-bounded hierarchies and probabilistic computation. *Journal of Computer and System Sciences*, 28:216–230, 1984.
- [59] Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences.*, 37(3):312–323, 1988.
- [60] L. L. Scott. Representation in characteristic  $p$ . In *Santa Cruz Conference on Finite Groups*, pages 319–322. American Mathematical Society, 1980.

- [61] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [62] C. C. Sims. Computational methods in the study of permutation groups. *Computational problems in Abstract Algebra*, pages 169–183, 1970.
- [63] C. C. Sims. Some group theoretic algorithms. *Topics in Algebra*, 697:108–124, 1978.
- [64] L. Stockmeyer. On approximating algorithms for #P. *SIAM Journal of Computing*, 14:849–861, 1985.
- [65] L. J. Stockmeyer. The polynomial hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.
- [66] Seinosuke Toda. Counting problems computationally equivalent to the determinant. manuscript, 1991.
- [67] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.
- [68] Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal of Computing*, 33(5):1093–1108, 2004.
- [69] Leslie G. Valiant. Relative complexity of checking and evaluating. *Inf. Process. Lett.*, 5(1):20–23, 1976.
- [70] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
- [71] B. L. van der Waerden. *Algebra*, volume I. Springer-Verlag, seventh edition, 1991.
- [72] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of 6th Structure in Complexity Theory Conference*, pages 270–284, 1991.
- [73] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, first edition, 1999.
- [74] Helmut Wielandt. *Finite Permutation Groups*. Academic Press, New York, 1964.

- [75] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29:1426–1481, 1985.

# Index

- $\Gamma_d$ , 15
- algebraic, 74
- algebraic closure, 75
- algebraic integers, 77
- algebraic numbers, 77
- alphabet, 7
- automorphism
  - field automorphism, 75
  - of a graph, 25
- Babai-Cameron-Pálffy bound, 19
- block, 18
- block system, 19
- centraliser, 14
- characteristic subgroup, 38
- Chebotarev's theorem, 107–109
- colour class, 34
- coloured graph, 34
- composition series, 15
- conjugate, 75
  - blocks, 19
- decision problem, 7
- degree, 74
- diagonal subgroup, 16
- direct product, 15
- discriminant
  - of a number field, 77
  - of a polynomial, 87
- embedding
  - complex embedding, 77
  - real embedding, 77
- empty string, 7
- extension
  - Galois extension, 75
  - normal extension, 75
  - of a field, 74
  - separable extension, 75
- fixed field, 76
- Frobenius, 76, 107
- functional problems, 7
- Galois correspondence
  - of blocks, 19
  - of fields, 76
- Galois group, 75
- gap-definable, 11
- height, 77
- imprimitive, 19
- index
  - of a subgroup, 14
  - of blocks, 19
- irreducible polynomials, 74
- Kummer-Dedekind Theorem, 78
- language, 7
- length of a string, 7
- letters, 7
- locally residual series, 46
- low complexity class, 11

- lowness, 11
- maximal increasing chain, 19
- maximal subblock, 19
- minimal polynomial, 74
- nilpotent groups, 16
- norm, 78
- normal
  - series, 16
  - subgroup, 14
  - tower, 16
- normal closure, 14, 75
- number field, 77
- O’Nan-Scott theorem, 39
- orbit, 17
- Orbit-Stabiliser formula, 17
- primitive, 19
  - element, 75
  - polynomial, 75
- pullback, 15
- ramified prime, 107
- regular action, 17
- residual series, 42
- residue subgroup, 40
- Scott’s Lemma, 16
- semidirect product, 15
- semisimple, 16
  - series, 43
- sift, 43
- simple, 16
- socle, 39
- solvable group, 15
- splitting field, 75
- stabiliser
  - point-wise, 17
  - setwise, 17
- string, 7
- strong generator set, 18, 43–45
- structure forest, 22
- structure tree, 22
- subnormal
  - series, 15
  - subgroup, 15
  - tower of groups, 15
- supersolvable group, 105
- symmetric group, 17
- tower of groups, 15
- transitive, 18
- traversal, 18