On Methods of Computing Galois Groups and their Implementations in MAPLE

by

TANG KO CHEUNG, SIMON

(TangSimon@cuhk.edu.hk)

A Thesis

Submitted to the Graduate School of

The Chinese University of Hong Kong

(Division of Mathematics)

In partial fulfillment of the requirements for

the degree of Master of Philosophy in Mathematics

August 1998

The Chinese University of Hong Kong
Graduate School

The undersigned certify that we have read a thesis, entitled "On Methods of Computing Galois Groups and their Implementations in MAPLE" submitted to the Graduate School by *Tang Ko Cheung, Simon* in partial fulfillment of the requirements for the degree of *Master of Philosophy in Mathematics*. We recommend that it be accepted.

[illegible]

Chairman

[illegible]

[illegible]

External Examiner

Abstract

This work is an investigation of the mathematics, if any, and the computing knowledge involved in the determination of Galois group of polynomials over the rationals up to degree seven. We first describe a finite procedure in van der Waerden's 1949 Algebra Text to determine the Galois group of polynomials. It requires the construction and factorization of a degree $n!$ polynomial in $n+1$ variables and thus is not suitable for practical purposes. Two workable methods are then described.

Stauduhar's 1973 method for a given polynomial involves finding high-precision approximations to the roots of the polynomial, and fixing an ordering for these roots. The roots are then used to create (relative) resolvent polynomials of small degree, the linear factors of which determine new ordering for the roots. Sequences of these resolvents isolate the Galois group of the polynomial.

Soicher-McKay's 1985 method proceeds by efficiently determine sufficient properties, so as to specify the Galois group to within conjugacy in the symmetric group. This conjugation is realized by relabelling the zeros of the polynomial. The main tool discussed is the (absolute) resolvent polynomial. For Υ in $\mathbb{Z}[x_1, \ldots, x_n]$, the complete factorization of a resolvent polynomial is used to determine the orbit length partition of $\{\Upsilon(x_{1^\sigma}, \ldots, x_{n^\sigma}) : \sigma \text{ in } S_n\}$ under the action of the Galois group. An important class of resolvent polynomials considered are the linear resolvent polynomials. The use of linear resolvents in determining Galois group is discussed and a practical exact method of computing linear resolvents is described.

Roughly speaking, Stauduhar's method is numerical in nature while Soicher-McKay's method is symbolic. The two methods are compared and contrasted.

The success of Soicher-McKay's method depends on exact polynomial factorization algorithms that have developed since 1968. As an important and interesting topic in a newly emerging field called SAC—Symbolic and Algebraic Computation (Computer algebra or Computational Algebra), we also describe algorithms to factor polynomials over finite fields, over the rationals and over algebraic number fields.

A demonstration for an implementation of Stauduhar/Soicher-McKay's method by Tang Simon in MAPLE are then given. It takes two seconds to obtain an answer for the polynomials that we encounter on a Pentium-133.

Finally in an Appendix we argue that Galois Theory is not dead, there is now Computational Galois Theory— have developed rapidly recently and that basically is to blossom Old-fashioned Galois theory with a Computer. We also include some notes on the field of SAC. The advent of computing technology prompts a renewed interest in the Constructive School of Mathematics and that activity necessarily interweaves mathematics, complexity theory and software systems. It spurs new areas of research and revive languishing areas. The importance of SAC in applications has grown in recent years—the methods of Computer Algebra and the applications of computer algebra systems in technological areas related to information processing, software engineering, etc. in which the symbolic nature of the objects studied makes the techniques of calculus and numerical analysis inapplicable.

These views should not just be personal as they are supported by a overwhelming body of testimonials.

Acknowledgment

Thank YOU all!

Contents

1 Introduction **5**

1.1 Motivation 5

1.1.1 Calculation of the Galois group 5

1.1.2 Factorization of polynomials in a finite number of steps IS feasible 6

1.2 Table & Diagram of Transitive Groups up to Degree 7 8

1.3 Background and Notation 13

1.4 Content and Contribution of THIS thesis 17

2 Stauduhar's Method **20**

2.1 Overview & Restrictions 20

2.2 Representation of the Galois Group 21

2.3 Groups and Functions 22

2.4 Relative Resolvents 24

2.4.1 Computing Resolvents Numerically 24

2.4.2 Integer Roots of Resolvent Polynomials 25

2.5 The Determination of Galois Groups 26

2.5.1 Searching Procedures 26

2.5.2 Data: $\Upsilon(x_1, x_2, \dots, x_n)$, Coset Representatives & Searching Diagram 27

2.5.3 Examples 32

2.6 Quadratic Factors of Resolvents 35

2.7 Comment 35

3 Factoring Polynomials Quickly **37**

3.1 History 37

3.1.1 From Feasibility to Fast Algorithms 37

3.1.2 Implementations on Computer Algebra Systems 42

3.2 Squarefree factorization 44

3.3 Factorization over finite fields 47

3.4 Factorization over the integers 50

3.5 Factorization over algebraic extension fields 55

3.5.1 Reduction of the problem to the ground field 55

3.5.2 Computation of primitive elements for multiple field extensions 58

4 Solcher-McKay's Method **60**

4.1 Overview, Restrictions and Background 60

4.2 Determining cycle types in $Gal_{\mathbb{Q}}(f)$ 62

4.3 Absolute Resolvents 64

4.3.1 Construction of resolvent 64

4.3.2 Complete Factorization of Resolvent 65

4.4 Linear Resolvent Polynomials 67

4.4.1 r-sets and r-sequences 67

4.4.2 Data: Orbit-length Partitions 68

4.4.3 Constructing Linear Resolvents Symbolically 70

4.4.4 Examples 72

4.5 Further techniques 72

4.5.1 Quadratic Resolvents 73

4.5.2 Factorization over $\mathbb{Q}(\sqrt{\mathrm{disc}(f)})$ 73

4.6 Application to the Inverse Galois Problem 74

4.7 Comment 77

A Demonstration of the MAPLE program **78**

B Avenues for Further Exploration **84**

B.1 Computational Galois Theory 84

B.2 Notes on SAC—Symbolic and Algebraic Computation 88

Bibliography **97**

List of Tables

1.1 Transitive Groups up to degree 7 11

2.1 The function $\Upsilon(x_1, x_2, \ldots, x_n)$ and right coset representatives 29

4.1 Orbit-length Partitions of Sets and Sequences under G 69

4.2 Polynomials $f(x)$ such that $Gal_{\mathbb{Q}}(f) = G$ 76

List of Figures

1.1 Transitive subgroups Lattices of permutation groups up to degree 7 10

2.1 Staudubar's method of traversing in the Transitive subgroups Lattices 32

Chapter 1

Introduction

1.1 Motivation

The existence of an algorithm for the determination of Galois groups is nothing new; indeed, the original definition of the Galois group contained, at least implicitly, a technique for its determination, and this technique has been described explicitly by many authors, e.g., van der Waerden [GaloisBks].

1.1.1 Calculation of the Galois group

A method for actually forming the Galois Group of an equation $f(x) = 0$ relative to a field F is the following.

Let the roots of the equation be $\alpha_1, \ldots, \alpha_n$. By means of the indeterminates $u_1, \ldots, u_n$, form the expression $\theta = u_1\alpha_1 + \cdots + u_n\alpha_n$; perform on it all permutations $s_{\underline{u}}$ of the indeterminates $\underline{u}$ and form the product $\mathfrak{R}(z, \underline{u}) = \prod_s (z - \theta^{s_{\underline{u}}})$. Evidently this product is a symmetric function of the roots, and therefore, by the Fundamental Theorem of Symmetric Polynomials (theorem 5), it can be expressed in terms of the coefficients of $f(x)$. Now decompose $\mathfrak{R}(z, \underline{u})$ into irreducible factors in $F[z, \underline{u}]$: $\mathfrak{R}(z, \underline{u}) = \mathfrak{R}_1(z, \underline{u})\mathfrak{R}_2(z, \underline{u}) \cdots \mathfrak{R}_r(z, \underline{u})$.

Theorem 1 *The permutations $s_{\underline{u}}$ which carry any of the factors, say $\mathfrak{R}_1$, into itself form a group $\mathfrak{g}$. It is exactly the Galois group Γ of the given equation.*

Proof. After adjoining all roots, $\mathfrak{R}$ and therefore $\mathfrak{R}_1$ are decomposed into linear factors $z - \sum u_\nu \alpha_\nu$ with the roots α_ν as coefficients in any sequential order. We now affix subscripts to the roots in such fashion that $\mathfrak{R}_1$ contains the factor $z - (u_1\alpha_1 +$

$\cdots + u_n\alpha_n)$. By $s_{\underline{u}}$ we shall hereafter denote any permutation of the $\underline{u}$, and by $s_{\underline{\alpha}}$ the same permutation of the $\underline{\alpha}$. Then, obviously, the product $s_{\underline{\alpha}}s_{\underline{u}}$ leaves invariant the expression $\theta = u_1\alpha_1 + \cdots + u_n\alpha_n$; that is, we have $\theta^{s_{\underline{\alpha}}s_{\underline{u}}} = \theta, \theta^{s_{\underline{\alpha}}} = \theta^{s_{\underline{u}}^{-1}}$.
If $s_{\underline{u}}$ belongs to the group $\breve{g}$, that is, if it leaves $\mathfrak{R}_1$ invariant, then $s_{\underline{u}}$ transforms every linear factor of $\mathfrak{R}_1$, including the factor $z-\theta$, into a linear factor of $\mathfrak{R}_1$ again. If, conversely, a permutation $s_{\underline{u}}$ transforms the factor $z-\theta$ into another linear factor of $\mathfrak{R}_1$, it transforms $\mathfrak{R}_1$ into a polynomial which is irreducible in $F[z,\underline{u}]$ and which is a divisor of $\mathfrak{R}(z,\underline{u})$, and so it transforms $\mathfrak{R}_1$ into one of the polynomials $\mathfrak{R}_j$. This $\mathfrak{R}_j$ has a linear factor in common with $\mathfrak{R}_1$. Therefore the permutation necessarily transforms $\mathfrak{R}_1$ into itself, which means that $s_{\underline{u}}$ belongs to $\breve{g}$. Thus $\breve{g}$ consists of the permutations of the $\underline{u}$ which transform $z-\theta$ into a linear factor of $\mathfrak{R}_1$ again.

The permutations $s_{\underline{\alpha}}$ of the Galois group Γ of $f(x)$ are characterized by the property that they transform the quantity $\theta = u_1\alpha_1 + \cdots + u_n\alpha_n$ into its conjugates. This means: $s_{\underline{\alpha}}$ transforms θ into an element satisfying the same irreducible equation as θ; that is, $s_{\underline{\alpha}}$ carries the linear factor $z-\theta$ into another linear factor of $\mathfrak{R}_1$. Now $\theta^{s_{\underline{\alpha}}} = \theta^{s_{\underline{u}}^{-1}}$; hence $s_{\underline{u}}^{-1}$carries the linear factor $z-\theta$ again into a linear factor of $\mathfrak{R}_1$; that is $s_{\underline{u}}^{-1}$and so $s_{\underline{u}}$ belong to $\breve{g}$. The converse is also true. Thus the Galois Group Γ consists of exactly the same permutations as the group $\breve{g}$, except that they are performed on $\underline{\alpha}$ the instead of the $\underline{u}$. ■

These sources show that the problem of finding the Galois group of a polynomial $f(x)$ of degree n over a given field F can be reduced to the problem of factoring over F a polynomial of degree $n!$ whose coefficients are symmetric functions of the roots of $f(x)$.

In principle, therefore, whenever we have a factoring algorithm over F, we also have a Galois group algorithm.

1.1.2 Factorization of polynomials in a finite number of steps IS feasible

Let I be an integral domain, where the unique factorization theorem be valid for it. Let K be the field of fractions of I. By Gauß's lemma, we may assume the coefficients of any polynomials over K to be in I, and perform its factorization over I. To factorize a polynomial in $I[x_1, x_2, \ldots, x_n]$, by the method of induction on the variable n we shall now reduce everything to the following problem.

Theorem 2 *Let any factorization in I be performable in a finite number of steps: moreover, let there be only a finite number of units in I. There is a method of factoring every polynomial in $I[x]$ into prime factors.*

Proof. The solution is due to Kronecker [GaloisBks](4), cf. Chapter,section 3.1 in this thesis. Let $f(x)$ be a polynomial of degree n in $I[x]$. If $f(x)$ can be factored, then one of the factors is of degree $\leqslant n/2$; thus, if s is the greatest integer $\leqslant n/2$, we must investigate whether $f(x)$ has a factor $g(x)$ of degree $\leqslant n/2$.
We form the functional values $f(a_0), f(a_1), \ldots, f(a_s)$ for $s+1$ integral arguments $a_0, a_1, \ldots, a_s$. If $f(x)$ is to be divisible by $g(x)$, then $f(a_0)$ must be divisible by $g(a_0)$, and $f(a_1)$ by $g(a_1)$, and so on. However, every $f(a_i)$ in I possesses only a finite number of factors; therefore, for every $g(a_i)$ there are only a finite number of possibilities all of which may be found explicitly. For every possible combination of values $g(a_0), g(a_1), \ldots, g(a_s)$ there is, one and only one polynomial $g(x)$ which may be formed by Lagrange's or, more conveniently, Newton's interpolation formula. In this way a finite number of possible factors $g(x)$ are found.
Employing the division algorithm, we may now find out whether each of these polynomials $g(x)$ is actually a factor of $f(x)$. If, apart from the units, none of the possible $g(x)$ is a factor of $f(x)$, then $f(x)$ is irreducible; otherwise, a factorization has been found, and we may proceed to apply the same procedure to the two factors, and so forth. In this manner we finally arrive at the irreducible factors. ■

Remark[1]. In the integral case, $I = \mathbb{Z}$, the procedure may frequently be shortened considerably. By factoring the given polynomial *modulo* 2 and possibly *modulo* 3, we get an idea what degrees the possible factor polynomials $g(x)$ might have, and to what residue classes the coefficients *modulo* 2 and 3 might belong. This limits the number of the possible $g(x)$ considerably. Moreover, when applying Newton's interpolation formula, one should note that the last coefficient λ_s must be a factor of the highest coefficient of $f(x)$, which limits the number of possibilities still further.
Finally, it is an advantage to use more than $s+1$ points a_i (preferably $0, \pm 1, \pm 2$ and so on). For determining the possible $g(a_i)$ we use those $f(a_i)$ which contain the least number of prime factors; the other points may afterwards be used in order to limit the number of possibilities still further by examining each $g(x)$, and to see whether it assumes values which are factors of the respective $f(a_i)$ at all points a_i.

As Kronecker has described a factoring algorithm for polynomials with rational coefficients, the problem of determining the Galois groups of such polynomials is solved in principle. It is obvious, however, that a procedure which requires the factorization of a polynomial of degree $n!$ is not suited to the uses of mortal men.

[1] Please see Chapter 3—Factoring Polynomials Quickly.

In this thesis we describe two practical and relatively simple procedures which have been used to develop programs for polynomials over $\mathbb{Q}$ of degrees 3 through 7.

To end this section, we remark that the problem of determining the Galois group of a given $f \in F[x]$ was declared by Hans Zassenhaus[2] to be one of the 4 main problems in constructive number theory, [Poh87]— can demand particularly hard computations, and may as well be used to create hard examples for the subalgorithms used, [PZ89] and [Zas71].

1.2 Table & Diagram of Transitive Groups up to Degree 7

The two methods that we are going to describe depend on a knowledge of transitive groups. For degrees 3 to 7, the number of transitive groups *up to conjugacy* are $2, 5, 5, 16$ and 7 respectively, see Table 1.1 below. Figure 1.1 is the transitive subgroups lattice of permutation groups up to degree 7, [PZ89].

The notation for the group names is similar to that of [Mck79] and [SM85], who also gives group generators, cf. [Sta73]. A_n is the alternating group of degree n; S_n is the symmetric group of degree n; $\mathbb{Z}_n$ denotes the cyclic group of order n; V_4 is the four-group; D_n denotes the dihedral group of order $2n$; F_n denotes a Frobenius group of order n; G_n denotes a group of order n. If A and B are groups then A/B means that A is represented on the cosets of B in A. Groups preceded by "+" are groups of even permutations.

$\star \quad \star \quad \star$

[illegible]

The transitive groups for degrees 3 to 7 are relatively well-know, [Sta73]. But for higher degrees, to "Classify transitive subgroups of S_n up to conjugacy" is a non-trivial group-theoretical question, [Coh93], p.317. It has been solved up to $n = 11$, [BM83] and [MR85], in 1985 and up to $n = 15$, in 1993, see below. For higher degrees, the groups, ~~but~~ every just the number of groups, will become unwieldy. Recent work by Alexander Hulpke confirms these results and extends the tables up to degree 31, Appendix B.1[8], with the aid of the Computational Group Theory system GAP, [GAP].

While 31 is still a very small number, the reader should have no problem to comprehend that $31! = 8,222,838,654,177,922,817,725,562,880,000,000$ with any Computer Algebra System like [MAPLE].

[2]Prof. Zassenhaus passed away on November 21, 1991, around six 6 o'clock in the morning.

We include some findings of Alexander here:

degree	2	3	4	5	6	7
prim.	1	2	2	5	4	7
trans.	*1*	*2*	*5*	*5*	*16*	*7*
fpf.	2	2	7	8	37	40
Total	2	4	11	19	56	96

degree	8	9	10	11	12	13	14	15
prim.	7	11	9	8	6	9	4	6
trans.	*50*	*34*	*45*	*8*	*301*	*9*	*63*	*104*
fpf.	200	258	1038					
Total	296	554	1593					

degree	16	17	18	19	20	21	22	23
prim.	22	10	4	8	4	9	4	7
trans.	*1,954*	*10*	*983*	*8*	*1,117*	*164*	*59*	*7*

degree	24	25	26	27	28	29	30	31
prim.	5	28	7	15	14	8	4	12
trans.	*26,813*	*211*	*96*	*2,382*	*1,852*	*8*	*5,712*	*12*

All Numbers refer to classes up to S_n-conjugacy:
prim.: number of primitive groups (see Definition 30)
trans.: number of transitive groups (including primitive)
fpf.: number of fix point free groups (i.e. groups that are not subgroups of smaller symmetric groups.)
Total: total(n-1)+fpf(n)
The numbers for degree > 23 are still to be checked !
Back to my Homepage[3].

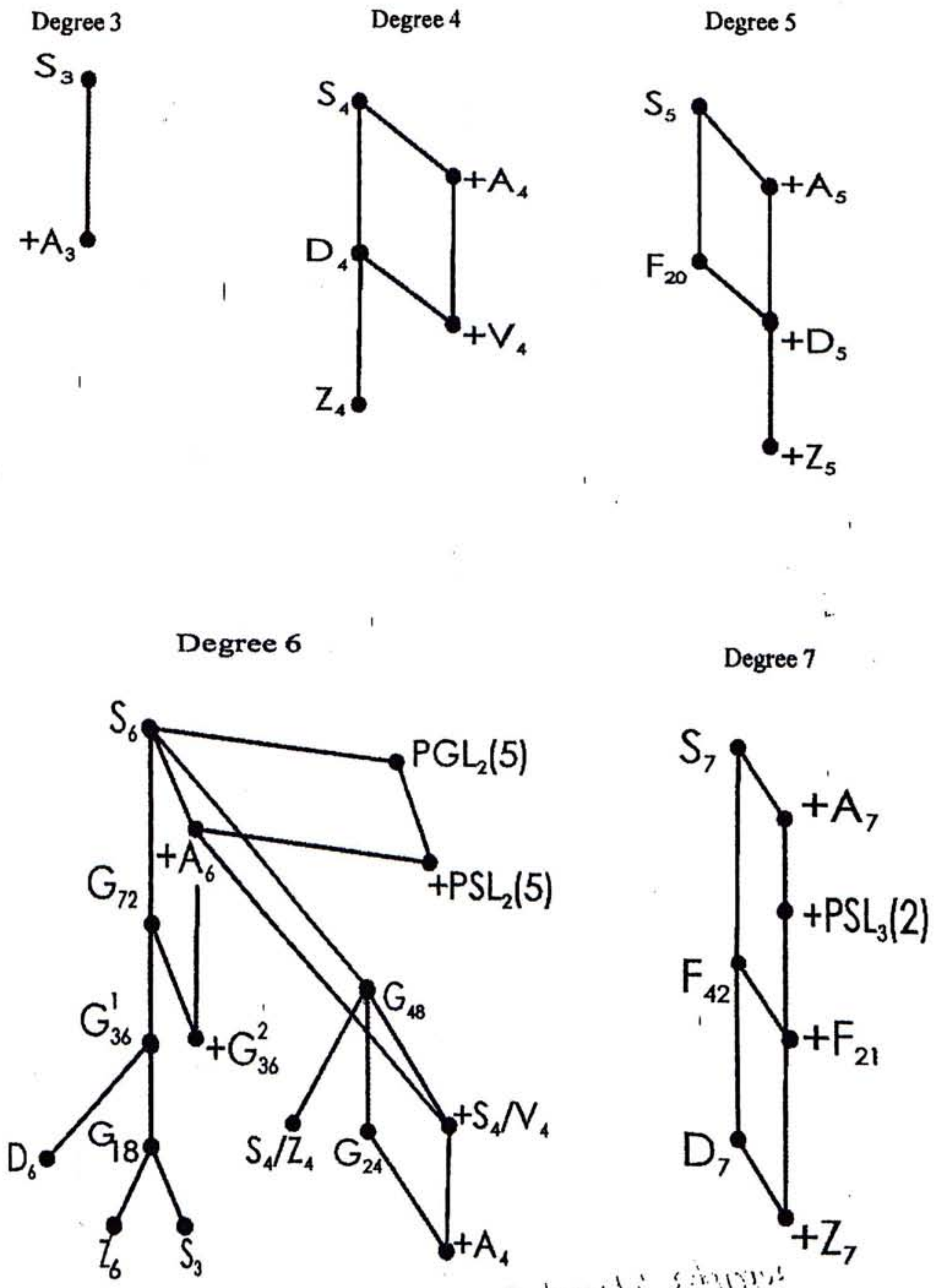
So we don't know whether the number of transitive subgroups of S_{24} is 26,813. Double Oh Eleven also told me that he found the following in one of Alexander's preprint:

> "... The classification of transitive permutation groups has been pursued for over a century since the Grand Prix of the Académie des Sciences in 1858 [*Grand prix de mathématiques*, C. R. Acad. Sci. Paris, 1858, 302-303]. An account of early work is given in [Bur1898] and [Mil1935], a very readable historical outline can be found in [Mark W. Short, **The primitive soluble permutation groups of degree less than 256**, Lecture Notes in Mathematics, Band 1519, Springer-Verlag, 1992]. This work led to a classification of the groups up to degree 15 [Mil1896, Mil1898, Kuh1904]. Having achieved these results development stopped for some time and was taken up again with the arrival of **symbolic computation**. The last twenty years have seen extensive work in this area [BM1983, Roy1987, But1993] noting errors in earlier tables. ...."

So we don't know whether CHINESE[4] Double Oh Eleven has good taste, cf. Appendix B.2.

[3]Courtesy of Double Oh Eleven :-)

[4]THINK OF HIS EUROPEAN COLLEAGUE_S.



of transitive subgroups

Figure 1.1: Transitive subgroups Lattices of permutation groups up to degree 7

Degree 3

Group	Description	Generators	Order
S_3		(123), (12)	6
$+A_3$		(123)	3

Degree 4

Group	Description	Generators	Order
S_4		(1342), (13)	24
$+A_4$		(134), (12)(34)	12
D_4	group of the square	(1234), (13)	8
$\mathbb{Z}_4$	cyclic four group	(1234)	4
$+V_4$	Klein 4-group	(12)(34), (13)(24)	4

Degree 5

Group	Description	Generators	Order
S_5		(12345), (12)	120
$+A_5$		(12345), (21345)	60
F_{20}	metacyclic five group	(12345), (2354)	20
$+D_5$		(12345), (25)(34)	10
$+\mathbb{Z}_5$	cyclic five group	(12345)	5

Degree 7

Group	Description	Generators	Order
S_7		(243756), (123)	5040
$+A_7$		(1234567), (123)	2520
$+PSL_3(2)$		(1234567), (235)(476), (2743)(56)	168
F_{42}	metacyclic seven group	(1234567), (243756)	42
$+F_{21}$		(1234567), (235)(476)	21
D_7		(1234567), (27)(45)(36)	14
$+\mathbb{Z}_7$	cyclic seven group	(1234567)	7

Table 1.1: Transitive Groups up to degree 7

Degree 6

Group	Description	Generators	Order
S_6		(142536), (12345)	720
$+A_6$		(1524)(36), (12345)	360
G_{72}	maximal group imprimitive on two sets of three letters	(123), (456), (12), (45), (14)(25)(36)	72
$+G_{36}^2$	$G_{72} \cap +A_6$	(123), (456), (12)(45), (1425)(36)	36
G_{36}^1		(123), (456), (12)(45), (14)(25)(36)	36
G_{18}		(123), (456), (14)(25)(36)	18
D_6	metacyclic six group	(123)(456), (12)(45), (14)(25)(36)	12
S_3	isomorphic to S_3	(123)(465), (14)(25)(36)	6
$\mathbb{Z}_6$	cyclic six group	(123)(456), (14)(25)(36)	6
G_{48}	maximal group imprimitive on three sets of two letters	(12), (34), (56), (135)(246), (13)(24)	48
$S_4/\mathbb{Z}_4$		(12)(34), (34)(56), (12)(56), (135)(246), (14)(23)(56)	24
G_{24}		(12)(34)(56), (34)(56), (56), (135)(246)	24
$+S_4/V_4$	$G_{48} \cap +A_6$ isomorphic to S_4	(135)(246), (13)(24), (12)(34), (34)(56)	24
$+A_4$	isomorphic to A_4	(12)(34), (34)(56), (12)(56), (135)(246)	12
$PGL_2(5)$	isomorphic to S_5	(126)(354), (12345), (2354)	120
$+PSL_2(5)$	$G_{120} \cap +A_6$ isomorphic to $+A_5$	(126)(354), (12345), (25)(34)	60

Table 1.1(continued) Transitive Groups up to degree 7

1.3 Background and Notation

We give some basic definitions and results of Galois Theory here, [GaloisBks]. Some rudimentary concepts in the theory of permutation groups that will be used in the sequel are also included at the end of this section and let us begin with some elementary stuffs in abstract algebra. The version of Gauß's lemma that we use is:

Lemma 3 *(Gauß's lemma) Let I be a ufd—Unique Factorization Domain, with field of fractions K and let $f(x) \in I[x]$. If $f(x)$ is reducible in $K[x]$ then $f(x)$ is reducible in $I[x]$. More precisely, if $f(x) = A(x)B(x)$ for some nonconstant polynomials $A(x), B(x) \in K[x]$, then there are nonzero elements $r, s \in K$ such that $rA(x) = a(x)$ and $sB(x) = b(x)$ both lie in $I[x]$ and $f(x) = a(x)b(x)$ is a factorization in $I[x]$.*

Definition 4 *Let R be a commutative ring with an identity element.*
A polynomial f in $R[x_1, \dots, x_n]$ which is unchanged by any permutation of the indeterminates $x_1, \dots, x_n$, i.e., $f(x_{1^\sigma}, \dots, x_{n^\sigma}) = f(x_1, \dots, x_n)$ for every $\sigma \in S_n$, is called a **symmetric polynomial** *of the variables $x_1, \dots, x_n$.*
Let $\Upsilon = (z-x_1)(z-x_2)\dots(z-x_n) = z^n - s_1 z^{n-1} + s_2 z^{n-2} - \dots + (-1)^n s_n$. Each s_j is a polynomial of total degree j in $x_1, \dots, x_n$. We call $s_1, \dots, s_n$ the **elementary symmetric polynomials** *of $x_1, \dots, x_n$.*

Theorem 5 *(**Fundamental Theorem of Symmetric Polynomials**) Let $f \in R[x_1, \dots, x_n]$ be symmetric, where R is a commutative ring with an identity element. Then there exists a polynomial $h \in R[x_1, \dots, x_n]$ such that $f = h(s_1, \dots, s_n)$ where the s_j are the elementary symmetric polynomials of $x_1, \dots, x_n$.*

Definition 6 *A* **splitting field** *of $f(x) \in F[x]$, $\mathrm{spl}_F(f)$, is a field extension E/F in which $f(x)$ splits (it is a product of linear factors) while $f(x)$ does not split in any proper subfield of E.*

Since a splitting field is unique up to isomorphism, we often refer to the splitting field.

Theorem 7 *(**Isomorphism Extension Theorem**) Let $\sigma : F \to F'$ be an isomorphism of fields, let $f(x) \in F[x]$, and let $f^*(x) = \sigma(f(x))$ (by acting on the coefficients) be the corresponding polynomial in $F'[x]$; let E be a splitting field of $f(x)$ over F and let E' be a splitting field of $f^*(x)$ over F'. Then there is an isomorphism $\widetilde{\sigma} : E \to E'$ extending σ.*

Definition 8 *The* **polynomial** $f(x) \in F[x]$ *is* **separable** *over F if each of its irreducible factors over F has no repeated roots (in its respective splitting field). Let E/F be an extension of fields, an* **element** *of E is* **separable** *over F if either it is transcendental or its irreducible polynomial over F is separable; if every element in E is separable over F, E is a* **separable extension** *of F.*

Definition 9 *Let E/F be an extension of fields. An element α of E is a* **primitive element** *if $E = F(\alpha)$.*

Theorem 10 *(***Theorem of Primitive Element***) If E is a finite separable extension of F, then E has a primitive element.*

Definition 11 *Let E/F be an* **extension of fields**. *Its* **Galois group**, *denoted by $Gal(E/F)$, (or sometimes $\mathcal{G}$), is the group of all the automorphisms of E fixing F pointwisely under the binary operation of composition.*

Theorem 12 *If $f(x) \in F[x]$ has n distinct roots in its splitting field E, then $Gal(E/F)$ is isomorphic to a permutation group of its roots, which is a subgroup of the symmetric group S_n. We denote this subgroup of S_n by $Gal_F(f)$, (or sometimes Γ), and call it* **the Galois Group of the polynomial** f.

Remark. However, the subgroup in S_n depends on the labelling of the roots; relabelling the roots amounts to conjugation by an element of S_n, cf. Chapter,section 2.2.

Definition 13 *E/F is a* **normal extension** *of fields if every irreducible polynomial over F which has a root in E splits in E.*

Definition 14 *E/F is a* **Galois extension** *of fields if E/F is finite, normal and separable.*

Definition 15 *Let $Aut(E)$ be the group of all the automorphisms of a field E. If G is a subset of $Aut(E)$, then $E^G = \{\alpha \in E : \sigma(\alpha) = \alpha \text{ for all } \sigma \in G\}$ is called the* **fixed field** *of G in E.*

The author was always puzzled with the various definitions of normal extensions by different writers in [GaloisBks]. Fortunately the following two theorems should explain the matter sufficiently clear.

Theorem 16 *Let E/F be a finite extension of fields with Galois Group $\mathcal{G} = Gal(E/F)$. The following conditions are equivalent:*
(i) $F = E^{\mathcal{G}}$;
(ii) every irreducible $p(x) \in F[x]$ with one root in E, has all its roots in E, and each root is simple;
(iii) E is a splitting field of some separable polynomial $f(x) \in F[x]$.

Theorem 17 *Let E/F be a finite extension of fields. Then E is a splitting field of a polynomial over F iff every irreducible polynomial over F which has a root in E splits in E.*

We are now able to state

Theorem 18 *(𝔗𝔥𝔢 𝔉𝔲𝔫𝔡𝔞𝔪𝔢𝔫𝔱𝔞𝔩 𝔗𝔥𝔢𝔬𝔯𝔢𝔪 𝔬𝔣 𝔊𝔞𝔩𝔬𝔦𝔰 𝔗𝔥𝔢𝔬𝔯𝔶) Let E/F be a Galois extension with Galois Group $\mathcal{G} = Gal(E/F)$. Let H be a subgroup of $\mathcal{G}$ and B be a subfield of E containing F,*
(i) The function $\gamma : Sub(\mathcal{G}) \to Lat(E/F)$, defined by $H \mapsto E^H$, is an order reversing bijection with inverse $\gamma^{-1} : B \mapsto Gal(E/B)$. In short, there is a **bijection** *between the subgroups of $\mathcal{G}$ and the lattice of subfields of E containing F.*
(ii) $E^{Gal(E/B)} = B$ and $Gal(E/E^H) = H$.
(iii) $E^{H\vee K} = E^H \cap E^K$ and $E^{H\cap K} = E^H \vee E^K$; $Gal(E/B \vee C) = Gal(E/B) \cap Gal(E/C)$ and $Gal(E/B \cap C) = Gal(E/B) \vee Gal(E/C)$,
where $H \vee K$ denotes the smallest subgroup (or subfield) containing both H and K.
(iii) Degree & Index: $[B : F] = [\mathcal{G} : Gal(E/B)]$ and $[\mathcal{G} : H] = [E^H : F]$.
(iv) B/F is a Galois extension iff $Gal(E/B)$ is a normal subgroup of $\mathcal{G}$. In this case, $Gal(B/F) \cong \mathcal{G}/Gal(E/B)$.

The following results are used somewhere in this thesis:

Theorem 19 *If $f(x) \in F[x]$ is irreducible over a field F, then its Galois group $Gal_F(f)$ is* **transitive** *on its roots, i.e., for every two roots α and β of f, there is a permutation $\sigma \in Gal_F(f)$ with $\sigma(\alpha) = \beta$. The converse is true if $f(x)$ has no repeated roots.*

Theorem 20 *Let $f(x) \in F[x]$ be a separable polynomial, and E/F be its splitting field. Let $f(x) = g(x)h(x)$ in $F[x]$, and let B/F and C/F be splitting fields of $g(x), h(x)$ respectively, contained in E. If $B \cap C = F$, then $Gal(E/F) \cong Gal(B/F) \times Gal(C/F)$, where $\times$ denote the direct product of groups.*

Definition 21 *Let $f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$ and $g(x) = b_0x^m + b_1x^{m-1} + \cdots + b_m$ be two univariate polynomials over an integral domain D.*

If $\alpha_1, \dots, \alpha_n$ are the roots of f and $\beta_1, \dots, \beta_m$ are the roots of g in their common splitting field, then the **resultant** $\text{res}_x(f,g)$ *of f and g is*

$$\text{res}_x(f,g) = a_0^m b_0^n \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j).$$

The resultant can be shown to be the determinant of the **Sylvester matrix** *of f and g, namely,*

$$\text{res}_x(f,g) = \begin{vmatrix} a_0 & a_1 & \cdots & a_n & & & & \\ & a_0 & a_1 & \cdots & a_n & & & \\ & & \cdots & \cdots & \cdots & \cdots & & \\ & & & a_0 & a_1 & \cdots & a_n & \\ b_0 & b_1 & \cdots & \cdots & b_m & & & \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \\ & & b_0 & b_1 & \cdots & \cdots & b_m & \end{vmatrix} \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} m \text{ rows} \\ \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} n \text{ rows} \end{array}.$$

(In all blank spaces we must substitute zeros.)
It is an element of the integral domain D.

Definition 22 *Let $f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$ be a polynomial of degree n over an integral domain D with roots $\alpha_1, \dots, \alpha_n$ in its splitting field. Let $\Delta = \prod_{i<j}(\alpha_i - \alpha_j)$. The* **discriminant** *of f, $\text{disc}(f)$, is defined as $\text{disc}(f) = a_0^{2n-2}\Delta^2$. Being defined in this way, it is an element of D. Notice that $\text{disc}(f) = \Delta^2$ for a monic polynomial f.*
We observe an interesting relation, $\text{res}_x(f, f') = (-1)^{\frac{n(n-1)}{2}} a_0 \, \text{disc}(f)$, where $f'(x)$ is the derivative of $f(x)$.

For computation of resultants and discriminants *symbolically* (quickly and avoids coefficients blowup), see [CAbks].

The following algorithm can be found in [Coh93] and is used to create squarefree resolvent. (This will become clear later.)

Theorem 23 *(**Tschirnhausen Transformation**). Given a monic irreducible polynomial T defining a number field $K = \mathbb{Q}(\theta)$, the following algorithm output another such polynomial U defining the same number field.*
1. [choose random polynomial] Let $n \leftarrow \deg(T)$. Choose at random a polynomial $A \in \mathbb{Z}[x]$ of degree less than or equal to $n-1$.
2. [Compute characteristic polynomial] Compute the characteristic polynomial U of $\alpha = A(\theta)$. Set $U \leftarrow \text{res}_y(T(y), X - A(y))$.
3. [Check degree] Using Euclid's algorithm, compute $V \leftarrow \gcd(U, U')$. If V is constant, then output U and terminate the algorithm, otherwise go to step 1.

Theorem 24 *Let $f(x) \in F[x]$ be a polynomial of degree n over a field F which has no repeated roots in its splitting field. $\sqrt{\mathrm{disc}(f)} \in F$ iff its Galois group $Gal_F(f) \subseteq A_n$.*

Theorem 25 *A field of finite size must has exactly p^n elements, where p is a rational prime. It is the splitting field of $x^{p^n} - x$ over $\mathbb{Z}_p$. Thus all finite fields with the same number of elements are isomorphic. We use $GF(p^n)$ to denote the* **Galois field** *(finite) with p^n elements.*

Theorem 26 *$Gal(GF(p^n)/GF(p)) \cong \mathbb{Z}_n$ with generator $u \mapsto u^p$,where $\mathbb{Z}_n$ denote the cyclic group of order n.*

Notice that $GF(p)$ is just the field $\mathbb{Z}_p$, where p is a prime number.

Theorem 27 *For every p and n there are polynomials $f(x)$ of degree n which are irreducible* mod p. *They are all divisors of $x^{p^n} - x$* mod p. *Moreover, $x^{p^n} - x$ is the product of all monic irreducible polynomials in $\mathbb{Z}_p[x]$ whose degrees divide n.*

Finally we include a few basic concepts in the theory of permutation groups, [Wie64].

Definition 28 *A permutation group G on Ω is called* **semiregular** *if every permutation $\neq id$ (the identity permulation) in G has no fixed point. G is called* **regular** *if it is semiregular and transitive.*

Definition 29 *Let G be a permutation group on Ω. We call a subset ψ of Ω a* **block** *of G if for each $g \in G$ the image set ψ^g either coincide with ψ or has no point in common with ψ.*

Definition 30 *A transitive permutation group is called* **imprimitive** *if it has at least one nontrivial block ψ (i.e., $\psi \neq \phi, \{\alpha\}, \Omega$). Otherwise, it is called a* **primitive group.**

1.4 Content and Contribution of THIS thesis

Although the author has made a MAPLE program, [MAPLE], which can determine the Galois group of any irreducible polynomials over $\mathbb{Q}$ up to degree 7 within seconds (usually less than two), cf. Appendix A, there has been no contribution to Mathematics and Computer Science.

Since with reference to Appendix B, this MAPLE program should adequately be considered as a TOY in *Computational Galois Theory*, cf. Appendix B.1. However it is the author's first attempt to make something nontrivial in a newly emerging field called *Symbolic and Algebraic Computation* (or *Computer Algebra, Computational Algebra*), cf. Appendix B.2.

There is a saying that "Pure mathematical ideas are not found in any computer program or its output." There are far more sayings that I can understand though. I give some findings in my journey here and I hope that some competent and industrious people may draw the proper conclusions, cf. [CAbks] and Appendix B.

* * *

Chapter 2 in this thesis describes Stauduhar's method of determining Galois groups, [Sta73]. The computational requirements of this method are the approximated roots (to any desired accuracy) and synthetic division of integral polynomials, these could easily be achieved in 1973.
have been

In Chapter 4, we discuss Soicher and McKay's method of computing Galois groups, [SM85] and [Soi84]. This method is symbolic and exact and requires the use of polynomial factorization algorithms developed since 1968, [Ber68] and [Zas69].

Since the topic of exact polynomial factorization has formed an important part in the field of *Symbolic and Algebraic Computation*, and because this field is relatively new, we find it worthwhile to include the material in Chapter 3 despite that they can easily be found in Computer Algebra Textbooks, [CAbks]. Those textbooks have been published since late eighties.

* * *

A splendid experience with Computer Algebra Software System like MAPLE, [MAPLE] or Computational Group Theory System like GAP, [GAP] is that you could be able to know the exact solutions to (some) arbitrary concrete questions within seconds. For instance, to factorize $x^{90}-1$, to do indefinite integration $\int \frac{1}{x^3+ax^2+x}dx$; $\int \frac{\sin x}{x}dx$, to display the subgroup lattice of S_5, to find out the Galois group of x^4-2; x^5+2^5. To know this kind of answers is not easily accomplished by methods in classical Mathematics text—at best a very tedious method that may not be at all clear and practical is available, which might not guarantee to work in all cases; maybe only scattered examples are found in the usual cases. Nowadays, all this can be done within seconds on a cheap Pentium[6].

[5]The author has difficulty in producing instructive examples since he hasn't played enough yet.

[6]The current price is less than US$1,000 for Pentium-166 at Hong Kong.

There is no need to be rich or powerful. Just use your mind lightly and softly, without any efforts. To put the game in another way: What is inside the computer? Does it succeed just by calculating fast? Is artificial intelligence inside? Is mathematics involved?[7] Pure mathematics or Applied mathematics?

This work is an investigation of the mathematics, if any, and the computing knowledge involved, for the computation of Galois group of polynomials over **Q**.

[7] It shouldn't be of very farsighted to conclude that if a computer says that the indefinite integral of sin(x)/x is not elementary then either it is lying or (s)he does in fact know some mathematics.

Chapter 2

Stauduhar's Method

The principal reference for this chapter is "R. P. Stauduhar, *The determination of Galois Groups*, Mathematics of Computation 27 (1973), 981-996," [Sta73].

2.1 Overview & Restrictions

A technique is described for the nontentative computer determination of the Galois groups of irreducible polynomials with integer coefficients. The technique for a given polynomial involves finding high-precision approximations to the roots of the polynomial, and fixing an ordering for these roots. The roots are then used to create (relative) resolvent polynomials of relatively small degree, the linear factors of which determine new ordering for the roots. Sequences of these resolvents isolate the Galois group of the polynomial. Machine implementation of the technique requires the use of multiple-precision integer and multiple-precision real and complex floating-point arithmetic. Using this technique, Stauduhar has developed programs for the determination of the Galois groups of polynomials of degree $N \leq 7$. Two exemplary calculations are given.

Restrictions The algorithm to be described will apply only to irreducible monic polynomials with integer coefficients. Since any polynomial with rational coefficients can easily be transformed into a monic polynomial with integer coefficients equivalent with respect to its Galois group, these latter two adjectives create no genuine restriction.

The irreducibility restriction is genuine, however. For suppose $f(x) = g(x)h(x)$ in $\mathbf{Q}[x]$, and suppose B and C are the splitting fields of g and h, respectively. If $B \cap C =$ the rational, then the Galois group of $f(x)$ is the direct sum of the Galois

groups of $g(x)$ and $h(x)$, theorem 20, and there is no difficulty. If, on the other hand, $B \cap C$ is larger than the rationals, then the group of $f(x)$ is not easily determined from those of $g(x)$ and $h(x)$ without explicit knowledge of the relations which exist between the roots of g and the roots of h.
As will become clear, the irreducibility restriction is not essential, but it greatly simplifies the work of implementing the algorithm for polynomials of a given degree. In this instance, a forthcoming article might be helpful, Appendix B.1[1].

There is another restriction. Application of the algorithm to polynomials of degree n requires knowledge of all transitive (theorem 19) permutation groups of that degree.
However, the memory size of computers currently (in 1973) available will limit the use of the algorithm in the near future to cases for which such knowledge already exists. Consequently, this restriction is not practically important in 1973.

2.2 Representation of the Galois Group

In the classical development of Galois theory, the Galois group of a polynomial is regarded as a group of permutations on the roots of the polynomial, theorem 12. From the standpoint of computation, this concrete, finite representation of the group seems to offer the best hold on the problem of its determination. Consequently, the Galois group will here be regarded as a group of permutations.

More specifically, let S_n be the symmetric group on n letters and $\pi, \sigma' \in S_n$ be maps of $\{1, 2, \ldots, n\}$ onto itself. Multiplication of permutations is composition, so that $k^{\sigma\pi} = (k^{\sigma})^{\pi}$.

Let $f(x)$ be a polynomial with rational coefficients and roots $\alpha_1, \alpha_2, ..., \alpha_n$. Let E be the splitting field of $f(x)$. Let $\mathcal{G} = Gal(E/\mathbf{Q})$ be the group of automorphisms of E fixing $\mathbf{Q}$ pointwisely, see theorem12. Suppose $\phi \in \mathcal{G} = Gal(E/\mathbf{Q})$. Then ϕ induces a permutation on $\alpha_1, \alpha_2, ..., \alpha_n$, which can be set forth as follows: $\begin{pmatrix} \alpha_1, & \cdots & , \alpha_n \\ \alpha_1^{\phi}, & \cdots & , \alpha_n^{\phi} \end{pmatrix}$ or $\begin{pmatrix} \alpha_1, & \cdots & , \alpha_n \\ \alpha_{i_1}, & \cdots & , \alpha_{i_n} \end{pmatrix}$ or $\begin{pmatrix} 1, & \cdots & , n \\ i_1, & \cdots & , i_n \end{pmatrix}$. Letting π_ϕ denote the final expression here, it is clear that the map $\phi \mapsto \pi_\phi$ defines an isomorphism from $\mathcal{G} = Gal(E/\mathbf{Q})$ onto a subgroup $\Gamma = Gal_{\mathbf{Q}}(f)$ of S_n.

It is important to observe that the group Γ depends on the chosen labelling of the roots of $f(x)$.

- For if a new labelling $\alpha_1' = \alpha_{1^\pi}, \cdots, \alpha_n' = \alpha_{n^\pi}$ is chosen, then the isomorphism given above will carry $\mathcal{G}$ onto $\pi\Gamma\pi^{-1}$.

Consequently, when the Galois group of a polynomial is given as a group of permutations, an ordering of the roots of the polynomial must also be given.

The material presented in the following two sections is well known from classical Galois theory. A few of the theorems and definitions are presented in a slightly unusual form, one which has been dictated by the numerical character of their application. The others are set forth simply for completeness and for clarity of exposition of the main algorithm.

2.3 Groups and Functions

Let $\Upsilon(x_1, \cdots, x_n)$ be a polynomial in the indeterminates $x_1, \cdots, x_n$. The course of the Galois algorithm requires that the action of permutations on the arguments of such functions be considered.

Definition 31 *Let $\Upsilon(x_1, \cdots, x_n)$ be a polynomial in the indeterminates $x_1, \cdots, x_n$. Let $\pi \in S_n$. Then $\Upsilon^\pi(x_1, \cdots, x_n) = \Upsilon(x_{1^\pi}, \cdots, x_{n^\pi})$.*

If two permutations are applied sequentially to a function, we obtain the following:

- $(\Upsilon^\pi)^\sigma(x_1, \cdots, x_n) = \Upsilon^{\pi\sigma}(x_1, \cdots, x_n)$.

It may be that a function $\Upsilon(x_1, \cdots, x_n)$ is left unchanged by the action of certain permutations. For example, $\Upsilon(x_1, \cdots, x_n) = x_1x_3 + x_2x_4$ is unchanged by any of the permutations $\{identity, (1234), (1423), (13)(24),$
$(12)(34), (1423), (13), (24)\}$.

The collection of all permutations on n letters which leave a function unchanged clearly forms a group. (The permutations in the above example form the group of the square.)

Definition 32 *Let $\Upsilon(x_1, \cdots, x_n)$ be a polynomial with integral coefficients in the indeterminates $x_1, \cdots, x_n$. Let G be a group of permutations on $1, ..., n$. If Υ is left unchanged by precisely the permutations of G, we say that Υ* **belongs to** *G.*

In this definition we restrict the coefficients of Υ to be integers for reasons that will be apparent later.

Theorem 33 *Let G be a subgroup of S_n. Then there is a function $\Upsilon(x_1, \cdots, x_n)$ which belongs to G.*

Proof.
Let $\Psi(x_1,\cdots,x_n) = x_1^1 x_2^2 \cdots x_n^n$. Define $\Upsilon(x_1,\cdots,x_n) = \sum_{\sigma\in G} \Psi^\sigma(x_1,\cdots,x_n)$. Clearly, Υ belongs to G. For if $\pi \in G$, then the application of π merely permutes the terms of Υ among themselves, but if $\pi \notin G$, then the terms of Υ are moved onto terms corresponding to the right coset $G\pi$ of G. ■

Definition 34 *Given a function $\Upsilon(x_1,\cdots,x_n)$ and a permutation $\pi \in S_n$, the function Υ^π is called a conjugate value or a conjugate function of the function Υ.*

Now we can ask the question: Given a polynomial $\Upsilon(x_1,\cdots,x_n)$, and a group $H \subset S_n$, how many distinct conjugate values does Υ take under the permutations of H? This is answered by the following:

Proposition 35 *Let H be a subgroup of S_n. Suppose $\Upsilon(x_1,\cdots,x_n)$ belongs to $G \subset S_n$. Then Υ takes exactly $[H : H\cap G]$ distinct conjugate values under the permutations of H.*

Proof. Suppose $\pi_1, \pi_2 \in H$. We will show that $\Upsilon^{\pi_1} = \Upsilon^{\pi_2}$ iff π_1 and π_2 lie in the same right coset of $H\cap G$. $\Upsilon^{\pi_1} = \Upsilon^{\pi_2}$ iff $(\Upsilon^{\pi_1})^{\pi_2^{-1}} = \Upsilon$ iff $\Upsilon^{\pi_1\pi_2^{-1}} = \Upsilon$ iff $\pi_1\pi_2^{-1} \in H\cap G$ iff $(H\cap G)\pi_1 = (H\cap G)\pi_2$. ■

Definition 36 *Suppose G and H are subgroups of S_n and $\Upsilon(x_1,\cdots,x_n)$ belongs to G. Let $G' = G\cap H$. We say then that Υ* **belongs to** G' **in** H. *That is, among the permutations in H, exactly those of G' leave Υ unchanged.*

Proposition 37 *Suppose G and H are subgroups of S_n, with $G \subset H$, and suppose $\Upsilon(x_1,\cdots,x_n)$ belongs to G in H. If $\pi \in H$ then Υ^π belongs to $\pi^{-1}G\pi$ in H.*

Proof. $\Upsilon(x_1,\cdots,x_n)$ belongs to G in H iff $\forall\sigma \in H, \Upsilon^\sigma = \Upsilon \Leftrightarrow \sigma \in G$
iff $\forall\sigma \in H, (\Upsilon^\pi)^\sigma = \Upsilon^\pi \Leftrightarrow \sigma \in \pi^{-1}G\pi$
iff Υ^π belongs to $\pi^{-1}G\pi$ in H. ■

Now suppose $\Upsilon(x_1,\cdots,x_n)$ belongs to G in H, and $[H : G] = k$. Then we can choose permutations $\pi_i \in H$ so that $H = G\pi_1 \cup \cdots G\pi_k$ and hence, as we have shown, so that the functions $\Upsilon = \Upsilon^{\pi_1}, \Upsilon^{\pi_2}, \cdots, \Upsilon^{\pi_k}$ are formally distinct. It should be noted, however, that the values of these functions are not necessarily distinct on a fixed n-tuple of numbers.

For example, let $\Upsilon(x_1,x_2,x_3,x_4) = x_1x_2^2 + x_2x_3^2 + x_3x_4^2 + x_4x_1^2$, so that Υ belongs to the cyclic group generated by (1234), with right coset representatives $\{identity, (12), (13), (23), (123), (132)\}$ in S_4. Now if we evaluate Υ and its conjugates on the four roots of $f(x) = x^4 - 2$, with the ordering $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) =$

$(\sqrt[4]{2}, -\sqrt[4]{2}, i\sqrt[4]{2}, -i\sqrt[4]{2})$, we observe that
$\Upsilon^{(23)}(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \Upsilon^{(123)}(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 0.$

2.4 Relative Resolvents

Functions and the Galois Group In this section, we consider the relation between the Galois group $\Gamma = Gal_{\mathbf{Q}}(f)$ of an irreducible nth degree polynomial $f(x)$ and the values taken on the roots of $f(x)$ by functions belong to subgroups of S_n.

Theorem 38 *Let $f(x)$ be a monic irreducible polynomial of degree n with integer coefficients. Let $\alpha_1, \alpha_2, ..., \alpha_n$ be a fixed ordering of the roots of $f(x)$.*
Suppose H is a transitive subgroup of S_n, and suppose that, with respect to the given ordering of the roots, the Galois group Γ of $f(x)$ is a subgroup of H. Let G be a subgroup of H and $\Upsilon(x_1, \cdots, x_n)$ a function belongs to G in H. Let $\pi_1, \pi_2, \cdots, \pi_k$ be representative for the right cosets of G in H.
Then the **relative resolvent** *polynomial*

$$\Re_{(H,G)}(x) = \prod_{i=1}^{k}(x - (\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i}) \tag{2.1}$$

has integer coefficients.

Proof. For each i, $1 \le i \le k$, $(\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i}$ is an algebraic integer. Hence, the coefficients of $\Re_{(H,G)}(x)$ are algebraic integers.
Now suppose $\sigma \in \Gamma$. Then $\sigma \in H$, and hence $\sigma(\Re_{(H,G)}(x)) =$
$\prod_{i=1}^{k}(x - ((\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i})^{\sigma}) = \prod_{i=1}^{k}(x - (\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i\sigma})$.
But the set $\pi_1\sigma, \pi_2\sigma, \cdots, \pi_k\sigma$ is also a set of right coset representatives for G in H. Thus, the application of σ has merely permuted the roots of $\Re_{(H,G)}(x)$, leaving the coefficients fixed. The coefficients of $\Re_{(H,G)}(x)$ are then algebraic integers left fixed by Γ and are therefore, by Galois theory, rational integers. ■

At this point it is worth mentioning that the roots of $\Re_{(H,G)}(x)$ may be not be distinct, as the example following Proposition 37 shows.

2.4.1 Computing Resolvents Numerically

Assume that high-precision approximations to the roots of $f(x)$ are known. Since the resolvents being dealt with are known to have integer coefficients, it is only necessary to calculate the coefficients or resolvents to within an accuracy of $\pm\frac{1}{2}$ in order to determine them exactly. To insure this accuracy, the roots of a typical resolvent $\Re_{(M_1,M_2)}(x) = \prod_{i=1}^{k}(x - (\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i})$ can be calculated to high precision, using the given approximations to $\alpha_1, \alpha_2, ..., \alpha_n$, and the product can then be expanded to obtain approximations to the coefficients. Multiple-precision complex floating-point arithmetic routines are generally required to obtain the necessary accuracy.

In the discussion of the Searching Procedures in the next section (also see theorem 39 below), it is assumed that those integer roots of resolvents with respect to which reordering is taking place are not repeated roots. In the case that all the integer roots of a resolvent have multiplicity greater than one, the resolvent can be recalculated with respect to a new function, or the input polynomial can be operated upon with a Tschirnhaus transformation, theorem 23 (which preserves the Galois group) in order to obtain a resolvent without repeated roots.

2.4.2 Integer Roots of Resolvent Polynomials

Theorem 39 *Let all the assumptions of Theorem 38 hold.* $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ *is a root of* $\Re_{(H,G)}(x)$, *since one of the coset representatives of* G *in* H *lies in* G *itself.*
Assume $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ *is not a repeated root of* $\Re_{(H,G)}(x)$.
Then $\Gamma \subset G$ *iff* $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ *is a rational integer.*

Proof. First, observe that $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ is an algebraic integer.
Now assume $\Gamma \subset G$. Let $\sigma \in \Gamma$. Then $\sigma \in G$, hence $\Upsilon^{\sigma} = \Upsilon$. Consequently, $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ is fixed under the action of all elements of the Galois group, hence it is a rational number, by Galois theory. Since it is an algebraic integer, it is a rational integer.
Conversely, assume $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ is a rational integer. Then $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$ is fixed by the Galois group Γ. But among the permutations of H only those of G fix $\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n)$, since it is not a repeated root of $\Re_{(H,G)}(x)$. Hence $(\Gamma \cap H) \subset G$. But by assumption $\Gamma \subset H$. Thus $\Gamma \subset G$. ■

Corollary 40 *Assume* $(\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i}$ *is not a repeated root of* $\Re_{(H,G)}(x)$. *Then* $\Gamma \subset \pi_i^{-1} G \pi_i$ *iff* $(\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i}$ *is a rational integer.*

Corollary 41 *Suppose* $(\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i}$ *is a rational integer, and not a repeated root of* $\Re_{(H,G)}(x)$, *so that* $\Gamma \subset \pi_i^{-1} G \pi_i$. *If the roots of* $f(x)$ *are reordered according to the rule* $\alpha_j' = \alpha_{j^{\pi_i}}$, *then* $\Upsilon(\alpha_1', \alpha_2', ..., \alpha_n')$ *is a rational integer, and with respect to this new ordering,* $\Gamma \subset G$.

Something now should be said about how integer roots of resolvent polynomials are identified. If a given (approximate) root of the resolvent $\Re_{(M_1,M_2)}(x)$ seems to be an integer to within some reasonable tolerance, it can be rounded to that integer and a synthetic division can be performed with $\mathfrak{R}_{(M_1,M_2)}(x)$ to test whether the integer is indeed a root of the resolvent.

2.5 The Determination of Galois Groups

Since polynomials over $\mathbb{Q}$ are separable, theorem 24 is well-adapted in our situation here. It is very useful in trying to determine the Galois group of a polynomial.

Theorem 42 *Let $f(x)$ be a monic irreducible polynomial of degree n with integer coefficients. Then the Galois group of $f(x)$ is a subgroup of the alternating group A_n iff the discriminant* $\operatorname{disc}(f(x))$ *is a perfect square.*

Now suppose that a monic irreducible polynomial $f(x)$ of degree n with integer coefficients is given. Assume that the discriminant $\operatorname{disc}(f(x))$ and its square root are known. (There is a simple recursive technique for computing the discriminant of a polynomial, given its coefficients. See [Sta73] and [CAbks].) Assume further that high-precision approximations to the roots of $f(x)$ are known.

2.5.1 Searching Procedures

Place the approximated roots in an (arbitrary) initial ordering $\alpha_1, \alpha_2, ..., \alpha_n$. Let Γ denote the Galois group of $f(x)$ with respect to this ordering. Now suppose that M is a *maximal* transitive subgroup of $S_n, M \neq A_n$,and $[S_n : M] = k$. (The case $M = A_n$ will be considered later in this section.) We know, *a priori*, that $\Gamma \subset S_n$. To determine if $\Gamma \subset M$, or some conjugate of M, calculate a resolvent polynomial of degree k, numerically, using a function $\Upsilon(\alpha_1', \alpha_2', ..., \alpha_n')$ that belongs to M in S_n, and a set $\pi_1, \cdots, \pi_k$ of right coset representatives for M in S_n.

According to Theorem 38, this resolvent is monic with integer coefficients. Test the resolvent for integer roots. If it has none, then Γ is not contained in any of the conjugates of M, and similar resolvents may be computed, corresponding to other conjugacy classes of maximal transitive subgroups of S_n.

Suppose, however, that $\mathfrak{R}_{(S_n,M)}(x)$ has an integer root. Then this root is $(\Upsilon(\alpha_1, \alpha_2, ..., \alpha_n))^{\pi_i}$, where π_i is one of the chosen coset representatives, and in consequence of the first corollary to Theorem 39, $\Gamma \subset \pi_i^{-1} M \pi_i$.

The roots of $f(x)$ must now be reordered, so that $\alpha_j' = \alpha_{j^{\pi_i}}$. After the reordering, according to the second corollary to Theorem 39, we have $\Gamma \subset M$.

Now, assuming that $\Gamma \subset M$, suppose M^* is a maximal transitive subgroup of M, and Υ^* is a function belonging to M^* in M. Then a resolvent polynomial $\Re_{(M:M^*)}(x)$ of degree $[M : M^*]$ is calculated, and this new polynomial is tested for integer roots. (This resolvent is, again by Theorem 38, monic with integer coefficients.) If an integer root of $\Re_{(M:M^*)}(x)$ is found, the roots of $f(x)$ are again reordered to insure that $\Gamma \subset M^*$.

Searching continues in this way until either none of the resolvents at a given level yield an integer root, or a *minimal* transitive subgroup of S_n is located. (We need consider only transitive subgroups of S_n in the course of the search, since $f(x)$ is assumed irreducible.)

———

At each level of the search, clearly, only groups not previously eliminated need be considered. Suppose, for example, that S_n has maximal subgroups M_1 and M_2, and it is discovered that $\Re_{(S_n:M_1)}(x)$ has no integer roots, but that $\Re_{(S_n:M_2)}(x)$ does, so that $\Gamma \not\subset M_1$, and $\Gamma \subset M_2$. Then, for the remainder of the search, groups which lie within $M_1 \cap M_2$ are automatically ruled out as possibilities for Γ.

We have not yet described how the discriminant is used. It is used in two ways. First, if none of the resolvents associated with the maximal transitive subgroups of S_n yield an integer root, then $\Gamma = A_n$ or $\Gamma = S_n$, depending on whether or not disc$(f(x))$ is a perfect square. Second, if disc$(f(x))$ is a square, and we have determined that $\Gamma \subset M$, then we know that $\Gamma \subset M \cap A_n$. Use of this fact simplifies the search procedure to some extent.

2.5.2 Data: $\Upsilon(x_1, x_2, \ldots, x_n)$, Coset Representatives & Searching Diagram

The following tables and diagrams contain the data needed to find the Galois groups of polynomials of degree $n \leq 7$. Information used in constructing the tables and trees presented here has been gleaned from various sources by Stauduhar. He has constructed a similar table and tree for the degree-eight case. It was not given in [Sta73], since it has not been checked by actual computation.

Table 2.1 describes, when required, functions belonging to these groups, as well as the necessary coset representatives.

The alternating and symmetric groups of the various degrees are not included in the tables. No functions are given belonging to the groups for which no resolvent is computed. For example, in the degree five case, if the Galois group Γ of a polynomial $f(x)$ is a subgroup of F_{20}, and $\mathrm{disc}(f(x))$ is a perfect square, then $\Gamma \subset +D_5$, otherwise $\Gamma = F_{20}$. Consequently, it is never necessary to compute a resolvent of the form $\Re_{(H,+D_5)}(x)$, when $\mathrm{disc}(f(x))$ is known.

The groups of degree six have been divided into three categories: the groups imprimitive (see definition 30) on two sets of three letters, the groups imprimitive on three sets of two letters but not two sets of three letters, and primitive groups. They are given in this order in the table.

Degree 4

Group	$\subset$	$\Upsilon(x_1,x_2,\ldots,x_n)$	right coset representatives
D_4	S_4	$x_1x_3+x_2x_4$	id, (23), (34)
$\mathbb{Z}_4$	D_4	$x_1x_2^2+x_2x_3^2+x_3x_4^2+x_4x_1^2$	id, (12)(34)
$+V_4$			

Degree 5

Group	$\subset$	$\Upsilon(x_1,x_2,\ldots,x_n)$	right coset representatives
F_{20}	S_5	$(x_1x_2+x_2x_3+x_3x_4+x_4x_5+x_5x_1$ $-x_1x_3-x_3x_5-x_5x_2-x_2x_4-x_4x_1)^2$	id, (12)(34), (12435), (15243), (12453), (12543)
$+D_5$			
$+\mathbb{Z}_5$	$+D_5$	$x_1x_2^2+x_2x_3^2+x_3x_4^2+x_4x_5^2+x_5x_1^2$	id, (12)(35)

Degree 6

Group	$\subset$	$\Upsilon(x_1,x_2,\ldots,x_n)$	right coset representatives
G_{72}	S_6	$x_1x_2x_3+x_4x_5x_6$	id, (2543), (236)(45), (25436), (25)(34), (2453), (25), (2345) (24536), (3645)
$+G_{36}^2$			
G_{36}^1	G_{72}	$(x_1-x_2)(x_2-x_3)(x_3-x_1)$ $\cdot(x_4-x_5)(x_5-x_6)(x_6-x_4)$	id, (56)
G_{18}	G_{36}^1 G_{72}	$(x_1-x_2)(x_2-x_3)(x_3-x_1)$ $+(x_4-x_5)(x_5-x_6)(x_6-x_4)$	id, (12)(45), (56), (12)(465)

Table 2.1: The function $\Upsilon(x_1,x_2,\ldots,x_n)$ and right coset representatives

Degree 6 (Continued)

D_6	G^1_{36} G_{72}	$x_1x_4+x_2x_5+x_3x_6$	id, (123), (132), (56), (123)(56), (132)(56)
S_3	G_{18}	$x_1x_4+x_2x_6+x_3x_5$	id, (123), (132)
$\mathbb{Z}_6$	G_{18}	$x_1x_6^2+x_2x_4^2+x_3x_5^2$ $+x_4x_2^2+x_5x_1^2+x_6x_2^2$	id, (123), (132)
G_{48}	S_6	$x_1x_2+x_3x_4+x_5x_6$	id, (24635), (26)(35), (354), (2345), (253), (345), (256)(34), (26435), (2346), (234), (25)(36), (2435), (24)(35), (26543)
$S_4/\mathbb{Z}_4$	G_{48}	$(x_1+x_2-x_3-x_4)$ $\cdot(x_3+x_4-x_5-x_6)$ $\cdot(x_5+x_6-x_1-x_2)$ $\cdot(x_1-x_2)$ $\cdot(x_3-x_4)(x_5-x_6)$	id, (12)
G_{24}	G_{48}	$(x_1+x_2-x_3-x_4)$ $\cdot(x_3+x_4-x_5-x_6)$ $\cdot(x_5+x_6-x_1-x_2)$	id, (13)(24)
$+S_4/V_4$			
$+A_4$	$+S_4/V_4$	see G_{24}	id, (13)(24)
$PGL_2(5)$	S_6	$(x_1x_2+x_3x_5+x_4x_6)$ $\cdot(x_1x_3+x_4x_5+x_2x_6)$ $\cdot(x_3x_4+x_1x_6+x_2x_5)$ $\cdot(x_1x_5+x_2x_4+x_3x_6)$ $\cdot(x_1x_4+x_2x_3+x_5x_6)$	id, (13), (23), (123), (132), (12)
$+PSL_2(5)$			

Table 2.1(Continued) The function $\Upsilon(x_1, x_2, \ldots, x_n)$ and right coset representatives

Degree 7

Group	$\subset$	$\Upsilon(x_1,x_2,\dots,x_n)$	right coset representatives
$+PSL_3(2)$	S_7	$x_1x_2x_4+x_1x_3x_7$ $+x_1x_5x_6+x_2x_3x_5$ $+x_2x_6x_7+x_3x_4x_6$ $+x_4x_5x_7$	id, (356), (365), (34)(56), (354), (364), (456), (345), (36)(45), (465), (35)(46), (346), (47)(56), (35)(47), (36)(47), (243756), (243675), (243)(57), (2475), (247536), (247563), (246375), (246)(57), (246753), (24)(375), (24)(36)(57), (24)(567), (245)(37), (245736), (245673)
F_{42}	S_7	$x_1x_2x_4+x_1x_2x_6$ $+x_1x_3x_4+x_1x_3x_7$ $+x_1x_5x_6+x_1x_5x_7$ $+x_2x_3x_5+x_2x_3x_7$ $+x_2x_4x_5+x_2x_6x_7$ $+x_3x_4x_6+x_3x_5x_6$ $+x_4x_5x_7+x_4x_6x_7$	Let A be the set consisting of the even coset representatives for $PSL_3(2)$ in S7. Let B be the set of all coset representatives for G_{21}in $PSL_3(2)$. Then the required 120 coset representatives here are.given by $A \cdot B$
$+F_{21}$	$+PSL_3(2)$	see $F_{42} \subset S_7$	id, (37)(56), (23)(74), (2347)(56), (24)(56), (24)(37), (2743)(56), (27)(34)
D_7	F_{42}	$x_1x_2+x_2x_3+$ $\dots +x_6x_7+x_7x_1$	id, (235)(476), (253)(467)
$+\mathbb{Z}_7$	$+F_{21}$	see $D_7 \subset F_{42}$	id, (235)(476), (253)(467)

Table 2.1(Continued) The function $\Upsilon(x_1, x_2, \dots, x_n)$ and right coset representatives

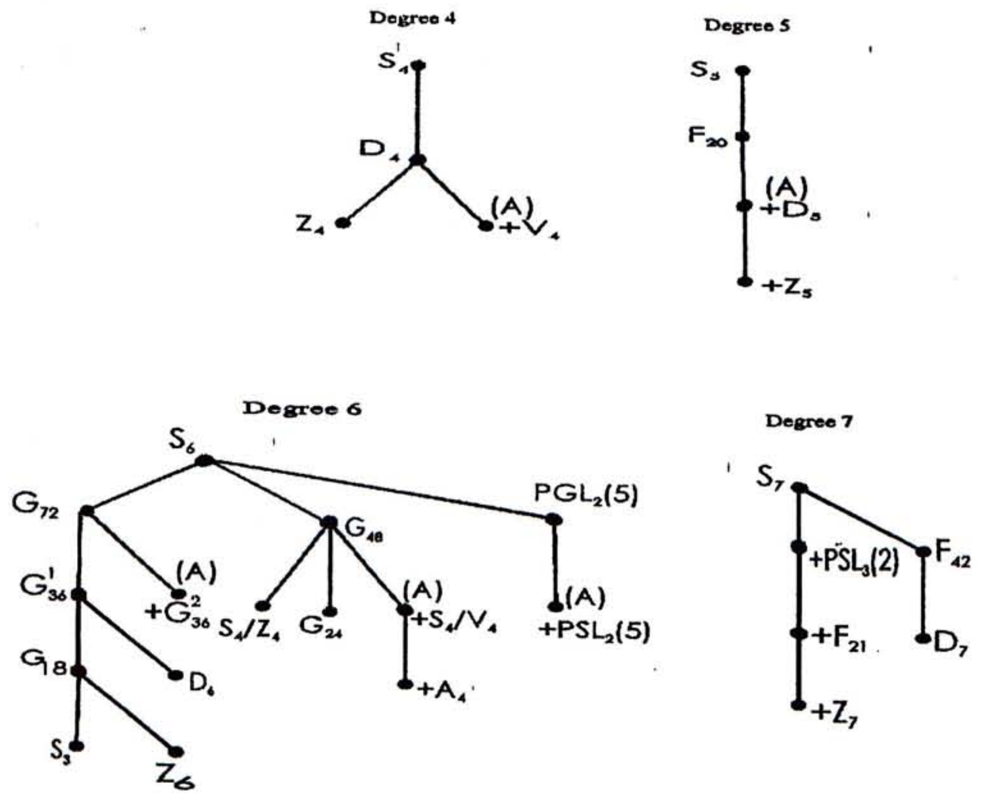


Figure 2.1: Stauduhar's method of traversing in the Transitive subgroups Lattices

The diagrams in Figure 2.1 indicate, for each degree, the order in which searching can be carried out (i.e., the order in which resolvents should be computed), so that optimal use is made of accumulated information. For these diagrams, the following conventions have been adopted: (1) at any particular node, searching proceeds from left to right on the branches leaving that node; (2) nodes isolated through examination of the discriminant are identified by a "(A)" (for alternating); an example is the node $+V_4$ in the tree for $n = 4$; (3) the alternating group A_n is not shown in tree n.

For $n = 3$, the only transitive groups are S_3 and A_3. Hence the Galois group of an irreducible polynomial of this degree is determined entirely by the value of the discriminant of the polynomial. Consequently, no tree is shown for this degree.

2.5.3 Examples

Example 1. Let $f(x) = x^6 - 42x^4 + 80x^3 + 441x^2 - 1680x + 4516$.
$f(x)$ can be shown to be irreducible over the rationals. Let Γ denote the Galois group of $f(x)$.

First of all, $\mathrm{disc}(f(x)) = -29947754653271 99186944$, clearly not a perfect square.

The roots of $f(x)$ are (approximately) $\alpha_1 = 4.392 - 1.570i$; $\alpha_2 = \overline{\alpha_1}$, . Let

$\alpha_3 = -5.490 - 0.780i$; $\alpha_4 = \overline{\alpha_3}$,

$\alpha_5 = 1.098 - 2.355i$; $\alpha_6 = \overline{\alpha_5}$.

this be the initial ordering of the roots. The maximal subgroup G_{72} of S_6 has the ten right coset representatives $\pi_1 = identity$ $\pi_6 = (2453)$, When the resolvent

$\pi_2 = (2543)$, $\pi_7 = (25)$,

$\pi_3 = (236)(45)$, $\pi_8 = (2345)$,

$\pi_4 = (25436)$, $\pi_9 = (24536)$,

$\pi_5 = (25)(34)$, $\pi_{10} = (3645)$.

$\Re_{(S_6,G_{72})}(x)$ is computed using the above data and the function $\Upsilon(x_1,\ldots,x_6) = x_1x_2x_3 + x_4x_5x_6$ given in Table 2.1, we obtain

$$\begin{aligned}\Re_{(S_6,G_{72})}(x) = \ & x^{10} + 80x^9 - 59166x^8 - 4390320x^7 \\ & +1200615393x^6 + 88076918880x^5 \\ & -7198940057856x^4 - 388801984512000x^3 \\ & +20193311991398400x^2 \\ & +595967000182784000x \\ & -4689149328097280000.\end{aligned}$$

[The actual calculation of the resolvent was made carrying 192 bits of precision. With this precision the coefficients of the resolvent were integers to within 2^{-96}.] The resolvent has a single integer root, -80, corresponding to the conjugate value Υ^{π_3}, and no repeated roots. Consequently, $\Gamma \subset \pi_3^{-1}G_{72}\pi_3$ and, after reordering the roots of $f(x)$ according to the rule $\alpha_j' = \alpha_{j^{\pi_3}}$, we know that $\Gamma \subset G_{72}$.

G_{72} has two maximal subgroups of order 36, $+G_{36}^2$ and G_{36}^1. Since $+G_{36}^2 \subset +A_6$, and since we know that $\mathrm{disc}(f(x))$ is not a perfect square, $\Gamma \not\subseteq +G_{36}^2$.
Computing the resolvent $\Re_{(G_{72},G_{36}^1)}(x)$, we find $\Re_{(G_{72},G_{36}^1)}(x) = (x + 137376)(x - 137376)$ and therefore $\Gamma \subset G_{36}^1$.

Now, G_{36}^1 contains two isomorphic versions of G_{18}, which are conjugate in G_{72} but not in G_{36}^1. Therefore, to test whether Γ is contained in some conjugate of G_{18}, one can either compute a single quartic resolvent $\Re_{(G_{72},G_{18})}(x)$, or a pair of quadratic resolvents $\Re_{(G_{36}^1,G_{18})}(x)$. Adopting the first course, G_{18} has the four right coset representatives $\{identity, (12)(45), (56), (12)(465)\}$ in G_{72}. We then find $\Re_{(G_{72},G_{18})}(x) = (x+360i)(x-360i)(\underset{\smile}{y}+648)(\underset{\smile}{y}-648)$ and we have $\Gamma \subset (56)G_{18}(56)$. Reordering the roots of $f(x)$ again, using the interchange (56), we have $\Gamma \subset G_{18}$.

Finally, G_{18} has the transitive subgroup S_3, and the resolvent associated with this subgroup turns out to be $\Re_{(G_{18},S_3)}(x) = x^3 - 1323x + 7722 = (x-33)(x-6)(x+39)$. Thus, $\Gamma \subset S_3$.

Since S_3 is a minimal transitive subgroup of S_6, $\Gamma = S_3$. Therefore, with respect to the final ordering

$$\begin{aligned} \alpha_1 &= 4.392 - 1.570i; & \alpha_2 &= -5.490 - 0.780i; \\ \alpha_3 &= 1.098 + 2.355i; & \alpha_4 &= 1.098 - 2.355i; \\ \alpha_5 &= 4.392 + 1.570i; & \alpha_6 &= -5.490 + 0.780i; \end{aligned}$$

the Galois group of $f(x)$ is {identity, $(123)(465)$, $(132)(456)$, $(14)(25)(36)$, $(15)(26)(34)$, $(16)(24)(35)$}, a group isomorphic to S_3, Table 1.1. □

Example 2. Let $f(x) = x^6 - 32x^4 + 160x^3 - 320x^2 + 384x - 256$.

Again, $f(x)$ is irreducible over the rationals, and again we let Γ denote the Galois group of $f(x)$.

First of all, $\operatorname{disc}(f(x)) = 403780252137947136 = (635437056)^2$.

An initial ordering for the roots of $f(x)$ is

$$\begin{aligned} \alpha_1 &= 1.587; & \alpha_2 &= 0.517 - 1.342i; \\ \alpha_3 &= \overline{\alpha_2} & \alpha_4 &= 2.534 + 1.927i; \\ \alpha_5 &= \overline{\alpha_4} & \alpha_6 &= -7.690. \end{aligned}$$

This time, we obtain the resolvent

$$\begin{aligned} \mathfrak{R}_{(S_6,G_{72})}(x) = {} & x^{10} + 160x^9 + 12544x^8 + 761856x^7 \\ & +35586048x^6 + 1375731712x^5 \\ & +39845888000x^4 + 935765999616x^3 \\ & +15169824489472x^2 \\ & +172073569746944x \\ & -30786325577728. \end{aligned}$$

which proves to have no integer roots. Hence Γ is not contained in any of the conjugates of G_{72}.

We next compute a resolvent with respect to the maximal subgroup G_{48} of index 15 in S_6,

$$\begin{aligned} \mathfrak{R}_{(S_6,G_{48})}(x) = {} & x^{15} + 96x^{14} + 4992x^{13} + 171520x^{12} \\ & 4546560x^{11} + 99237888x^{10} \\ & +1895104512x^9 + 31195136000x^8 \\ & +448874414080x^7 + 5653059376768x^6 \\ & +63843346677760x^5 + 606767209775104x^4 \\ & +4504321181876224x^3 + 28162341078040576x^2 \\ & +71405583642656768x + 0. \end{aligned}$$

This resolvent has the single integer root 0 corresponding to the conjugate value $\Upsilon^{(26543)}$ of the function $\Upsilon(x_1, \ldots, x_6) = x_1x_2 + x_3x_4 + x_5x_6$. After reordering the roots of $f(x)$ according to (26543), we have $\Gamma \subset G_{48}$.

Since $\operatorname{disc}(f(x))$ is a perfect square, $\Gamma \subset G_{48} \cap +A_6 = +S_4/V_4$. There is only one transitive subgroup of $+S_4/V_4$ which is not also a subgroup of G_{72}. This group is $+A_4$, and computing the resolvent $\mathfrak{R}_{(+S_4/V_4,+A_4)}(x) = x^2 - 103424$, we find that $\Gamma = +S_4/V_4$, since this resolvent has no integer roots.

Thus, with respect to the final ordering $\alpha_1 = 1.587$; $\alpha_2 = -7.690$; of
$\alpha_3 = 0.517 - 1.342i$; $\alpha_4 = \overline{\alpha_3}$
$\alpha_5 = 2.534 + 1.927i$; $\alpha_6 = \overline{\alpha_5}$
the roots of $f(x)$, the Galois group Γ of $f(x)$ is a group of 24 even permutations, isomorphic to S_4. Generators for this group are given in Table 1.1. □

Please see Appendix A for the demonstration of the author's MAPLE program on these examples—It took two seconds to obtain an answer on his cheap Pentium-133.

2.6 Quadratic Factors of Resolvents

Suppose M_1 and M_2 are non-conjugate maximal transitive subgroups of S_n, and $f(x)$ is an irreducible polynomial of degree n with Galois group Γ. It has been shown, above, that a resolvent polynomial $\Re_{(S_n,M_1)}(x)$ can be used to determine if Γ is a subgroup of some conjugate of M_1. This is done by searching, in effect, for linear factors of $\Re_{(S_n,M_1)}(x)$.
It is sometimes possible to determine if Γ is a subgroup of another maximal transitive subgroup M_2 by searching for higher degree factors of $\Re_{(S_n,M_1)}(x)$. There are obvious practical advantages to this approach if $[S_n : M_1]$ is substantially smaller than $[S_n : M_2]$. For example, S_7 has two 'maximal' transitive subgroups: $+PSL_3(2)$, of index 30, and F_{42}, of index 120. It turns out that by looking for quadratic factors of the resolvent $\Re_{(S_7,+PSL_3(2))}(x)$ of degree 30, one can avoid ever dealing with a resolvent of degree 120. (A similar situation occurs in the degree eight case.)
A difficulty is encountered, however, in using quadratic factors of resolvents. Under certain circumstances a quadratic factor of $\Re_{(S_n,M_1)}(x)$ will guarantee that Γ is a subgroup of some conjugate of M_2, but will fail to specify exactly *which* conjugate. To put it another way, it is sometimes impossible to extract from the quadratic factor the information necessary to reorder the roots of $f(x)$. As it turns out, this unpleasant situation can always be avoided in the degree seven and degree eight cases. Even so, the procedure for obtaining reordering information from a quadratic factor is somewhat complicated and will not be discussed here, see [Sta69].

2.7 Comment

Stauduhar gave a method for computing Galois groups of polynomials f in $\mathbb{Z}[x]$. He used approximations to the roots and worked down from the symmetric group, S_n, through the transitive subgroups, to identify $Gal_{\mathbb{Q}}(f)$ as a transitive group of degree up to 8.

- The required precision of the roots can be very large. Stauduhar himself reported calculations using 192 bits (≈ 60 digits) approximations to the roots of a degree 6 polynomial. In Appendix B.1[2], it is stated that calculations for certain degree 11 and 12 polynomials require thousands of digits of precision!
- Traversing the subgroup lattice from the symmetric group down to the Galois group of the polynomial can take substantial time.
- Much tabulated information is necessary and should store into the computer.

In fact, this method becomes *unwieldy* when there are many transitive subgroups, as there are when the degree is, say 12, there are about 300 such subgroups. A reason for this is that of storing or computing polynomial invariants and data for traversing the transitive subgroups to identify $Gal_{\mathbf{Q}}(f)$.

Chapter 3

Factoring Polynomials Quickly

While only linear (or quadratic) factors of resolvent polynomials are required in Stauduhar's method of computing Galois groups, Soicher & McKay have developed a method based on the complete factorization of resolvent polynomials, [SM85].

Exact and efficient polynomial factorization algorithms have developed since 1968, [Ber68] and [Zas69]. Perhaps part of the motivation of such **Algorithmic** *pure* **Mathematics** was due to the advent of **computing technology**. Polynomial factorization has been one of the important topic of a newly emerging field called *Symbolic and algebraic Computation,* or *Computer Algebra, Computational Algebra.* Textbooks in this area have been published since late eighties, [CAbks]. See Appendix B.2 for more information of SAC.

The principal references for this chapter are "Factoring Polynomials Quickly, Susan Landau, Notices of the American Mathematical Society, [Special Article Series], Vol. 34, No. 1 (1987), pp. 3-8," [Lan87] and also the Computer Algebra textbooks, [CAbks].

3.1 History

3.1.1 From Feasibility to Fast Algorithms

Computer science has a way of reaching back to the origins of mathematics: arithmetic and computations. Finding primes, factoring integers—the old problems recur. This is the story of another: factoring polynomials into irreducibles over the rationals. The question of computability was answered centuries ago, but an efficient solution was arrived at only recently.

The problem has a venerable history: Isaac Newton tried his hand at it, and saw a way to find linear and quadratic divisors. In 1793, Friedrich von Schubert, an astronomer, generalized Newton's technique and determined all factors.

Von Schubert's idea was to calculate $f(1), f(2), \ldots, f(n)$ where n is the degree of $f(x)$, the polynomial in question. Then factor the $f(i)$. If $d(1), d(2), \ldots, d(n)$ is a particular sequence of divisors of $f(1), f(2), \ldots, f(n)$, then the $d(i)$ define a potential factor of $f(x)$, one which can easily be found by interpolation. The complete factorization of $f(x)$ can be determined in this way.

Von Schubert's technique would satisfy logicians—it shows that the question is **decidable**—but not someone who really wanted to factor. It's too slow. The algorithm requires at least 2^n steps to show that a polynomial of degree n is irreducible. It is infeasible for factoring polynomials of degree 20 or more. The issue is **complexity** : how long must a factoring algorithm take?

Computer scientists believe that **polynomial time** solutions—algorithms which take a polynomial number of steps in the size of the input—are the only feasible kind. Von Schubert's algorithm is exponential in the degree of the polynomial.

Actually there are two parts which contribute to a polynomial's size: its degree, and the number of bits needed to express its coefficients. Considering first the degree, there are two possible measures for univariate polynomials: the sparse notation, in which '$x^n - 1$' would be written as $(n, 1; 0, -1)$, taking $O(\log n)$ bits, and the dense, in which the same polynomial would be $(1, 0, 0, \ldots, 0, -1)$, requiring $O(n)$ bits. Since a polynomial of degree n may have as many as n factors, the more natural dense notation has been accepted as the 'right' measure of a polynomial's size.

As for the coefficients, suppose $f(x) = f_n x^n + \cdots + f_1 x + f_0$. Then if $g(x)$ is a factor of $f(x)$ of degree m, the ith coefficient of $g(x)$ is less than $\binom{m}{i}(\sum_{i=0}^{n} f_i^2)^{\frac{1}{2}}$. (We will denote $(\sum_{i=0}^{n} f_i^2)^{\frac{1}{2}}$ by $|f(x)|$.) This means that $g(x)$ may be expressed in a number of bits that is polynomial in the size of $f(x)$, and since there are at most n factors of $f(x)$, a complete factorization can be written down in polynomial space. In theory, at least, a polynomial time solution to the polynomial factorization problem is possible.

Certain parts of the problem are easy. For example, it has long been known how to pull out multiple factors of a polynomial. Suppose $f(x)$ has an irreducible factor, $g(x)$, of multiplicity k. Then $g^{k-1}(x)$ divides the $\gcd(f(x), f'(x))$ (while $g^k(x)$ does not). The gcd is quickly computed by the subresultant version of the Euclidean

Algorithm, which also avoids any coefficient blowup. Instead of factoring $f(x)$, one factors $\frac{f(x)}{\gcd(f(x),f'(x))}$ and $\gcd(f(x), f'(x))$. Iterating this procedure means that only squarefree polynomials need be factored.

If you can't factor polynomials over the rationals, why not try factoring over smaller fields? In 1967 Berlekamp discovered **deterministic** and **probabilistic** methods to factor squarefree polynomials mod p. Sometime later Rabin created an even simpler version of the probabilistic algorithm with expected running time of $O(n^3 \log p)$ steps for factoring a squarefree polynomial of degree n. Meanwhile Zassenhaus countered with Hensel's Lemma (which explains how to lift a squarefree factorization mod m to one mod m^2). Zassenhaus's idea was to factor a polynomial over the integers by first factoring over a suitable prime modulus (one which does not divide the discriminant of the polynomial, and thus keeps $f(x)$ squarefree mod p), then raising that to a factorization mod p^2 , then mod p^4, and so on... until the modulus was large enough—though still polynomial sized in terms of the original problem—to lift to a factorization over the integers.

$$\begin{aligned} \text{Consider} \quad f(x) &= x^4 - 8x^3 + x^2 - 24x - 6 \\ &= (x^2 + 2x + 3)(x^2 + 3) \qquad (\text{mod } 5) \\ &= (x^2 - 8x - 2)(x^2 + 3) \qquad (\text{mod } 25) \\ &= (x^2 - 8x - 2)(x^2 + 3) \qquad (\mathbb{Z}) \end{aligned}$$

It's a good idea, and it works well much of the time. Difficulties arise because polynomials may have a finer factorization in $\mathbb{Z}/p\mathbb{Z}$ than they do in $\mathbb{Z}$, and the ensuing problem of combining factors mod p to determine factors in $\mathbb{Z}$ may be costly. For certain polynomials, it's a disaster. Swinnerton-Dyer pointed out one such: the polynomial whose roots are $\pm\sqrt{2} \pm \sqrt{3} \pm \cdots \pm \sqrt{p_n}$, where p_n is the nth prime. This polynomial factors into either linear or quadratic polynomials mod m for any modulus m one might choose, yet it is a polynomial which is irreducible over the integers. To discover its irreducibility, one must look at all possible combinations of mod m factors, an exponential nightmare. The class of polynomials which raise difficulties like this is small—essentially those with 'nice' Galois groups—and so, despite its **worst-case** exponential running time, Berlekamp-Hensel became the factoring algorithm of choice during the 1970s.

Other classic problems fell into polynomial time, yet polynomial factorization remained stubbornly exponential. Pieces were chipped away: Weinberger showed that under the assumption of the Extended Riemann Hypothesis, one could test irreducibility in polynomial time, and Cantor showed that—ERH or no—irreducible polynomials have short proofs of that fact. Cantor and Zassenhaus gave a new, improved mod p factoring routine. The central question, however, of how to avoid the **exponential** increase which arose from looking at combinations of mod p factors remained.

* * *

Suppose p does not divide the discriminant of $f(x)$, and $h(x)$ is an irreducible factor of $f(x)$ in $\mathbf{Z}/p\mathbf{Z}$. One approach is to consider the unique factor of $f(x)$ in $\mathbf{Z}$ which $h(x)$ divides (which exists since $f(x)$ is squarefree in $\mathbf{Z}/p\mathbf{Z}$). Call it $h_0(x)$. The issue then, is to efficiently determine $h_0(x)$ from $h(x)$.

It was classical mathematics which provided the tool: Minkowski lattices. Lattices are simple generalization of $\mathbf{Z}^n$, but they are the key to important **theorems**, and now, to important **algorithms**. Hendrik Lenstra introduced them into computer science with an elegant polynomial time solution for integer linear programming with a bounded number of variables.

Let $b_1, b_2, \ldots, b_n$ be a set of linearly independent vectors in $\mathbf{Z}^n$. Then the n-dimensional lattice $L \in \mathbf{Z}^n$ with basis b_i is the set of integral linear combinations of the b_i. Several natural questions immediately arise: Given a basis of a lattice, how does one quickly determine an orthogonal one? How does one find short vectors? Is there a fast algorithm for determining the shortest vector in a lattice?

It was this last question which Lenstra answered for fixed dimension in 1981. Shortly afterwards Lovász found a polynomial time basis reduction algorithm which computes, among other things, a nonzero basis vector b such that $|b| \leqslant 2^{\frac{n-1}{2}} |x|$ for all nonzero vectors x in the lattice L. Arjen Lenstra, Hendrik Lenstra and Lovász (hence L^3) combine ideas to create a polynomial time polynomial factorization algorithm, [LLL82].

The L^3 algorithm builds an m-dimensional lattice L whose vectors are polynomials in x determined by $h(x)$, an irreducible factor of $f(x)$ mod p. (Recall that $h(x)$ determines $h_0(x)$, a unique irreducible factor of $f(x)$ in $\mathbf{Z}[x]$.) If $h_0(x)$ is of degree m then it will be found by the Lovász basis reduction algorithm, since any vector in L which is linearly independent of $h_0(x)$ will be $2^{\frac{m}{2}}$ times longer than it. The proof of this fact is surprisingly simple, and we present it here below. However we omit the proof that basis reduction can be done quickly, and instead refer the interested reader to the original paper (or Computer Algebra Textbooks, [CAbks].) Note that the bounds we show are less than optimal, and are chosen for the sake of a simpler argument.

Let the polynomial to be factored be $f(x) \in \mathbf{Z}[x]$, and suppose that it is primitive (which means that the gcd of its coefficients is 1), squarefree and of degree n. Factor $f(x)$ mod p, where p is chosen so that $p \nmid \operatorname{disc}(f(x))$. (One can find such a p which will be polynomial size in n and $|f(x)|$.) Pick an irreducible factor of $f(x)$ in $\mathbf{Z}/p\mathbf{Z}$, say $h(x)$ of degree l ($l < n$). Now raise the factorization to one mod p^k (where k is chosen so that $p^k \geqslant (2^{\frac{n}{2}} \binom{2n}{n} |f(x)|^2)^n$), using Hensel lifting, and let $\tilde{h}(x)$ be the image of $h(x)$. We will find $h_0(x)$, the unique irreducible factor of $f(x)$ in $\mathbf{Z}[x]$ which $\tilde{h}(x)$ divides.

We assume $h_0(x)$ has degree m. Define a lattice with basis as follows: $b_i = p^k x^i$ for $0 \leqslant i < l$, and $b_i = \tilde{h}(x)x^{i-l}$ for $l \leqslant i \leqslant m$. Now if we think of the polynomials as vectors, with the coefficient of x^i as the $(i+1)$st coordinate, then the b_i are linearly independent, since they form an upper diagonal matrix.

But since $h_0(x)$ is a factor of $f(x)$, we have that $|h_0(x)| \leqslant \binom{2m}{m}^{\frac{1}{2}} |f(x)| \leqslant \binom{2n}{n}^{\frac{1}{2}} |f(x)|$.

Remember that k was chosen to satisfy $p^k \geqslant (2^{\frac{n}{2}} \binom{2n}{n} |f(x)|^2)^n$. It is clear that $h_0(x)$ is in the lattice, and we claim that any vector of L which is linearly independent of $h_0(x)$ is $2^{\frac{m}{2}}$ times longer than $h_0(x)$. Thus $h_0(x)$ can be determined by the basis reduction algorithm.

We prove the claim. Let $g(x)$ be any element of L which is linearly independent of $h_0(x)$. Then $\gcd(g(x), h_0(x)) = 1$ in $\mathbf{Z}[x]$. Thus the polynomials $h_0(x)x^i, 0 \leqslant i < \deg(g(x))$, and $g(x)x^j, 0 \leqslant j < \deg(h_0(x))$, are linearly independent. Consider the resultant, $\det(R)$, of $h_0(x)$ and $g(x)$. By Hadamard's inequality and the fact that the degree of $g(x) \leqslant m$, we have $\det(R) \leqslant |h_0(x)|^n |g(x)|^n$.

Now $\tilde{h}(x)$ divides both $h_0(x)$ and $g(x)$ modulo p^k, since both polynomials are elements of L. In particular, $\det(R)$ must be zero modulo p^k. But because $\det(R) \neq 0$, we know that $p^k \leqslant \det(R)$. Then $|g(x)| \geqslant 2^{\frac{n}{2}} \binom{2n}{n} |f(x)|$, thus proving the claim.

The polynomial $h_0(x)$ has degree between l and n, the degree of $f(x)$, so that the basis reduction algorithm to determine $h_0(x)$ from $h(x)$ is done at most n times. As mentioned earlier, one can pick bounds somewhat more carefully, in which case the L^3 algorithm takes $O(n^{9+\epsilon} + n^{7+\epsilon} |f(x)|)$ steps to factor $f(x)$ (for any $\epsilon > 0$).

Randomness is central to computer algorithms. It can mean an exponential speedup for an algorithm, as in the Solovay-Strassen primality test, which is an $O(n)$ deterministic algorithm, and an $O(\log n)$ probabilistic one. Many cryptographic schemes rely on random bits. In a certain sense, the roots of polynomials are not random. If an approximate root of an integer polynomial is given, along with a bound on the degree and coefficient size of its minimal polynomial, then in polynomial time, the minimal polynomial can be determined. Therein lies another factoring scheme.

In particular, if $f(x)$, a monic polynomial over $\mathbf{Z}$ is given, a root α of $f(x)$ can be approximated in polynomial time. The lattice algorithm can be used to determine

$g(x)$, the minimal polynomial of α. Of course, $g(x)$ is a factor of $f(x)$. This variant of the algorithm was discovered by Kannan, Arjen Lenstra, and Lovász, and independently by Schönhage. Its running time is identical to that of L^3.

Arjen Lenstra generalized the L^3 algorithm to a variety of situations: for algebraic number fields, for multivariate polynomials over finite fields, fields of characteristic p, and for $\mathbf{Q}$. A number of others, including von zur Gathen, Kaltofen, Trager, Landau, Chistov, and Grigoriev, used different techniques to also find polynomial time solutions for diverse factoring problems.

$\star \quad \star \quad \star$

Before L^3, Arjen Lenstra had seen a connection between lattices and factorization over algebraic number fields; after L^3 he gave a polynomial time method for factoring a monic polynomial $f(x)$ in $\mathbf{Q}(\alpha)[x]$, where α satisfies a monic irreducible polynomial, $g(t)$, over $\mathbf{Q}$. The technique is the same as for L^3: determine a factorization of $f(x)$ over a finite field (in this case, $\mathbf{Z}[t]/(p, H(t))$, where $g(t)$ is squarefree in $\mathbf{Z}/p\mathbf{Z}$ and $H(t)$ is one of its irreducible factors in that field), extend to a factorization over an appropriate ring $\mathbf{Z}[t]/(p^k, H(t))$ for a large enough k, and use basis reduction to factor. The details are messy, and we will not present them.

In some ways though, Kronecker had everyone beat by a century. In 1882, he proposed using norms for factoring polynomials over algebraic number fields. The idea is simple. Again let $f(x)$ be a squarefree polynomial of degree n in $\mathbf{Q}(\alpha)[x]$, where α satisfies $g(t)$, an irreducible monic polynomial of degree m over $\mathbf{Q}$. One can view the polynomial $f(x)$ as a polynomial in x and α or, equivalently, in x and t. We define the $Norm_{\mathbf{Q}(\alpha)/\mathbf{Q}}(f(x)) = \prod f(x, \alpha_i)$, where the product is over all conjugates, α_i, of α. We can compute it quickly since it is the $Resultant_t(g(t), f(x))$. Then if $F(x) = Norm_{\mathbf{Q}(\alpha)/\mathbf{Q}}(f(x))$, $F(x)$ is a polynomial in $\mathbf{Q}[x]$ of degree mn. If it is squarefree and equal to $\prod_{i=1}^{r} f_i(x)$, where the $f_i(x)$ are irreducible in $\mathbf{Q}[x]$, then $f(x) = \prod_{i=1}^{r} \gcd(f_i(x), f(x))$. As long as $f(x)$ is squarefree to begin with, one can always 'twiddle' the polynomial so as to ensure that $F(x)$ will be. Landau, building on work of Trager, showed that all these reductions can be performed in polynomial time. Thus one can factor over algebraic number fields in polynomial time using norms.

3.1.2 Implementations on Computer Algebra Systems

When you build a better mousetrap, it's important to remember the mouse. Almost all the algorithms described above have been implemented; how good are they?

Polynomial time factoring algorithms developed since L^3 have depended upon the lattice algorithm. The old Berlekamp-Hensel algorithm is exponential. Yet for many factoring problems, the practical algorithm is Berlekamp-Hensel. It's faster.

For univariates over $\mathbb{Q}$, the L^3 algorithm requires at least $O(n^7 + n^4 \log|f(x)|)$ steps. Berlekamp-Hensel is almost always quicker on univariate polynomials. Berlekamp-Hensel runs into trouble only when the Galois group is 'nice', which is loosely defined as a permutation group on n elements which is small compared to S_n. Most—in a strong sense—irreducible polynomials do not have nice Galois groups. It is important to keep in mind, however, that the polynomials one might choose to factor are often not typical.

Another situation in which lattices provide a good algorithm is for algebraic number fields. This is an early algorithm of Arjen Lenstra's which is practical (despite a worst-case exponential running time) because it avoids computations on algebraic numbers until it searches for the true factors.

And what happens in practice? MACSYMA, the workhorse of symbolic computation programs, uses Berlekamp-Hensel to factor univariate polynomials, and a variation of Wang's original algorithm for multivariates. It's impressive; it factored $x^{51}+48x^{43}+30x^{31}+81x^{25}+47x^{35}+536x^{23}+81x^{17}+209x^{11}+891x^5+12x^{26}+564x^{18}+228x^6+972$
$= \left(x^{25}+x^{17}+11x^5+12\right)\cdot\left(x^{26}+47x^{18}+19x^6+81\right)$ in one minute on a VAX, and $(ax^3y^2+bx^2y^3+dfxy+11)\cdot(a^3b^2e^4+a^2b^3+3ab+2)$
$$(abc^2y+abx+ac^2f^2+4)\cdot(abc^3d^2+d^3xy^2+dex^2y+5)$$
—a polynomial with two hundred and fifty terms— in only fifteen seconds. What's theoretically slow can still be practically fast.

Interesting though, the two examples above both took only 1 second for its factorization by "MAPLE V (see [MAPLE]) release 3.0, 1994 for Windows 3.1" on the author's private Pentium-133.

Notation

Throughout the remaining sections in this chapter we let I be a ufd—unique factorization domain and K the field of fractions of I.

Definition 43 *A univariate polynomial $f(x)$ over the ufd I is* **primitive** *iff there is no prime in I which divides all the coefficients in $f(x)$.*

Definition 44 *Up to multiplication by units we know that every polynomial $a(x) \in K[x]$ can be decomposed uniquely into $a(x) =$cont(a)$\cdot$ pp(a), where cont(a) $\in K$ and* pp(a) *is a primitive polynomial in $I[x]$. cont(a) is the* **content** *of $a(x)$,* pp(a) *is the* **primitive part** *of $a(x)$.*

Remark: From Gauß's Lemma we will usually assume $a(x) \in I[x]$ and develop methods for working directly in the ufd I.

3.2 Squarefree factorization

By just computing gcds—greatest common divisors, we can produce a so-called squarefree factorization of a polynomial, i.e., a partial solution to the problem of factoring polynomials. Squarefree factorization is only a first step in the complete factorization of a polynomial. However, it is relatively inexpensive and it is a prerequisite of many factorization algorithms.

Throughout this chapter let K be a computable field generated as $Q(I)$, the field of fractions of I, where I is a ufd—unique factorization domain. Whenever I is a ufd, then also $I[x]$ is a ufd from Gauß's lemma.

Definition 45 *A polynomial $a(x_1, \ldots, x_n)$ in $I[x_1, \ldots, x_n]$ is squarefree iff every nontrivial factor $b(x_1, \ldots, x_n)$ of a (i.e., b not similar to a and not a constant) occurs with multiplicity exactly 1 in a.*

By Gauß's lemma we know that for *primitive* polynomials the squarefree factorizations in $I[x]$ and $K[x]$ are the same.

There is a simple criterion for deciding squarefreeness.

Theorem 46 *Let $a(x)$ be a nonzero polynomial in $K[x]$, where* $\mathrm{char}(K) = 0$ *or $K = \mathbb{Z}_p$ for a prime p. Then $a(x)$ is* **squarefree** *if and only if* $\gcd(a(x), a'(x)) = 1$. *($a'(x)$ is the derivative of $a(x)$.)*

Proof. If $a(x)$ is not squarefree, i.e., for some non-constant $b(x)$ we have $a(x) = b(x)^2 \cdot c(x)$, then $a'(x) = 2b(x)b'(x)c(x) + b^2(x)c'(x)$. So $a(x)$ and $a'(x)$ have a non-trivial gcd.
On the other hand, if $a(x)$ is squarefree, i.e., $a(x) = \prod_{i=1}^n a_i(x)$, where $a_i(x)$ are pairwise relatively prime irreducible polynomials, then

$$a'(x) = \sum_{i=1}^{n}(a_i'(x) \prod_{\substack{j=1\\ j\neq i}}^{n} a_j(x)).$$

Now it is easy to see that none of the irreducible factors $a_i(x)$ is a divisor of $a'(x)$. $a_i(x)$ divides all the summands of $a'(x)$ except the i-th. This finishes the proof for characteristic 0. In $\mathbb{Z}_p[x]$, $a_i'(x)$ cannot vanish, for otherwise we could write $a_i(x) = b(x^p) = b(x)^p$ for some $b(x)$, and this would violate our assumption of squarefreeness. Thus, $\gcd(a(x), a'(x)) = 1$. ■

The problem of squarefree factorization for $a(x) \in K[x]$ consists of determining the squarefree pairwise relatively prime polynomials $b_1(x), \ldots, b_s(x)$, such that $a(x) = \prod_{i=1}^s b_i(x)^i$.

Definition 47 *The representation of a as*

$$a = \prod_{i=1}^{s} b_i(x)^i$$

is called the squarefree factorization of a.

In characteristic 0 (e.g., when $a(x) \in \mathbb{Z}[x]$), we can proceed as follows. We set $a_1(x) := a(x)$ and $a_2(x) := \gcd(a_1, a_1')$. Then $a_2(x) = \prod_{i=1}^s b_i(x)^{i-1} = \prod_{i=2}^s b_i(x)^{i-1}$, $c_1(x) := a_1(x)/a_2(x) = \prod_{i=1}^s b_i(x)$ contains every squarefree factor exactly once. Now we set $a_3(x) := \gcd(a_2, a_2') = \prod_{i=3}^s b_i(x)^{i-2}$, $c_2(x) := a_2(x)/a_3(x) = \prod_{i=2}^s b_i(x)$ contains every squarefree factor of multiplicity ≥ 2 exactly once. So we have $b_1(x) = c_1(x)/c_2(x)$.
Next we set $a_4(x) := \gcd(a_3, a_3') = \prod_{i=4}^s b_i(x)^{i-3}$, $c_3(x) := a_3(x)/a_4(x) = \prod_{i=3}^s b_i(x)$. So we have $b_2(x) = c_2(x)/c_3(x)$.
Iterating this process until $c_{s+1}(x) = 1$, we ultimately get the desired squarefree factorization of $a(x)$. This process for computing a squarefree factorization is summarized in SQFR_FACTOR.

Algorithm SQFR_FACTOR (in: a; out: F);
[a is a primitive polynomial in $\mathbb{Z}[x]$,
$F = [b_1(x), \ldots, b_s(x)]$ is the list of squarefree factors of a.]

```
1. $F := []$;
   $a_1 := a$;
   $a_2 := \gcd(a_1, a_1')$;
   $c_1 := a_1/a_2$;
   $a_3 := \gcd(a_2, a_2')$;
   $c_2 := a_2/a_3$;
   $F := \text{CONS}(c_1/c_2, F)$;
2. while $c_2 \neq 1$ do
   { $a_2 := a_3$; $a_3 := \gcd(a_3, a_3')$;
   $c_1 := c_2$; $c_2 := a_2/a_3$;
   $F := \text{CONS}(c_1/c_2, F)$ };
3. $F := \text{INV}(F)$; return.
```

If the polynomial $a(x)$ is in $\mathbb{Z}_p[x]$, the situation is slightly more complicated. First we determine $d(x) = \gcd(a(x), a'(x))$.
If $d(x) = 1$, then $a(x)$ is squarefree and we can set $a_1(x) = a(x)$ and stop.
If $d(x) \neq 1$ and $d(x) \neq a(x)$, then $d(x)$ is a proper factor of $a(x)$ and we can carry out the process of squarefree factorization both for $d(x)$ and $a(x)/d(x)$.
Finally, if $d(x) = a(x)$, then we must have $a'(x) = 0$, i.e., $a(x)$ must contain only terms whose exponents are a multiple of p. So we can write $a(x) = b(x^p) = b(x)^p$ for some $b(x)$, and the problem is reduced to the squarefree factorization of $b(x)$.

An algorithm for squarefree factorization in $\mathbb{Z}_p[x]$ along these lines is presented in Akritas(1989), [CAbks], namely PSQFFF.

Algorithm PSQFFF (Polynomial Squarefree factorization over a finite field)
Input: $a(x)$, a nonconstant monic polynomial in $\mathbb{Z}_p[x]$, $p > 0$ is prime.
Output: $b_1(x), \ldots, b_e(x)$ and e such that $a(x) = \prod_{1 \leq i \leq e} b_i(x)^i$ is the squarefree factorization of $a(x)$.

```
1. [Initialize.] $k := 0$; $m := 1$; $e := 0$.
2. [Main loop.] $j := 1$; $a_2(x) := \gcd(a(x), a'(x))$; $c_1(x) := a(x)/a_2(x)$;
   if $c_1(x) = 1$, then go to 7.
3.     [Update.] $e' := jm$; if $e' > e$, then
       do $\{b_{e+1}(x) := b_{e+2}(x) := \cdots b_{e'-1}(x) := 1; e := e'\}$.
4.     [Compute $b_{e'}(x)$.] $c_2(x) := \gcd(a_2(x), c_1(x))$; $b_{e'}(x) := c_1(x)/c_2(x)$.
5.     [Update.] If $c_2(x) \neq 1$, then do $\{a_2(x) := a_2(x)/c_2(x)$; $c_1(x) := c_2(x)$;
       $j := j + 1$; go to 3.$\}$
6. [Finished?] If $a_2(x) = 1$, then exit.
7. [$a_2'(x) = 0$] $a(x) := (a_2(x))^{\frac{1}{p}}$; $k := k + 1$; $m := mp$; go to 2.
```

Throughout the execution of PSQFFF we have $m = p^k$, and whenever we arrive at step 6, the value of e is the largest index i such that p^k does divide i and $a(x)$ has a nonconstant factor to power i. We also assume that the polynomials $a_2(x)$ and $c_2(x)$ calculated in steps 2 and 4, respectively, are monic.

3.3 Factorization over finite fields

We will reduce the computation of the factors of an integral polynomial to the computation of the factors of the polynomial modulo a prime number. So we have to investigate this problem first, i.e., we consider the problem of factoring a polynomial $a(x) \in \mathbb{Z}_p[x]$, p a prime number. W.l.o.g. we may assume that $\mathrm{lc}(a(x)) = 1$, i.e., $a(x)$ is monic.

In the sequel we describe E. R. Berlekamp's (1968) algorithm, [Ber68] for factoring squarefree univariate polynomials in $\mathbb{Z}_p[x]$. Throughout this section let $a(x)$ be a squarefree polynomial of degree n in $\mathbb{Z}_p[x]$, p a prime number, having the following factorization into irreducible factors $a(x) = \prod_{i=1}^{r} a_i(x)$.

By the Chinese Remainder theorem for polynomials, for every choice of $s_1, \ldots, s_r \in \mathbb{Z}_p$ there exists a uniquely determined polynomial $v(x) \in \mathbb{Z}_p[x]$ such that

$$\begin{aligned} &v(x) \equiv s_i \mod a_i(x) \text{ for } 1 \leq i \leq r, \text{ and} \\ &\deg(v) < \deg(a_1) + \cdots + \deg(a_r) = n. \end{aligned} \tag{3.1}$$

In (3.1) it is essential that a is squarefree, i.e., the a_i's are relatively prime.

Lemma 48 *For every $a_i, a_j, i \neq j$, there exist $s_1, \ldots, s_r \in \mathbb{Z}_p$ such that the corresponding solution $v(x)$ of (3.1) generates a factorization $b \cdot c$ of a with $a_i \mid b$ and $a_j \mid c$.*

Proof. If $r = 1$ there is nothing to prove. So assume $r \geq 2$. Choose $s_i \neq s_j$ and the other s_k's arbitrary. Let v be the corresponding solution of (3.1). Then $a_i(x) \mid \gcd(a(x), v(x) - s_i)$ and $a_j(x) \nmid \gcd(a(x), v(x) - s_i)$. ■

So we could solve the factorization problem over $\mathbb{Z}_p$, if we could get a complete overview of the solutions $v(x)$ of (3.1) for all the choices of $s_1, \ldots, s_r \in \mathbb{Z}_p$. Fortunately this can be achieved by linear algebra methods.

If $v(x)$ satisfies (3.1), then $v(x)^p \equiv s_i^p = s_i \equiv v(x) \mod a_i(x)$ for $1 \leq i \leq r$. So we have

$$v(x)^p \equiv v(x) \mod a(x) \text{ and } \deg(v) < n \tag{3.2}$$

Every solution of (3.1) for some $s_1, \ldots, s_r$ solves (3.2). But what about the converse of this implication? Is every solution of (3.2) also a solution of (3.1) for some

$s_1, \ldots, s_r$? From the fact that $GF(p)$ is the splitting field of $x^p - x$, theorem 25, (cf. Fermat's little theorem), we get that $v(x)^p - v(x) = (v(x) - 0)(v(x) - 1) \ldots (v(x) - (p-1))$. So if $v(x)$ satisfies (3.2), then $a(x)$ divides $v(x)^p - v(x)$ and therefore every irreducible factor $a_i(x)$ must divide one of the factors $v(x) - s$ of $v(x)^p - v(x)$. Thus, every solution of (3.2) is also a solution of (3.1) for some $s_1, \ldots, s_r$. In particular, there are exactly p^r solutions of (3.2).

By Fermat's little theorem and the freshmen's dream[1], the solutions of (3.2) constitute a vector space over $\mathbb{Z}_p$. So we can get a complete overview of the solutions of (3.2), if we can compute a basis for this vector space.

Let the $(n \times n)$-matrix Q_a over $\mathbb{Z}_p$, $Q_a = Q = \begin{pmatrix} q_{0,0} & \cdots & q_{0,n-1} \\ \vdots & & \vdots \\ q_{n-1,0} & \cdots & q_{n-1,n-1} \end{pmatrix}$, be defined by $x^{pk} \equiv q_{k,n-1}x^{n-1} + \cdots + q_{k,1}x + q_{k,0} \bmod a(x)$ for $0 \leq k \leq n-1$. That is, the entries in the k-th row of Q are the coefficients of $\mathrm{rem}(x^{pk}, a(x))$. Using the representation of $v(x) = v_{n-1}x^{n-1} + \cdots + v_0$ as the vector $(v_0, \ldots, v_{n-1})$, we have $v \cdot Q = v \Leftrightarrow v(x) = \sum_{j=0}^{n-1} v_j x^j = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} v_k \cdot q_{k,j} \cdot x^j \equiv \sum_{k=0}^{n-1} v_k x^{pk} = v(x^p) = v(x)^p \bmod a(x)$.

We summarize all these results in the following theorem.

Theorem 49 *With the notation used above, a polynomial $v(x) = v_{n-1}x^{n-1} + \cdots + v_1 x + v_0$ in $\mathbb{Z}_p[x]$ solves (3.2) if and only if the vector $(v_0, \ldots, v_{n-1})$ is in the nullspace of the matrix $Q - I$ (I is the identity matrix of dimension n), i.e., $v \cdot (Q - I) = (0, \ldots, 0)$.*

Now we are ready to formulate **Berlekamp's algorithm** for factoring squarefree univariate polynomials in $\mathbb{Z}_p[x]$.

[1] At Hong Kong here, some students of the science stream may study the Binomial Theorem in Secondary Four—at the age of 16.

```
Algorithm FACTOR_B(in: $a(x), p$; out: $F$);
[$p$ is a prime number, $a(x)$ is a squarefree polynomial in $\mathbb{Z}_p[x]$,
$F$ is the list of prime factors of $a$.]
  1. form the $(n \times n)$-matrix $Q$ over $\mathbb{Z}_p$, where the $k$-th line $(q_{k,0}, \ldots, q_{k,n-1})$
     of $Q$ satisfies
     $\mathrm{rem}(x^{pk}, a(x)) = q_{k,n-1}x^{n-1} + \cdots + q_{k,0}$, for $0 \le k \le n-1$;
  2. by column operations transform the matrix $Q - I$ into (e.g., lower-right)
     triangular form;
     from the triangular form read off the rank $n - r$ of the matrix $Q - I$;
       [There are exactly $r$ linearly independent solutions $v^{[1]}, \ldots, v^{[r]}$ of
       $v \cdot (Q - I) = (0, \ldots, 0)$. Let $v^{[1]}$ be the trivial solution $(1,0,...,0)$.
       So (after interpretation of vectors as polynomials) there are $p^r$
       solutions $t_1 v^{[1]} + \cdots + t_r v^{[r]}$ of (3.2),
       and therefore $r$ irreducible factors of $a(x)$.]
  3. if $r = 1$, then $a(x)$ is irreducible and we set $F := [a]$;
     otherwise, compute $\gcd(a(x), v^{[2]}(x) - s)$ for $s \in \mathbb{Z}_p$ and put the
     factors of $a$ found in this way into the list $F$;
     as long as $F$ contains fewer than $r$ factors, choose the next $v^{[k]}(x)$,
     $k = 3, ..., r$, and compute $\gcd(f(x), v^{[k]}(x) - s)$ for $f$ in $F$;
     add the factors found in this way to $F$;
       [ultimately, $F$ will contain all the factors of $a(x)$,
       Lemma 48.]
  4. return.
```

Example. Let us use **FACTOR_B** for factoring the polynomial $a(x) = x^5 + x^3 + 2x^2 + x + 2$ in $\mathbb{Z}_3[x]$.

First we have to check for squarefreeness. $a'(x) = 2x^4 + x + 1$, so $\gcd(a, a') = 1$ in $\mathbb{Z}_3[x]$ and therefore $a(x)$ is squarefree.

The rows of the (5×5)-matrix Q are the coefficients of $x^0, x^3, x^6, x^9, x^{12}$ modulo $a(x)$. So $Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 2 & 2 \\ 2 & 0 & 2 & 1 & 1 \end{pmatrix}$. $Q - I$ can be transformed into the triangular form $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$. We read off $r = 2$, i.e., there are 2 irreducible factors of $a(x)$. The null-space of $Q - I$ is spanned by $v^{[1]} = (1,0,0,0,0)$ and $v^{[2]} = (0,0,2,1,0)$.

Now we get the factors by appropriate gcd computations:
$\gcd(a(x), v^{[2]}(x) + 2) = x^2 + x + 2$,
$\gcd(a(x), v^{[2]}(x) + 1) = x^3 + 2x^2 + 1$. □

The basic operations in **FACTOR_B** are the setting up and solution of a system of linear equations and the gcd computations for determining the actual factors. The complexity of **FACTOR_B** is proportional to $n^3 + prn^2$, where n is the degree of the polynomial, cf. [CAbks].

3.4 Factorization over the integers

From Gauß's lemma, we know that factorizations of univariate integral polynomials are essentially the same in $\mathbb{Z}[x]$ and $\mathbb{Q}[x]$. For reasons of efficiency we concentrate on the case of integral polynomials. The factorization of integers is a much harder problem than the factorization of polynomials. For this reason we do not intend to factor the content (gcd of the coefficients) of integral polynomials. Throughout this section we assume that the polynomial to be factored is a primitive (content $= 1$) non-constant polynomial.

The problem of factoring a primitive univariate integral polynomial $a(x)$ consists in finding pairwise relatively prime irreducible polynomials $a_i(x)$ and positive integers m_i such that $a(x)=\prod_{i=1}^{r} a_i(x)^{m_i}$.

As for polynomials over finite fields we first compute a squarefree factorization of $a(x)$. By application of SQFR_FACTOR, p.45, our factorization problem is reduced to the problem of factoring a primitive squarefree polynomial.

So from now on let us assume that $a(x)$ is primitive and squarefree over $\mathbb{Z}$.

— — —

We would like to use the fast factorization algorithm modulo a prime p FACTOR_B, p.49, first. The problem of factorization over $\mathbb{Z}$ is reduced to factorization modulo p and a subsequent lifting of the result to a factorization modulo p^k. If k is high enough, the integer factors can be constructed, [Zas69]. The lifting process is based on a *constructive* form of Hensel's lemma (1918), which is a simple proposition in p-adic analysis.

Lemma 50 **(Hensel lemma)** *Let $a(x) \in \mathbb{Z}[x]$ be primitive and squarefree. Let p be a prime number not dividing $\mathrm{lc}(a)$. Let $a_1(x), \ldots, a_r(x) \in \mathbb{Z}_p[x]$ be pairwise relatively prime such that $a \equiv a_1 \cdot \ldots \cdot a_r \bmod p$ and $\mathrm{lc}(a_1) = \mathrm{lc}(a) \bmod p$, $\mathrm{lc}(a_2) = \ldots = \mathrm{lc}(a_r) = 1$.*
Then for every natural number k there are polynomials $a_1^{(k)}(x), \ldots, a_r^{(k)}(x) \in \mathbb{Z}_{p^k}[x]$ with $\mathrm{lc}(a_1^{(k)}) = \mathrm{lc}(a) \bmod p^k$, $\mathrm{lc}(a_2^{(k)}) = \ldots = \mathrm{lc}(a_r^{(k)}) = 1$ such that $a(x) \equiv a_1^{(k)}(x) \cdot \ldots \cdot a_r^{(k)}(x) \bmod p^k$ and $a_i^{(k)}(x) \equiv a_i(x) \bmod p$ for $1 \le i \le r$.

Proof. We proceed by induction on k. For $k = 1$ we can obviously choose $a_i^{(1)} = a_i$ and all the requirements are satisfied.

So now assume that the $a_i^{(k)}$ satisfy the requirements. That is, for some $\hat{d}(x) \in \mathbb{Z}_p[x]$ we have $a - \prod_{i=1}^r a_i^{(k)} \equiv p^k \hat{d} \bmod p^{k+1}$. We replace the leading coefficient of $a_1^{(k)}$ by ($\mathrm{lc}(a) \bmod p^{k+1}$). Then for some $d(x) \in \mathbb{Z}_p[x]$ we have $a - \prod_{i=1}^r a_i^{(k)} \equiv p^k d \bmod p^{k+1}$, where $\deg(d) < \deg(a)$.

We will determine $b_i(x) \in \mathbb{Z}_p[x]$ with $\deg(b_i) < \deg(a_i)$ such that $a_i^{(k+1)} = a_i^{(k)} + p^k b_i$. Using this ansatz[2], we get

$$a - \prod_{i=1}^r a_i^{(k+1)} \equiv \underbrace{a - \prod_{i=1}^r a_i^{(k)}}_{p^k d} - p^k \Big(\sum_{i=1}^r (b_i \underbrace{\prod_{j=1, j\neq i}^r a_j}_{=:\tilde{a}_i}) \Big) \bmod p^{k+1}.$$

So the $a_i^{(k+1)}$'s will constitute a factorization modulo p^{k+1} if and only if

$$d \equiv \sum_{i=1}^r b_i \cdot \tilde{a}_i \bmod p.$$

A solution is guaranteed by the following theorem[3]: "For $a_1, \ldots, a_r \in K[x]$ pairwise relatively prime and $c \in K[x]$, K a field, with $\deg(c) < \deg(a_1) + \ldots + \deg(a_r) = n$ there exist $u_1, \ldots, u_r \in K[x]$ with $\deg(u_i) < \deg(a_i)$ for $1 \le i \le r$, such that $c = \sum_{i=1}^r (u_i \prod_{j=1, j\neq i}^r a_j)$",
and algorithm LIN_COMB below. ■

The following algorithm LIN_COMB will be used as a subalgorithm in the lifting process.

Algorithm **LIN_COMB** (in: $[a_1, \ldots, a_r]$; out: $[b_1, \ldots, b_r]$);
$[a_i \in K[x]$ (K a field) pairwise relatively prime;

[2]ansatz means hair-line.

[3]This is a generalization of a property related to the extended Euclidean algorithm, see step 2 of LIN_COMB.

```
$b_i \in K[x]$, $\deg(b_i) < \deg(a_i)$ and $1 = \sum_{i=1}^{r} b_i\tilde{a}_i$, where $\tilde{a}_i = \prod_{j=1,j\neq i}^{r} a_j$]
  1.  $d := 1$; $i := 0$;
      for $j = 2$ to $r$ do $a_j^* := \prod_{k=j}^{r} a_k$;
  2.  while $i < r - 1$ do
      {$i := i + 1$;
      compute $u, v$ such that $d = ua_i + va_{i+1}^*$,
        $\deg(u) < \deg(a_{i+1}^*)$, $\deg(v) < \deg(a_i)$
        [by the extended Euclidean Algorithm, cf. [CAbks].]
      $b_i := v$; $d := u$};
  3.  $b_r := d$;
      return.
```

We summarize these algorithmic ideas in LIFT_FACTORS.

```
Algorithm LIFT_FACTORS (in: $a, [a_1, \dots, a_r], p, K$; out: $F$);
[$a$ is a primitive squarefree polynomial in $\mathbb{Z}[x]$, $p$ is a prime number not dividing
lc($a$) and s.t. ($a$ mod $p$) is squarefree in $\mathbb{Z}_p[x]$, $a_1, \dots, a_r \in \mathbb{Z}_p[x]$ pairwise relatively
prime, lc($a_1$) = lc($a$) mod $p$, lc($a_2$) = ... = lc($a_r$) = 1, and $a \equiv a_1 \cdot \ldots \cdot a_r$ mod $p$,
$K \in \mathbb{N}$;
$F = [\bar{a}_1, \dots, \bar{a}_r]$, $\bar{a}_i \in \mathbb{Z}_{p^K}[x]$, such that $a \equiv \bar{a}_1 \cdot \ldots \cdot \bar{a}_r$ mod $p^K$, lc($\bar{a}_1$) = lc($a$) mod
$p^K$, lc($\bar{a}_2$) = ... = lc($\bar{a}_r$) = 1, and $\bar{a}_i \equiv a_i$ mod $p$.]
  1.  by an application of LIB_COMB to $[a_1, \dots, a_r]$ compute $v_i \in \mathbb{Z}_p[x]$ s.t.
      $\deg(v_i) < \deg(a_i)$ and $1 \equiv \sum_{i=1}^{r} v_i\tilde{a}_i$ mod $p$, where $\tilde{a}_i = \prod_{j=1,j\neq i}^{r} a_j$;
  2.  for $i = 1$ to $r$ do $\bar{a}_i := a_i$;
      $k := 1$;
  3.  while $k < K$ do
      { replace lc($\bar{a}_1$) by ( lc($a$) mod $p^{k+1}$ );
        $\tilde{d} := ( (a - \prod_{i=1}^{r} \bar{a}_i)$ mod $p^{k+1}$ );
        $d := \tilde{d} / p^k$;
        for $i = 1$ to $r$ do
        {$b_i := \mathrm{rem}(dv_i, a_i)$;
          $\bar{a}_i := \bar{a}_i + p^k b_i$};
        $k := k + 1$ };
  4.  $F := [\bar{a}_1, \dots, \bar{a}_r]$;
      return.
```

As for the general lifting algorithm LIFT there is also a quadratic lifting scheme for LIFT_FACTORS. The interested reader is referred to [CAbks].

Now we put all the subalgorithms together and we get the **Berlekamp-Hensel**

algorithm FACTOR_BH for factoring primitive univariate squarefree polynomials over the integers.

```
Algorithm FACTOR_BH(in: $a$; out: $F$);
[$a$ is a primitive squarefree polynomial in $\mathbb{Z}[x]$; $F = [a_1, \ldots, a_r]$, where $a_1, \ldots, a_r$
are primitive irreducible polynomials in $\mathbb{Z}[x]$ such that $a = a_1 \cdot \ldots \cdot a_r$.]
 1. choose a prime number $p$ such that $p \nmid \mathrm{lc}(a)$ and $a$ is squarefree modulo $p$
    (i.e., $p$ does not divide the discriminant of $a$);
 2. $[u_1, \ldots, u_s] :=$ FACTOR_B($a, p$); (p.49)
    normalize the $u_i$'s such that $\mathrm{lc}(u_1) = \mathrm{lc}(a) \bmod p$ and $\mathrm{lc}(u_2) = \cdots =$
    $\mathrm{lc}(u_s) = 1$;
 3. determine a natural number $B$ which bounds the absolute value of any
    coefficient in a factor of $a$ over the integers ( for instance, use the
    Landau-Mignotte bound: Let $a(x) = \sum_{i=0}^{m} a_i x^i$ and $b(x) = \sum_{i=0}^{n} b_i x^i$
      be polynomials over $\mathbb{Z}$ ($a_m \neq 0 \neq b_n$) such that $b$ divides $a$.
      Then $\sum_{i=0}^{n} |b_i| \leq 2^n |\frac{b_n}{a_m}| \sqrt{\sum_{i=0}^{m} a_i^2}$. )
    $K := \min\{k \in \mathbb{N} \mid p^k \geq 2|\mathrm{lc}(a)|B\}$;
 4. $[v_1, \ldots, v_s] :=$ LIFT_FACTORS($a$, $[u_1, \ldots, u_s]$, $p$, $K$);
 5. [combine factors]
    $\bar{a} := a$;
    $C := \{2, \ldots, s\}$; [$v_1$ will be included in the last factor]
    $i := 0$;
    $m := 0$;
    while $m < |C|$ do
    {$m := m + 1$;
    for all $\{i_1, \ldots, i_m\} \subseteq C$ do
      { [integers modulo $p^K$ are centered around 0, i.e., the representation of
      $\mathbb{Z}_{p^K}$ is $\{q \mid -p^K/2 < q \leq p^K/2\}$]
      $\tilde{b} := (\mathrm{lc}(\bar{a}) \cdot v_{i_1} \cdot \ldots \cdot v_{i_m} \bmod p^K)$, interpreted as a polynomial over the
      integers;
      $b := \mathrm{pp}(\tilde{b})$;
      if $b \mid \bar{a}$ then $\{i := i + 1;\ a_i := b;\ \bar{a} := \bar{a}/b;\ C := C \setminus \{i_1, \ldots, i_m\}\}$ }
    };
    $i := i + 1$;
    $a_i := \bar{a}$;
 6. $F := [a_1, \ldots, a_i]$;
    return.
```

Step (5) is necessary, because irreducible factors over the integers might factor further modulo a prime p. In fact, there are irreducible polynomials over the integers which factor modulo every prime number. An example of this is x^4+1, cf. [CAbks].

The complexity of FACTOR_BH would be polynomial in the size of the input except for step (5). Since in step(5), in the worst case, we have to consider all possible combinations of factors modulo p, this might lead to a combinatorial explosion, rendering the algorithm FACTOR_BH exponential in the size of the input. Nevertheless, in practical examples the combinations of factors does not present an insurmountable problem. Basically all the major *computer algebra systems* employ some variant of FACTOR_BH as the standard factoring algorithm for polynomials over the integers.

Example. We want to factor the primitive squarefree integral polynomial

$$a(x) = 6x^7 + 7x^6 + 4x^5 + x^4 + 6x^3 + 7x^2 + 4x + 1.$$

We use FACTOR_BH in the process. A suitable prime is 5, $a(x)$ stays squarefree modulo 5.
By an application of the Berlekamp algorithm FACTOR_B, $a(x)$ is factored modulo 5 into

$$a(x) \equiv \underbrace{(x-2)}_{u_1} \cdot \underbrace{(x^2-2)}_{u_2} \cdot \underbrace{(x^2+2)}_{u_3} \cdot \underbrace{(x^2-x+2)}_{u_4} \bmod 5.$$

By an application of LIFT_FACTORS we lift this factorization to a factorization modulo 25, getting

$$a(x) \equiv \underbrace{(6x+3)}_{v_1} \cdot \underbrace{(x^2-7)}_{v_2} \cdot \underbrace{(x^2+7)}_{v_3} \cdot \underbrace{(x^2+9x-8)}_{v_4} \bmod 25.$$

The Landau-Mignotte bound for a is rather big. Let us assume that by some additional insight we know that $K = 2$ is good enough for constructing the integral factors. Now we have to try combinations of factors modulo 25 to get the factors over the integers. So we set $\bar{a} := a$ and $C := \{2,3,4\}$. Testing the factors v_2, v_3, v_4 we see that only v_4 yields a factor over the integers: $a_1(x) := \text{pp}(\text{lc}(\bar{a}) \cdot v_4 \bmod 25) = 3x^2 + 2x + 1$.
So now $\bar{a} := \bar{a}/a_1 = 2x^5 + x^4 + 2x + 1$. The combination of v_2 and v_3 yields the factor $a_2(x) := \text{pp}(\text{lc}(\bar{a}) \cdot v_2 \cdot v_3 \bmod 25) = x^4 + 1$.
We set $\bar{a} := \bar{a}/a_2 = 2x + 1$. Now C has become empty, and the last factor is $a_3(x) := \bar{a}(x) = 2x + 1$.
FACTOR_BH returns $F = [a_1, a_2, a_3]$, i.e., the factorization $a(x) = (3x^2 + 2x + 1) \cdot (x^4 + 1) \cdot (2x + 1)$. □

3.5 Factorization over algebraic extension fields

3.5.1 Reduction of the problem to the ground field

We describe an algorithm that has been presented in van der Waerden, [GaloisBks] and slightly improved by B. Trager, [Tra76].

Let K be a computable field of characteristic 0 such that there is an algorithm for factoring polynomials in $K[x]$. Let α be algebraic over K with minimal polynomial $p(y)$ of degree n. Throughout this section we call K the **ground field** and $K(\alpha)$ the **extension field**. Often we will write a polynomial $f(x) \in K(\alpha)[x]$ as $f(x,\alpha)$ to indicate the occurrence of α in the coefficients. Let $\alpha = \alpha_1, \alpha_2, \dots, \alpha_n$ be the roots of $p(y)$ in a splitting field of p over K. By $\phi_j, 1 \le j \le n$, we denote the canonical field isomorphism that takes α into α_j, i.e.,

$$\begin{aligned} \phi_j : K(\alpha) &\to K(\alpha_j) \\ \alpha &\mapsto \alpha_j \\ a &\mapsto a \qquad \text{for all } a \in K. \end{aligned}$$

ϕ_j can be extended to $\phi_j : K(\alpha)[x] \to K(\alpha_j)[x]$ by letting it act on the coefficients.

We will reduce the problem of factorization in $K(\alpha)[x]$ to factorization in $K[x]$. This reduction will be achieved by associating a $g \in K[x]$ with the given $f \in K(\alpha)[x]$ such that the factors of f are in a computable 1-1 correspondence with the factors of g, i.e.,

$$\begin{aligned} f(x) \in K(\alpha)[x] &\longleftrightarrow g \in K[x] \\ \text{factors of } f &\xleftrightarrow{1\text{-}1} \text{factors of } g. \end{aligned}$$

A candidate for such a function is the **norm**, which maps an element in the extension field to the product of all its conjugates over K. This product is an element of K.

$$\begin{aligned} \mathrm{norm}_{[K(\alpha)/K]} : \; & K(\alpha) \to K \\ & \beta \mapsto \prod_{\beta' \sim \beta} \beta' \end{aligned}$$

, where $\beta' \sim \beta$ means that β' is conjugate to β relative to $K(\alpha)$ over K. That is, if $\beta = q(\alpha)$ is the standard representation of β in $K(\alpha)$, then $\mathrm{norm}_{[K(\alpha)/K]}(\beta) = \prod_{i=1}^{n} q(\alpha_i)$. If the field extension is clear from the context, we write just $\mathrm{norm}(\cdot)$ instead of $\mathrm{norm}_{[K(\alpha)/K]}(\cdot)$. Since the norm is symmetric in the α_i's, by the fundamental theorem on symmetric polynomials, theorem 5 it can be expressed in terms of the coefficients of p and thus lies in K. The norm can be generalized from $K(\alpha)$ to $K(\alpha)[x]$ by defining the norm of a polynomial $h(x,\alpha)$ to be $\prod_{i=1}^{n} h(x,\alpha_i)$, which can be computed as $\mathrm{norm}(h(x,\alpha)) = \mathrm{res}_y(p(y), h(x,y))$, cf. definition 21. One important property of the norm is multiplicativity, i.e., $\mathrm{norm}(f \cdot g) = \mathrm{norm}(f) \cdot \mathrm{norm}(g)$.

Theorem 51 *If $f(x,\alpha)$ is irreducible over $K(\alpha)$, then* $\mathrm{norm}(f) = h(x)^j$ *for some irreducible $h \in K[x]$ and some $j \in \mathbb{N}$.*

Proof. Assume $\mathrm{norm}(f) = g(x)h(x)$ and g, h are relatively prime. For $1 \le i \le n$ let

$f_i(x) = f(x, \alpha_i)$. Clearly $f = f_1$ divides $\text{norm}(f) = \prod f_i$. So, since f is irreducible, $f|g$ or $f|h$. W.l.o.g. let us assume that $f|h$, i.e., $h(x) = f_1(x, \alpha) \cdot \tilde{h}(x, \alpha)$. Then $h(x) = \phi_j(h) = \phi_j(f_1)\phi_j(\tilde{h}) = f_j\tilde{h}(x, \alpha_j)$. Therefore, $f_j|h$ for $1 \leq j \leq n$. Since g and h are relatively prime, this implies that $\gcd(f_j, g) = 1$ for $1 \leq j \leq n$. Thus, $\gcd(\text{norm}(f), g) = 1$, i.e., $g = 1$. ■

Remark: The previous theorem yields a method for finding minimal polynomials for elements $\beta \in K(\alpha)$. Let $\beta = q(\alpha)$, $b(x) = \text{norm}(x - \beta) = \text{norm}(x - q(\alpha))$. $x - \beta \mid b(x)$, so $b(\beta) = 0$. Therefore the minimal polynomial $p_\beta(x)$ has to be one of the irreducible factors of $b(x)$. By the above theorem, $b(x) = p_\beta(x)^j$ for some $j \in \mathbb{N}$. So $p_\beta(x)$ can be determined by squarefree factorization of $b(x)$, Algorithm SQFR_FACTOR, p.45.

$K(\alpha)[x]$ is a Euclidean domain, so by successive application of the Euclidean algorithm the problem of factoring in $K(\alpha)[x]$ can be reduced to the problem of factoring squarefree polynomials in $K(\alpha)[x]$, Algorithm SQFR_FACTOR, p.45.

From now on let us assume that $f(x, \alpha) \in K(\alpha)[x]$ is squarefree.

Theorem 52 *Let $f(x, \alpha) \in K(\alpha)[x]$ be such that $F(x) = \text{norm}(f)$ is squarefree. Let $F(x) = \prod_{i=1}^{r} G_i(x)$ be the irreducible factorization of $F(x)$. Then $\prod_{i=1}^{r} g_i(x, \alpha)$, where $g_i(x, \alpha) = \gcd(f, G_i)$ over $K(\alpha)$, is the irreducible factorization of $f(x, \alpha)$ over $K(\alpha)$.*

Proof. The statement follows from

a. every g_i divides f,

b. every irreducible factor of f divides one of the g_i's.

c. the g_i's are relatively prime, and

d. every g_i is irreducible.

Ad (a): This is obvious from $g_i = \gcd(f, G_i)$.

Ad (b): Let $v(x, \alpha)$ be an irreducible factor of f over $K(\alpha)$. By Theorem 51, $\text{norm}(v) = w(x)^k$ for some irreducible $w(x) \in K[x]$. $v|f$ implies $\text{norm}(v)|\,\text{norm}(f)$. Since $\text{norm}(f)$ is squarefree, $\text{norm}(v)$ is irreducible and must be one of the G_i's. So $v|g_i(x, \alpha)$.

Ad (c): Suppose the irreducible factor v of f divides both g_i and g_j for $i \neq j$. Then the irreducible polynomial $\text{norm}(v)$ divides both $\text{norm}(G_i) = G_i^n$ and $\text{norm}(G_j) = G_j^n$. This would mean that G_i and G_j have a common factor.

Ad (d): Clearly every g_i is squarefree. Assume that $v_1(x, \alpha)$ and $v_2(x, \alpha)$ are distinct irreducible factors of f and that both of them divide $g_i = \gcd(f, G_i)$. $v_1|G_i$ implies $\text{norm}(v_1)|\text{norm}(G_i)=G_i(x)^n$. Because of the squarefreeness of $\text{norm}(f)$, we must have $\text{norm}(v_1) = G_i$. Similarly we get $\text{norm}(v_2) = G_i$. But $(v_1 \cdot v_2)|f$ implies

$\mathrm{norm}(v_1 \cdot v_2) = G_i(x)^2 | \mathrm{norm}(f)$, in contraction to the squarefreeness of $\mathrm{norm}(f)$. ■

So we can solve our factorization problem over $K(\alpha)$, if we can show that we can restrict our problem to the situation in which $\mathrm{norm}(f)$ is squarefree. The following lemmata and theorem will guarantee exactly that.

Lemma 53 *If $f(x)$ is a squarefree polynomial in $K[x]$, then there are only finitely many $s \in K$ for which $\mathrm{norm}(f(x - s\alpha))$ is not squarefree.*

Proof. Let $\beta_1, \ldots, \beta_m$ be the distinct roots of f. Then the roots of $f(x - s\alpha_j)$ are $\beta_i + s\alpha_j$, $1 \le i \le m$. Thus, the roots of $G(x) = \mathrm{norm}(f(x - s\alpha_j)) = \prod_{k=1}^{n} f(x - s\alpha_k)$ are $\beta_i + s\alpha_k$ for $1 \le i \le m, 1 \le k \le n$. G can have a multiple root only if $s = \frac{\beta_j - \beta_i}{\alpha_k - \alpha_l}$, where $k \ne l$. There are only finitely many such values. ■

Lemma 54 *If $f(x, \alpha)$ is a squarefree polynomial in $K(\alpha)[x]$, then there exists a squarefree polynomial $g(x) \in K[x]$ such that $f \mid g$.*

Proof. Let $G(x) = \mathrm{norm}(f(x, \alpha)) = \prod g_i(x)^i$ be the squarefree factorization of the norm of f. Since f is squarefree, $f | g := \prod g_i(x)$. ■

Theorem 55 *For any squarefree polynomial $f(x, \alpha) \in K(\alpha)[x]$ there are only finitely many $s \in K$ for which $\mathrm{norm}(f(x - s\alpha))$ is not squarefree.*

Proof. Let $g(x)$ be as in the above lemma. By the previous Lemma, there are only finitely many $s \in K$ for which $\mathrm{norm}(g(x - s\alpha))$ is not squarefree. But $f | g$ implies $\mathrm{norm}(f(x - s\alpha)) | \mathrm{norm}(g(x - s\alpha))$. If $\mathrm{norm}(f(x - s\alpha))$ is not squarefree, then neither is $\mathrm{norm}(g(x - s\alpha))$. ■

```
Algorithm SQFR_NORM(in: $f$; out: $g, s, N$);
[$f \in K(\alpha)[x]$ squarefree; $s \in \mathbb{N}$, $g(x) = f(x - s\alpha)$,
$N(x) = \mathrm{norm}(g(x, \alpha))$ is squarefree.]
  1.  $s := 0$; $g(x, \alpha) := f(x, \alpha)$;
  2.  $N(x) := \mathrm{res}_y(p(y), g(x, y))$;
  3.  while $\deg(\gcd(N(x), N'(x))) \ne 0$ do
      {$s := s + 1$;
       $g(x, \alpha) := g(x - \alpha, \alpha)$;
       $N(x) := \mathrm{res}_y(p(y), g(x, y))$};
      return.
```

So over a field of characteristic 0 we can always find a transformation of the form $f(x - s\alpha)$, $s \in \mathbb{N}$, such that $\mathrm{norm}(f(x - s\alpha))$ is squarefree. These considerations give rise to an algorithm for computing a linear change of variable which transforms f to a polynomial with squarefree norm.

Now we are ready to present an algorithm for factoring polynomials over the extension field.

```
Algorithm FACTOR_ALG(in: $f$; out: $F$);
[$f \in K(\alpha)[x]$ squarefree; $F = [f_1, \dots, f_r]$, where $f_1, \dots, f_r$ are the irreducible factors
of $f$ over $K(\alpha)$.]
 1. $[g, s, N] :=$ SQFR_NORM($f$);
 2. $L :=$ list of irreducible factors of $N(x)$ over $K$;
 3. if LENGTH($L$) $= 1$ then return($[f]$);
 4. $F := []$;
    for each $H(x)$ in $L$ do
    {$h(x,\alpha) := \gcd(H(x), g(x,\alpha))$;
     $g(x,\alpha) := g(x,\alpha)/h(x,\alpha)$;
     $F :=$ CONS($h(x + s\alpha, \alpha), F$)};
    return.
```

Example. We apply the factorization algorithm FACTOR_ALG to the domain $\mathbb{Q}(\sqrt[3]{2})[x]$, i.e., $K = \mathbb{Q}$, α a root of $p(y) = y^3 - 2$. Let us factor the polynomial $f(x,\alpha) = x^4 + \alpha x^3 - 2x - 2\alpha$.

$f(x,\alpha)$ is squarefree. First we have to transform f to a polynomial g with squarefree norm. The norm of f itself is $\mathrm{norm}(f) = \mathrm{res}_y(p(y), f(x,y)) = (x^3-2)^3(x^3+2)$, i.e., it is not squarefree. The transformation $x \mapsto x - \alpha$ does not work, but $x \mapsto x - 2\alpha$ does: $g(x,\alpha) := f(x - 2\alpha, \alpha) = x^4 - 7\alpha x^3 + 18\alpha^2 x^2 - 42x + 18\alpha$, $N(x) = \mathrm{norm}(g) = x^{12} - 56x^9 + 216x^6 - 6048x^3 + 11664$, and $N(x)$ is squarefree.

The factorization of $N(x)$ is $N(x) = (x^3 - 2)(x^3 - 54)(x^6 + 108)$.

Computing the gcd of all the factors of $N(x)$ with $g(x,\alpha)$ gives us the factorization of $g(x,\alpha)$: $g(x,\alpha) = (x - \alpha)(x - 3\alpha)(x^2 - 3\alpha x + 3\alpha^2)$, which can be transformed by $x \mapsto x + 2\alpha$ to the factorization $f(x,\alpha) = (x + \alpha)(x - \alpha)(x^2 + \alpha x + \alpha^2)$. □

3.5.2 Computation of primitive elements for multiple field extensions

Over a field K of characteristic 0 every algebraic extension field $K(\alpha)$ is separable, i.e., α is a root of multiplicity 1 of its minimal polynomial. So every multiple algebraic extension $K \subset K(\alpha_1) \subset \ldots \subset K(\alpha_1, \ldots, \alpha_n)$ can be expressed as a simple algebraic extension, i.e., $K(\alpha_1, \ldots, \alpha_n) = K(\gamma)$, theroem 10 for some γ algebraic over K. Such a γ is a **primitive element** for the field extension. We will describe how to compute primitive elements.

Clearly it suffices to find primitive elements for double field extensions $K \subset K(\alpha) \subset K(\alpha,\beta)$, where $p(\alpha) = 0$ for some irreducible $p(x) \in K[x]$ and $q(\beta,\alpha) = 0$ for some irreducible $q(x,\alpha) \in K(\alpha)[x]$. Let $n = \deg(p)$ and $m = \deg(q)$.

Theorem 56 *If $N(x) = \mathrm{norm}_{[K(\alpha)/K]}(q(x,\alpha))$ is squarefree, then $K(\alpha,\beta) = K(\beta)$, and $N(x)$ is the minimal polynomial for β over K.*

Proof. Let $\alpha_1, \dots \alpha_n$ be the roots of $p(x)$, and $\beta_{i_1}, \dots, \beta_{i_m}$ the roots of $q(x,\alpha_i)$. $\mathrm{norm}_{[K(\alpha)/K]}(q) = \prod_{i=1}^{n} q(x,\alpha_i)$, so if this norm is squarefree, then all the β_{i_j} must be different. So for every β in $\{\beta_{i_j} \mid 1 \le i \le n, 1 \le j \le m\}$there is a uniquely determined α in $\{\alpha_1, \dots \alpha_n\}$ such that $q(\beta,\alpha) = 0$. Thus, $\gcd(q(\beta,x),p(x))$ must be linear, $\gcd(q(\beta,x),p(x)) = x - r(\beta)$ for some $r(y) \in K[y]$, and therefore $\alpha = r(\beta)$. So $K(\alpha,\beta) = K(\beta)$.

β is a root of $N(x) = \mathrm{norm}_{[K(\alpha)/K]}(q)$. By theorem 51, and the squarefreeness of $N(x)$, $N(x)$ must be the minimal polynomial for β over K. ■

```
Algorithm PRIMITIVE_ELEMENT(in: $p, q$; out: $N, A, B$);
[$p$ and $q$ are the minimal polynomials for $\alpha$ and $\beta$, respectively, as above; $N(x)$ is
the minimal polynomial of $\gamma$ over $K$ such that $K(\alpha,\beta) = K(\gamma)$, $A$ and $B$ are the
standard representations of $\alpha$ and $\beta$ in $K(\gamma)$ respectively.]
  1. $[g, s, N] :=$ SQFR_NORM($q(x,\alpha)$);
  2. $A :=$ solution of the linear equation $\gcd(g(\gamma,x),p(x)) = 0$ in $K(\gamma)$,
     where $N(\gamma) = 0$;
  3. $B := \gamma - sA$;
     return.
```

Example. Let us compute a primitive element for the multiple extension $\mathbf{Q}(\sqrt{2},\sqrt{3})$, i.e., for $\mathbf{Q}(\alpha,\beta)$, where α is a root of $p(x) = x^2 - 2$ and β is a root of $q(x,\alpha) = q(x) = x^2 - 3$.

The norm of q is not squarefree, in fact $\mathrm{norm}_{[\mathbf{Q}(\sqrt{2})/\mathbf{Q}]}(q) = (x^2-3)^2$. So we need a linear transformation of the form $x \mapsto x - s\alpha$, and in fact $s = 1$ works.
$g(x,\alpha) := q(x-\alpha,\alpha) = x^2 - 2\alpha x - 1$,
$N(x) = \mathrm{norm}_{[\mathbf{Q}(\sqrt{2})/\mathbf{Q}]}(g(x,\alpha)) = (x^2 - 2\alpha x - 1)(x^2 + 2\alpha x - 1) = x^4 - 10x^2 + 1$.

$N(x)$ is irreducible. Let γ be a root of $N(x)$. So $\gamma = \beta + \alpha$. We get the representation of α in $K(\gamma)$ as the solution of the linear equation $\gcd(g(\gamma,x),p(x)) = \gcd(-2\gamma x + (\gamma^2 - 1), x^2 - 2) = x + \frac{1}{2}(-\gamma^3 + 9\gamma) = 0$,
i.e., $\alpha = A(\gamma) = \frac{1}{2}(\gamma^3 - 9\gamma)$. Finally $\beta = B(\gamma) = \gamma - A(\gamma) = -\frac{1}{2}(\gamma^3 - 11\gamma)$. □

Chapter 4

Soicher-McKay's Method

The principal references for this chapter are

- "L. Soicher and J. McKay, *Computing Galois groups over the rationals*, Journal of Number Theory 20 (1985), 273-281," [SM85] and
- "L. Soicher, *An algorithm for computing Galois groups*, in `Computational Group Theory` (M. D. Atkinson, Ed.), pp. 291-296, Academic Press, 1984," [Soi84].

4.1 Overview, Restrictions and Background

Let $f = f(x)$ be a polynomial in $\mathbf{Q}[x]$, (thus is a separable polynomial over $\mathbf{Q}$), and let $\{\alpha_1, ..., \alpha_n\}$ be the roots of f, where α_i distinct. We regard $Gal_{\mathbf{Q}}(f)$, the Galois group over the rationals, to be the group of permutations of the (indices of the) zeros of f induced by the group of automorphisms of the splitting field, $\mathrm{spl}_{\mathbf{Q}}(f)$, of f, cf. Chapter,section2.2.

We describe feasible computational techniques to determine $Gal_{\mathbf{Q}}(f)$. By efficiently determine sufficient properties (invariants), the aim is to specify $Gal_{\mathbf{Q}}(f)$ to *within conjugacy* in the symmetric group S_n of degree n. This conjugation is realised by relabelling the zeros of f.

The main tool discussed is the (absolute) resolvent polynomial. For Υ in $\mathbb{Z}[x_1, \ldots, x_n]$, the complete factorization of a resolvent polynomial is used to determine the orbit length partition of $\{\Upsilon(x_{1^\sigma}, \ldots, x_{n^\sigma}) : \sigma \text{ in } S_n\}$ under the action of $Gal_{\mathbf{Q}}(f)$.

An important class of resolvent polynomials considered are the linear resolvent polynomials, where $\Upsilon = e_1x_1 + \cdots e_rx_r$, e_i in $\mathbb{Z}$ and $0 < r \leq n$. The use of linear resolvents in determining $Gal_{\mathbb{Q}}(f)$ is discussed. A practical exact method of computing linear resolvents is described.

Each transitive permutation group of degree 3 to 7 is realised as a Galois group over the rationals. The exact computation furnish a proof of the result.

Restrictions We consider only irreducible f so that $Gal_{\mathbb{Q}}(f)$ is transitive, theorem 19, and assume without loss of generality that f is monic with integer coefficients.

Background We prove a result of Galois theory here. One may like to compare Kaplansky, [GaloisBks][2], p.20.

Let $\mathcal{G} = Gal(E/F)$ be the Galois group of a Galois extension E/F of fields.

Lemma 57 *Let $S = \{\beta_1, \ldots, \beta_k\}$ be a finite subset of distinct elements in E and $g(x) = \prod_{i=1}^{k}(x - \beta_i)$. Then $\mathcal{G}$ maps S onto S (i.e. $\{\beta^\phi : \beta$ in S and ϕ in $\mathcal{G}\} = S$) if and only if $g(x)$ is in $F[x]$.*

Proof. Let $g(x) = \sum_{i=0}^{k} a_i x^i$, β in S, and ϕ in $\mathcal{G} = Gal(E/F)$. Suppose $g(x)$ is in $F[x]$. As ϕ is an automorphism of E fixing F pointwisely we have: $0 = g(\beta) = g(\beta)^\phi = (\sum_{i=0}^{k} a_i\beta^i)^\phi = \sum_{i=0}^{k} a_i(\beta^\phi)^i = g(\beta^\phi)$. Thus β^ϕ is in S for all β in S and ϕ in $\mathcal{G}$. Recall that each ϕ is bijective on E, every ϕ in $\mathcal{G}$ maps S onto S.

Conversely, suppose $\mathcal{G}$ maps S onto S. Then each ϕ in $\mathcal{G}$ induces a permutation of S. Thus $a_i^\phi = a_i$ for each coefficient a_i of $g(x)$, because each a_i is a symmetric function of $\beta_1, \ldots, \beta_k$. From Galois theory, this implies that a_i is in F. ■

Theorem 58 *Let β be in S where $S = \{\beta_1, \ldots, \beta_k\}$ is a finite subset of distinct elements in E. Denote by $\beta^{\mathcal{G}}$ the set $\{\beta^\phi : \phi$ in $\mathcal{G}\}$. Then $S = \beta^{\mathcal{G}}$ if and only if $g(x) = \prod_{i=1}^{k}(x - \beta_i)$ is an irreducible polynomial over F.*

Proof. If $S = \beta^{\mathcal{G}}$, then by the previous lemma, $g(x)$ is in $F[x]$. Suppose $g(x)$ is reducible. Then $g(x)$ has a factor $h(x)$ in $F[x]$ where $h(x) = \prod_{i \text{ in } \mathrm{I}}(x - \beta_i)$, for some I properly contained in $\{1, \ldots, k\}$. Then by the previous lemma $\mathcal{G}$ maps $\{\beta_i : i$ in I$\}$ onto itself, which contradicts the fact that $S = \beta^{\mathcal{G}}$.

Conversely, suppose that $g(x)$ is an irreducible polynomial in $F[x]$. By the previous lemma, we know that $\mathcal{G}$ maps S onto itself. Thus $\beta^{\mathcal{G}}$ is contained in S. Suppose $\beta^{\mathcal{G}} = \{\beta_i : i$ in I$\}$, where I is properly contained in $\{1, \ldots, k\}$. Then by the previous lemma, $h(x) = \prod_{i \text{ in } \mathrm{I}}(x - \beta_i)$ is in $F[x]$. Since $h(x)$ is a proper divisor of $g(x)$, we have arrived at the desired contradiction. ■

4.2 Determining cycle types in $Gal_{\mathbf{Q}}(f)$

A well-known method of determining cycle types in $Gal_{\mathbf{Q}}(f)$ is the following, in van der Waerden [GaloisBks].

First we prove:

Theorem 59 *Let I be an integral domain with identity, and let the Unique Factorization Theorem be valid for it. Let P be a prime ideal in I, and let $\overline{I} = I\ /P$ be the quotient ring. Let the fields of fractions of I and $\overline{I}$ be F and $\overline{F}$. Let $f(x) = x^n + \cdots$ be a polynomial in $I[x]$, and let $\overline{f}(x)$ be the polynomial assciated with it in the homomorphism $I \rightarrow \overline{I}$, assuming that neither has a double root.*
Then the Galois group $\overline{\Gamma}$ of $\overline{f}$ relative to $\overline{F}$ (as a permutation group of the suitably arranged roots) is a subgroup of the Galois group Γ of f.

Proof. By Gauß's lemma, the factorization of $\mathfrak{R}(z,\underline{u}) = \prod_s (z - \theta^{s\underline{u}})$ (see theorem 1) into factors $\mathfrak{R}_1\mathfrak{R}_2\cdots\mathfrak{R}_k$ that are irreducible in $F[z,\underline{u}]$ can actually be carried out in $I[z,\underline{u}]$. The natural homomorphism carries this factorization down into $\overline{I}[z,\underline{u}]$: $\overline{\mathfrak{R}}(z,\underline{u}) = \overline{\mathfrak{R}}_1\overline{\mathfrak{R}}_2\cdots\overline{\mathfrak{R}}_k$.

The polynomials $\overline{\mathfrak{R}}_1, \ldots$ may be reducible. By theorem 1, the permutations in Γ carry $\mathfrak{R}_1$ (and so $\overline{\mathfrak{R}}_1$) into itself; the other permutations of the $\underline{u}$'s carry $\overline{\mathfrak{R}}_1$ into $\overline{\mathfrak{R}}_2, \ldots, \overline{\mathfrak{R}}_k$.
By theorem 1 again, the permutations in $\overline{\Gamma}$ carry an irreducible factor of $\overline{\mathfrak{R}}_1$ into itself so that they cannot carry $\overline{\mathfrak{R}}_1$ into $\overline{\mathfrak{R}}_2, \ldots; \overline{\mathfrak{R}}_k$, but must carry $\overline{\mathfrak{R}}_1$ into $\overline{\mathfrak{R}}_1$, which means that $\overline{\Gamma}$ is a subgroup of Γ. ■

The theorem is frequently used for determining the group $Gal_{\mathbf{Q}}(f)$. In particular, we often choose the ideal (p) in such a manner that the polynomial $f(x)$ factors mod p, since in this way the Galois group of f can be determined more easily.

Theorem 60 *Let $\mathbf{Z}$ be the ring of integers and $f(x) \in \mathbf{Z}[x]$ be a monic polynomial. For a prime p not dividing* disc(f), *the discriminant of f, the partition of n induced by the degrees of the irreducible factors of f modulo p (called the* **degree partition** *of f* mod *p) is the cycle type of a permutation in $Gal_{\mathbf{Q}}(f)$.*

Proof. Let $P = (p)$ be the ideal generated by p, p being a prime number. Let $f(x)$ factor modulo p thus: $f(x) = f_1(x)f_2(x)\cdots f_h(x)$ mod p. It follows that $\overline{f} = \overline{f}_1\overline{f}_2\cdots\overline{f}_h$.

The Galois group $\overline{\Gamma}$ of $\overline{f}(x)$ is always cyclic, since the automorphism group of a Galois field is always cyclic, theorem 26. Let the generating permutation s of $\overline{\Gamma}$, written as a product of cycles, be $(12\ldots j)(j+1\ldots)\ldots$. Since the transitivity sets of the group $\overline{\Gamma}$ correspond exactly to the irreducible factors of $\overline{f}$, theorem 58, the numbers occurring in the cycles $(12\ldots j)$, $(\ldots)$, $\ldots$ must exactly denote the roots of $\overline{f}_1$ and $\overline{f}_2, \cdots$.

Thus, as soon as the degrees $j, k, \ldots$ of $f_1, f_2, \ldots$ are known, the type of the substitution s is known as well: s consists of a cycle of j terms, of a cycle of k terms, and so on. Since, with a suitable arrangement of the roots, $\overline{\Gamma}$ is a subgroup of Γ by the above theorem, $\Gamma = Gal_{\mathbf{Q}}(f)$ must contain a permutation of the same type. ■

Thus, for example, if a quintic with integral coefficients resolves modulo any prime number into an irreducible factor of the second and into one of the third degree, the Galois group contains a permutation of the type (1 2) (3 4 5).

Example. Consider the equation $x^5 - x - 1 = 0$.
The left member factors modulo 2 into $(x^2+x+1)(x^3+x+2)$. It is irreducible modulo 3; for it had a linear or quadratic factor, it would have a factor in common with $x^9 - x$, theorem 27, and would therefore have to have a factor in common with $x(x^9 - x)$, i.e. with either $x^5 - x$ or $x^5 + x$, which evidently is not the case.
Hence the group contains a cycle of five symbols and a product $(ik)(lmn)$. The third power of the latter permutation is (ik); this, transformed by (12345) and its powers, gives a chain of transpositions $(ik), (kp), (pq), (qr), (ri)$ which together generate the symmetric group. Thus the group $Gal_{\mathbf{Q}}(f)$ is the symmetric group. □

In fact, cf. [LO77], we have

Theorem 61 *(**Čebotarev Density Theorem**) As $k \to \infty$, the proportion of occurrences of a degree partition T of f mod p_i, $i = 1, \ldots, k$, (p_i distinct primes) tends to the proportion of permutations in $Gal_{\mathbf{Q}}(f)$ having cycle type T.*

But full power of this result seems difficult to use in practice.

Butler and McKay [BM83] have tabulated the transitive permutation groups of degree up to 11, and the cycle type distribution of permutations in these groups. After f is factorized modulo various primes, these tables are used to obtain a set of groups $\{H_i\}$ such that for all i, $H_i \not\supseteq Gal_{\mathbf{Q}}(f)$.

In fact, if $Gal_{\mathbf{Q}}(f)$ is A_n or S_n, then $Gal_{\mathbf{Q}}(f)$ can usually be quickly determined using modulo p factorizations and the fact that $Gal_{\mathbf{Q}}(f)$ is a group of even permutations if and only if disc(f) is a rational integral square, theorem 42.

If $Gal_{\mathbf{Q}}(f)$ is neither A_n nor S_n, an historical and very useful method to determine $Gal_{\mathbf{Q}}(f)$ is the construction and factorization of appropriate resolvent polynomials.

4.3 Absolute Resolvents

Let $\Upsilon = \Upsilon(x_1, ..., x_n)$ be a polynomial in $\mathbb{Z}[x_1, ..., x_n]$ and let σ be a permutation in S_n. We define $\Upsilon^\sigma = \Upsilon(x_{1^\sigma}, ..., x_{n^\sigma})$.

Definition 62 *Let* $\{\Upsilon_1, ..., \Upsilon_k\} = \Upsilon^{S_n}$, *where the* Υ_i *are distinct functions. The* **absolute resolvent** *polynomial* $\Re(\Upsilon, f)$ *associated with* Υ *and* f *is defined by*

$$\Re(\Upsilon, f) = \prod_{i=1}^{k}(x - \Upsilon_i(\alpha_1, \ldots, \alpha_n)),$$

where $\alpha_1, \ldots, \alpha_n$ *(*α_i *distinct) are the roots of* f.

We may take $\Upsilon_i = \Upsilon^{\sigma_i}$, $1 \leqslant i \leqslant k$, where $\{\sigma_1, \ldots \sigma_k\}$ is a set of right coset representatives of $\mathrm{Stab}_{S_n}(\Upsilon)$ (the stabilizer in S_n of Υ) in S_n.

The coefficients of a resolvent polynomial $\Re(\Upsilon, f)$ are algebraic integers which are symmetric functions of the zeros $\alpha_1, \ldots, \alpha_n$ of f, hence these coefficients are rational integers by the fundamental theorem of symmetric polynomials, theorem 5.

This resolvent were introduced by Lagrange around 1770 in order to compute the relations between the roots of a polynomial and to study (in his own language) the field-extensions associated to these roots. It enabled him to unify in a way the former methods (Cardan, Ferrari) used to solve algebraic equations up to degree 4, and to understand why should not exist such methods beyond degree 4, cf. **Old-fashioned Galois theory** books by Edwards or Tignol [GaloisBks]. Notice that the Galois theory (now **classical**) as presented by Artin and Kaplansky do not contain this stuff, but **Modern** Galois theory (1970, 1985, 1990+ + +) is based on similar stuffs, and pull the computer into the game of *Symbolic and Algebraic Computation*, cf. Appendix B.1.

4.3.1 Construction of resolvent

The resolvent polynomial $\Re(\Upsilon, f)$ can be constructed by expanding $\Re(\Upsilon, f)$ symbolically in the zeros of f and then determining the coefficients of $\Re(\Upsilon, f)$ as polynomials in the coefficients of f. Unfortunately, unless $\deg(\Re(\Upsilon, f))$ is small or f is sparse, this leads to very extensive *symbolic* manipulation. However, if we use this method, we get an explicit formula for the coefficients of $\Re(\Upsilon, f)$ in terms of the coefficients of f. Such formulae have been published for certain resolvent polynomials, cf. the citations in [SM85].

$\Re(\Upsilon, f)$ can also be formed using high-precision numerical approximations to the zeros of f. If the coefficients of $\Re(\Upsilon, f)$ are determined to within an absolute error less than $\frac{1}{2}$, then these coefficients are determined exactly by rounding. Stauduhar, cf. chapter 2 in this thesis employs this method.

We next define a special type of absolute resolvent polynomials which can be computed symbolically and at the same time very useful to determine the conjugacy class of $Gal_{\mathbf{Q}}(f)$ in S_n.

Definition 63 *A resolvent polynomial $\Re(\Upsilon, f)$, where $\Upsilon = e_1x_1 + \cdots + e_rx_r$, for some r, $1 \le r \le n$, and $e_1, \ldots, e_r$ nonzero integers, is called a* **linear** *(absolute)* **resolvent** *polynomial.*

Soicher in [Soi84] details a new, *practical*, exact algorithm LRINT, cf. p. 71 below, to construct linear resolvent polynomials. This algorithm does not expand the resolvent symbolically in the zeros of f.

We shall now assume throughout that the zeros of $\Re(\Upsilon, f)$ are distinct (cf. theorem 65 below), for if not, we may apply an appropriate Tschirnhaus transformation (theorem 23) to f preserving the Galois group, then recompute $\Re(\Upsilon, f)$.

4.3.2 Complete Factorization of Resolvent

First we observe that any subgroup of S_n acts on $\Upsilon^{S_n} = \{\Upsilon^\sigma : \sigma \in S_n\}$.

Definition 64 *For a group G acting on a finite set S we call the partition of $|S|$ induced by the lengths of the orbits of S under G the* **orbit-length partition** *of S under G.*

We have the following theorem

Theorem 65 *Suppose $\Re(\Upsilon, f)$ has distinct roots. Then the orbit-length partition of Υ^{S_n} under $Gal_{\mathbf{Q}}(f)$ is the same as the partition of* $\deg(\Re(\Upsilon, f))$ *induced by the degrees of the irreducible factors of $\Re(\Upsilon, f)$.*

We would like to prove this theorem by the following lemma and proposition.

Let $E = \mathrm{spl}_{\mathbf{Q}}(f)$, $\phi \in \mathcal{G} = Gal(E/\mathbf{Q})$ and $\sigma_\phi \in \Gamma = Gal_{\mathbf{Q}}(f)$ as a subgroup in S_n. Recall the isomorphism $\begin{array}{ccc} \mathcal{G} & \rightarrow & \Gamma \\ \phi & \mapsto & \sigma_\phi \end{array}$ as in Chapter,section 2.2: ϕ induces a permutation on $\alpha_1, \alpha_2, \ldots, \alpha_n$, which can be set forth as follows: $\begin{pmatrix} \alpha_1, & \cdots & ,\alpha_n \\ \alpha_1^\phi, & \cdots & ,\alpha_n^\phi \end{pmatrix}$ or $\begin{pmatrix} \alpha_1, & \cdots & ,\alpha_n \\ \alpha_{i_1}, & \cdots & ,\alpha_{i_n} \end{pmatrix}$ or $\begin{pmatrix} 1, & \cdots & ,n \\ i_1, & \cdots & ,i_n \end{pmatrix}$. Letting σ_ϕ denote the final expression here.

Lemma 66 *We have* $\Upsilon(\alpha_1,\ldots,\alpha_n)^{\phi} = \Upsilon(x_1,\ldots,x_n)^{\sigma_\phi}|_{x_1=\alpha_1,\ldots,x_n=\alpha_n}$.

Proof. L.H.S.$= \Upsilon(\alpha_1,\ldots,\alpha_n)^{\phi} = \Upsilon(\alpha_1^{\phi},\ldots,\alpha_n^{\phi}) = \Upsilon(\alpha_{i_1},\ldots,\alpha_{i_n})$

R.H.S.$= \Upsilon(x_1,\ldots,x_n)^{\sigma_\phi}|_{x_1=\alpha_1,\ldots,x_n=\alpha_n} = \Upsilon(x_{1^{\sigma_\phi}},\ldots,x_{n^{\sigma_\phi}})|_{x_1=\alpha_1,\ldots,x_n=\alpha_n}$
$= \Upsilon(x_{i_1},\ldots,x_{i_n})|_{x_1=\alpha_1,\ldots,x_n=\alpha_n} = \Upsilon(\alpha_{i_1},\ldots,\alpha_{i_n})$. ■

Proposition 67 *Let* $t \in I \subset \{1,\ldots,k\}$ *and* $\{\Upsilon_1,...,\Upsilon_k\} = \Upsilon^{S_n}$, *where the* Υ_i *are distinct functions. Let* $\Gamma = Gal_{\mathbf{Q}}(f)$.
(1) If $\Upsilon_t^{\Gamma} = \{\Upsilon_i : i \text{ in } I\}$ *and the* $\Upsilon_i(\alpha_1,\ldots,\alpha_n)$ *are distinct for* i *in* I, *then* $g(x) = \prod_{i \text{ in } I}(x - \Upsilon_i(\alpha_1,\ldots,\alpha_n))$ *is an irreducible polynomial over* $\mathbf{Q}$.
(2) If $g(x) = \prod_{i \text{ in } I}(x - \Upsilon_i(\alpha_1,\ldots,\alpha_n))$ *is a non-repeated irreducible factor of* $\Re(\Upsilon, f)$ *then* $\Upsilon_t^{\Gamma} = \{\Upsilon_i : i \text{ in } I\}$.

Proof. (1) Apply the above lemma and theorem 58.
(2) As $E = \mathrm{spl}_{\mathbf{Q}}(f)$ is separable over $\mathbf{Q}$, $g(x)$ must have distinct zeros. By theorem 58 and the above lemma, $\{\Upsilon_i(\alpha_1,\ldots,\alpha_n) : i \text{ in I}\} = \{\Upsilon_t(x_1,\ldots,x_n)^{\sigma}|_{x_1=\alpha_1,\ldots,x_n=\alpha_n} : \sigma \text{ in } \Gamma\}$. As $g(x)$ is a non-repeated factor of $\Re(\Upsilon, f)$, for all i in I and $j = 1,...,k$, $\Upsilon_i(\alpha_1,\ldots,\alpha_n) = \Upsilon_j(\alpha_1,\ldots,\alpha_n)$ if and only if $i = j$. The result follows. ■

Proof of theorem 65: $\mathcal{G} = Gal(E/\mathbf{Q})$ as a group of automorphism on E, acts on the set of zeros of $\Re(\Upsilon, f)$, by fixing $\mathbf{Q}$ pointwisely and permuting the $\{\alpha_i\}$. As the zeros of $\Re(\Upsilon, f)$ are distinct this action is equivalent to the action by $\Gamma = Gal_{\mathbf{Q}}(f)$ on Υ^{S_n} from the lemma.
The orbits of the action by $\mathcal{G}$ on the zeros of $\Re(\Upsilon, f)$ are precisely the sets of zeros of the distinct irreducible factors (over $\mathbf{Q}$) of $\Re(\Upsilon, f)$, theorem 58. Once again as the zeros of $\Re(\Upsilon, f)$ are distinct, the theorem follows from the proposition. ■

To factorize $\Re(\Upsilon, f)$, we use Berlekamp-Hensel algorithm FACTOR_BH, p. 53. Alternatively, one can often determine candidates for factors of $\Re(\Upsilon, f)$ by using numerical approximations to the zeros of $\Re(\Upsilon, f)$.

Often (cf. Chapter 2 in this thesis and the citations in [SM85]) resolvent polynomials are used to determine if $Gal_{\mathbf{Q}}(f)$ is contained in some given proper subgroup G of S_n. If Υ is chosen so that $G = \mathrm{Stab}_{S_n}(\Upsilon)$, then $\Re(\Upsilon, f)$ has a linear factor if and only if $Gal_{\mathbf{Q}}(f)$ is contained in some conjugate of G in S_n. Although linear factors are easy to find, they give information only about the Galois group's containment in one group and its conjugates. The *complete* factorization of well-chosen resolvent polynomial can often determine $Gal_{\mathbf{Q}}(f)$ among possible candidates.

4.4 Linear Resolvent Polynomials

Linear resolvents form a general class of useful resolvent polynomials for $f(x)$ of any degree. Often the factorization of linear resolvents of relatively low degree can be used to determine $Gal_{\mathbf{Q}}(f)$. We may use linear resolvents to determine the orbit-length partition of r-sets or r-sequences under $Gal_{\mathbf{Q}}(f)$ $(1 \leqslant r \leqslant n)$.

4.4.1 r-sets and r-sequences

r-sets A subgroup G of S_n acts on the $\binom{n}{r}$ r-sets contained in $\{1, ..., n\}$, where the action is defined by $\{i_1, ..., i_r\}^\sigma = \{i_{1^\sigma}, ..., i_{r^\sigma}\}$ for all $\sigma \in G$. Now let $\Upsilon = x_1 + \cdots x_r$. It is clear that the action of G on Υ^{S_n} is equivalent to the action of G on the r-sets contained in $\{l, ..., n\}$. Thus the factorization of $\Re(\Upsilon, f)$ determines the orbit-length partition of r-sets under $Gal_{\mathbf{Q}}(f)$.
Erbach, Fischer, and McKay, cf. [EFM79] and [Mck79], suggest using resolvents of this type in order to determine the transitivity of $Gal_{\mathbf{Q}}(f)$ on r-sets.

The following remark is of interest: for f irreducible and $n = rs$, $r, s \neq 1$, $\Re(\Upsilon, f)$ has t irreducible factors of degree s if and only if $Gal_{\mathbf{Q}}(f)$ has t systems of imprimitivity of s blocks of size r, cf. definition 30.

r-sequences A subgroup G of S_n acts on the $\frac{n!}{(n-r)!}$ r-sequences of distinct elements of $(1, ..., n)$ where the action is defined by $(i_1, ..., i_r)^\sigma = (i_{1^\sigma}, ..., i_{r^\sigma})$ for all $\sigma \in G$. Now let $\Upsilon = e_1x_1 + \cdots + e_rx_r$, where $e_1, \ldots, e_r$ are *distinct* nonzero integers.

Now suppose $\Re(\Upsilon, f)$ has distinct zeros, then $\Re(\Upsilon, f)$ is reducible if and only if $Gal_{\mathbf{Q}}(f)$ is not r-ply transitive.

There is also a simple field-theoretic interpretation to the factorization of $\Re(\Upsilon, f)$. Let $b = e_1\alpha_{1^\sigma} + \cdots + e_r\alpha_{r^\sigma}$, $\sigma \in S_n$ be a zero of $\Re(\Upsilon, f)$. We see that $\mathrm{Stab}_{Gal_{\mathbf{Q}}(f)}(b) = \cap_{i=1}^{r} Stab_{Gal_{\mathbf{Q}}(f)}(\alpha_{i^\sigma})$; hence $\mathbf{Q}(b) = \mathbf{Q}(\alpha_{1^\sigma}, \cdots, \alpha_{r^\sigma})$. The degrees of the irreducible factors of $\Re(\Upsilon, f)$ correspond to the degrees over $\mathbf{Q}$ of nonconjugate subfields of $\mathrm{spl}_{\mathbf{Q}}(f)$ generated by r-sets of the zeros of f.

For irreducible f and $r = 2$, we note that $\Re(\Upsilon, f)$ has irreducible factors all of degree n if and only if $\mathbf{Q}(\alpha_i) = \mathbf{Q}(\alpha_j)$ for all $1 \leqslant i, j \leqslant n$ if and only if $\mathrm{spl}_{\mathbf{Q}}(f) = \mathbf{Q}(\alpha_i)$ for all $1 \leqslant i \leqslant n$ if and only if $Gal_{\mathbf{Q}}(f)$ is a regular permutation group, definition 28.

We also note that if $r = n - 1$ or $r = n$, then $\Re(\Upsilon, f)$ has degree $n!$ and $\mathrm{spl}_{\mathbf{Q}}(f) = \mathbf{Q}(b)$ for each zero b of $\Re(\Upsilon, f)$.

4.4.2 Data: Orbit-length Partitions

For the transitive permutation groups G of degree 3 to 7, Table 4.1 contains the orbit-length partitions of r-sets (r up to $\frac{1}{2}$ degree of G) and 2-sequences (with distinct elements) under G. This table was computed by Butler, [BM83] and [MR85], using the group-theoretical computer language CAYLEY (now it is called MAGMA). We can also use GAP, [GAP], since it is a free software. But how and of course—why?

For irreducible f of degree up to 7, Table 4.1 is used to determine candidates for $Gal_{\mathbf{Q}}(f)$ given the factorization of a linear resolvent which determines the orbit-length partition of r-sets or 2-sequences under $Gal_{\mathbf{Q}}(f)$.

G	2-sets	3-sets	2-sequences
		Degree 3	
$+A_3$			3^2
S_3			6
		Degree 4	
Z_4	$2,4$		4^3
$+V_4$	2^3		4^3
D_4	$2,4$		$4,8$
$+A_4$	6		12
S_4	6		12
		Degree 5	
$+Z_5$	5^2		5^4
$+D_5$	5^2		10^2
F_{20}	10		20
$+A_5$	10		20
S_5	10		20
		Degree 6	
Z_6	$3, 6^2$	$2, 6^3$	6^5
S_3	$3^3, 6$	$2, 6^3$	6^5
D_6	$3, 6^2$	$2, 6, 12$	$6, 12^2$
$+A_4$	$3, 12$	$4^2, 6^2$	$6, 12^2$
G_{18}	$6, 9$	$2, 18$	$6^2, 18$
G_{24}	$3, 12$	$6^2, 8$	$6, 12^2$
$+S_4/V_4$	$3, 12$	$4^2, 12$	$6, 24$
S_4/Z_4	$3, 12$	$8, 12$	$6, 24$
G_{36}^1	$6, 9$	$2, 18$	$12, 18$
$+G_{36}^2$	$6, 9$	$2, 18$	$12, 18$
G_{48}	$3, 12$	$8, 12$	$6, 24$
$+PSL_2(5)$	15	10^2	30
G_{72}	$6, 9$	$2, 18$	$12, 18$
$PGL_2(5)$	15	20	30
$+A_6$	15	20	30
S_6	15	20	30
		Degree 7	
$+Z_7$	7^3	7^5	7^6
D_7	7^3	$7^3, 14$	14^3
$+F_{21}$	21	$7^2, 21$	21^2
F_{42}	21	$14, 21$	42
$+PSL_3(2)$	21	$7, 28$	42
$+A_7$	21	35	42
S_7	21	35	42

Table 4.1: Orbit-length Partitions of Sets and Sequences under G

4.4.3 Constructing Linear Resolvents Symbolically

The resolvent algorithm revolves around the following observation of Trager, [Tra76]. Let $f(x) = (x - \alpha_1) \ldots (x - \alpha_n)$, $g(x) = (x - \beta_1) \ldots (x - \beta_m)$ be non-constant polynomials over the integers. Then the resultant (definition 21) eliminating y, $\mathrm{res}_y(f(y), g(x - y)) = \prod_{i=1}^{n} g(x - \alpha_i)$, is the degree nm monic polynomial having zeros $\alpha_i + \beta_j$ $(i = 1, \ldots, n; j = 1, \ldots, m)$.

The following notation is used in algorithm LRINT below, [Soi84]. Let t be a non-zero integer and $f(x)$ a monic polynomial of degree n.
Then we define $f_{(t)}(x) = t^n f(\frac{x}{t})$. Thus $f_{(t)}(x)$ is the monic polynomial whose zeros are t times those of $f(x)$.
Next we define $\mathrm{mult}(t, \Upsilon)$ to be the number of distinct terms of the (multivariate) polynomial Υ having the coefficient t.

Algorithm LRINT

Input: A monic integral polynomial $f(x)$ of positive degree n, and $\Upsilon = e_1x_1 + \cdots + e_rx_r$, where $r \le n$ and the e_i are non-zero integers.

Returns: $\Re(\Upsilon, f)$.

```
1.  if $r = 0$ ($\Upsilon \equiv 0$) then return ("$x$") and stop.
2.  if $r = 1$ then return ($f_{(e_1)}(x)$) and stop.
3.  Permute the labelling of $x_1, \ldots, x_n$ in $\Upsilon$ so that $\mathrm{mult}(e_r, \Upsilon) \le \mathrm{mult}(e_i, \Upsilon)$
    for $i = 1, ..., r$.        (This ensures that the degree of $u(x)$ in step 4
    is as samll as possible. Note that the symmetry allows relabelling of
    the variables of $\Upsilon$ without changing $\Re(\Upsilon, f)$.)
4.  set $\Upsilon' := e_1x_1 + \cdots + e_{r-1}x_{r-1}$ and set $u(x) := \Re(\Upsilon', f)$ (recursively).
5.  Let $a_1, \ldots, a_k$ be the $k$ (say) distinct elements of $\{e_1, \ldots, e_{r-1}\}$, and set
     $\Upsilon_i := \Upsilon' + e_rx_{i'}$ where $i'$ (not necessarily uniquely determined) is
     chosen so that $a_ix_{i'}$ is a term of $\Upsilon'$, $(i = 1, \ldots, k)$.
    If any of the $\Upsilon_i$ now have only $r - 2$ terms (i.e. $a_i + e_r = 0$), then
     relabel the variables of these $\Upsilon_i$ with $1, 2, ..., r-2$ to conform with the
     input rules for this algorithm.
6.  set $v(x) := \prod_{i=1}^{k} \Re(\Upsilon_i, f)^{c_i}$ (recursively), where
    $c_i = n - r + 2$ if $a_i + e_r = 0$, and $c_i = \mathrm{mult}(a_i + e_r, \Upsilon_i)$ otherwise.
     (Observe that $\mathrm{res}_y(u(y), f_{(e_r)}(x - y))/v(x) = \Re(\Upsilon, f)^c$, where
     $c = \mathrm{mult}(e_r, \Upsilon)$.)
7.  set $c := \mathrm{mult}(e_r, \Upsilon)$ and $m := (n \deg(u) - \deg(v))/c + 1$.
    [$m - 1 = \deg(\Re(\Upsilon, f))$]
    Choose distinct integers $s_1, \ldots, s_m$ such that for $i = 1, \ldots, m : v(s_i) \ne 0$
    and if $c$ is even then $|s_i| > |\theta|$ for any zero $\theta$ of $R(\Upsilon, f)$.
    (A bound on the magnitude of the zeros of $R(\Upsilon, f)$ is calculated by
    bounding the magnitude of the zeros of $f$.)
8.  for $i = 1, ..., m$ : set $t_i := res_y(u(y), f_{(e_r)}(s_i - y))/v(s_i)$.
9.  (For non-negative real $t$, let $t^{1/c}$ denote the non-negative real $c$-th root of $t$.)
    for $i = 1, ..., m$ :
     if $t_i < 0$ then set $t_i := -|t_i|^{1/c}$
     else if $c$ is even and $m - 1$ is odd and $s_i < 0$ then set $t_i := -t_i^{1/c}$
     else set $t_i := t_i^{1/c}$.
     ( now $t_i = \Re(\Upsilon, f)(s_i)$ )
10. set $w(x)$ to be the unique polynomial of degree (at most) $m - 1$ such that
     $t_i = w(s_i)$ for $i = 1, \ldots, m$.
    return ($w(x)$) and stop.
```

4.4.4 Examples

Example 1. Consider $f(x) = x^7 - 14x^5 + 56x^3 - 56x + 22$.

$\mathrm{disc}(f) = 2^6 7^{10}$; f is irreducible over $\mathbf{Q}$. Compute and factorize $\mathfrak{R} = \mathfrak{R}(x_1 + x_2 + x_3, f)$ of degree 35 to determine the orbit-length partition of 3-sets under $Gal_{\mathbf{Q}}(f)$. Factorizing $\mathfrak{R}$ into irreducible factors over $\mathbf{Q}$, we find that $\mathfrak{R} = \mathfrak{R}_1\mathfrak{R}_2\mathfrak{R}_3$, where

$$\begin{aligned}
\mathfrak{R}_1 = {} & x^7 - 28x^5 + 224x^3 - 448x + 94, \\
\mathfrak{R}_2 = {} & x^7 - 28x^5 + 224x^3 - 448x + 192, \text{ and} \\
\mathfrak{R}_3 = {} & x^{21} - 84x^{19} + 2436x^{17} - 31136x^{15} + 6358x^{14} + 203840x^{13} - 84392x^{12} \\
& -733824x^{11} + 420728x^{10} + 1480192x^9 - 988064x^8 - 1652036x^7 \\
& +1138368x^6 + 986496x^5 - 620928x^4 - 284032x^3 + 137984x^2 \\
& +27104x - 10648.
\end{aligned}$$

$\mathfrak{R}$ has distinct zeros and its factorization shows that the orbit-length partition of 3-sets under $Gal_{\mathbf{Q}}(f)$ is 7^2, 21. From Table 4.1 we see that $Gal_{\mathbf{Q}}(f)$ is $+F_{21}$, the Frobenius group of order 21 on 7 letters. □

Example 2. Consider $f(x) = x^7 - 7x^3 + 14x^2 - 7x + 1$.

f is irreducible and $\mathrm{disc}(f) = 7^8 17^2$, thus $Gal_{\mathbf{Q}}(f)$ is a transitive subgroup of A_7. Letting $\Upsilon = x_1 + x_2 + x_3$, compute and factorize $\mathfrak{R}(\Upsilon, f)$ of degree 35 to determine the orbit-lengths of the action of $Gal_{\mathbf{Q}}(f)$ on the 3-subsets of $\{1, ..., 7\}$. It takes 6 minutes to compute $\mathfrak{R}(\Upsilon, f)$ using LRINT, p.71 on the PDP-11/34.

The factorization of $\mathfrak{R}(\Upsilon, f)$, of degree 35, takes approximately 10 minutes on the PDP-11/34. The degree 7 factor is $x^7 - 14x^4 + 7x^3 + 14x^2 - 56x - 32$. $\mathfrak{R}(\Upsilon, f)$ is found to have irreducible factors of degrees 7 and 28 which proves that $Gal_{\mathbf{Q}}(f)$ is $+PSL_3(2)$ from Table 4.1. □

Soicher-McKay remarked that the PDP-11/34 minicomputer was much slower than a typical large main-frame computer.

> *Please see Appendix A for the demonstration of the author's MAPLE program on these examples—It took two seconds to obtain an answer on his cheap Pentium-133.*

4.5 Further techniques

For $n = \deg(f) = 3, 4, 5, 7$, the conjugacy class in S_n of transitive $Gal_{\mathbf{Q}}(f)$ is determined completely by the "squareness" of $\mathrm{disc}(f)$, (theorem 42) and the orbit-lengths of the action of $Gal_{\mathbf{Q}}(f)$ on 2-sets, 3-sets, and 2-sequences, with the exception of distinguishing F_{20} from S_5.

For degree 6, all the transitive groups can be differentiated by disc(f) and the orbit-lengths on 2-sets, 3-sets, and 2-sequences except to distinguish S_4/Z_4 from G_{48}, G_{36}^1 from G_{72}, and $PGL_2(5)$ from S_6.

4.5.1 Quadratic Resolvents

The ability to compute linear resolvents efficiently allows us to compute certain useful quadratic resolvents. When $\Re(\Upsilon, f)$ is a resolvent such that for some $\Upsilon^\sigma = -\Upsilon$ for some $\sigma \in S_n$, we see that $\Re(\Upsilon^2, f)(x^2) = \Re(\Upsilon, f)(x)$.

Suppose $\deg(f) = 5$ and $Gal_{\mathbf{Q}}(f)$ is either F_{20} or S_5. We compute and factorize $\Re = \Re((x_1 + x_2 - x_3 - x_4)^2, f)$ of degree 15, using a linear resolvent, to distinguish between these candidates. Now $Gal_{\mathbf{Q}}(f) = F_{20}$ if and only if $\Re$ is reducible. In this case $\Re$ has irreducible factors of degrees 5 and 10.

4.5.2 Factorization over $\mathbf{Q}(\sqrt{\text{disc}(f)})$

The factorization of $\Re(\Upsilon, f)$ over $\mathbf{Q}(\sqrt{\text{disc}(f)})$ when disc(f) is not a square is also useful to determine the conjugacy class of $Gal_{\mathbf{Q}}(f)$ in S_n.

We assume that all polynomials discussed have distinct zeros.

Let E be $\text{spl}_{\mathbf{Q}}(f)$ and $\Gamma = Gal_{\mathbf{Q}}(f)$ as a subgroup in S_n. Suppose $g(x)$ is a monic irreducible factor of a resolvent polynomial $\Re(\Upsilon, f)$ such that $\Upsilon_j(\alpha_1, \ldots, \alpha_n)$ is a zero of g for a specific $\Upsilon_j \in \Upsilon^{S_n}$ ($\alpha_1, \ldots, \alpha_n$ the zeros of f).

Let d be the squarefree part of disc(f), define $g_d(x)$ to be the monic integral polynomial of degree $2 \cdot \deg(g)$ having the zeros $b_k \pm d^{\frac{1}{2}}$, where the b_k run through the zeros of g.

Theorem 68 *The following are equivalent:*
(1) $Stab_\Gamma(\Upsilon_j)$ *is a subgroup of* A_n.
(2) $\mathbf{Q}(\Upsilon_j(\alpha_1, \ldots, \alpha_n))$ *contains* $\mathbf{Q}(\sqrt{\text{disc}(f)})$.
(3) $g(x)$ *is reducible over* $\mathbf{Q}(\sqrt{\text{disc}(f)})$.
(4) $g_d(x)$ *is reducible over* $\mathbf{Q}$.

Proof. The equivalence of (1) and (2) follows immediately from the fundamental theorem of Galois theory and the observation that $\mathbf{Q}(\Upsilon_j(\underline{\alpha})) = E^{Stab_\Gamma(\Upsilon_j)}$ (as $\Re(\Upsilon, f)$ has distinct roots, cf. theorem 39) and $\mathbf{Q}(\sqrt{\text{disc}(f)}) = E^{\Gamma \cap A_n}$.

Let $\Delta = \sqrt{\text{disc}(f)}$, and h be the minimial polynomial of $\Upsilon_j(\underline{\alpha})$ over $\mathbf{Q}(\Delta)$. g is irreducible over $\mathbf{Q}(\Delta)$
iff $g = h$

iff $[\mathbf{Q}(\Upsilon_j(\underline{\alpha})):\mathbf{Q}] = [\mathbf{Q}(\Delta, \Upsilon_j(\underline{\alpha})):\mathbf{Q}(\Delta)]$
iff $[\mathbf{Q}(\Upsilon_j(\underline{\alpha}), \Delta):\mathbf{Q}(\Upsilon_j(\underline{\alpha})] = [\mathbf{Q}(\Delta):\mathbf{Q}] = 2$
iff $\mathbf{Q}(\Delta) \not\subseteq \mathbf{Q}(\Upsilon_j(\underline{\alpha}))$. Hence (2) and (3) are equivalent.

Finally from theorem 51, we have: if $\mathrm{norm}_{\mathbf{Q}(\Delta)/\mathbf{Q}}(g(x,\Delta))$ is squarefree, $g(x,\Delta)$ is irreducible over $\mathbf{Q}(\Delta)$ iff $\mathrm{norm}_{\mathbf{Q}(\Delta)/\mathbf{Q}}(g(x,\Delta))$ is irreducible over $\mathbf{Q}$. The norm of g is clearly not squarefree, so we consider a linear transformation of the form $x \mapsto x - \Delta$. We let $h(x,\Delta) = g(x-\Delta)$ and consider its norm. It is convenient to use $d^{\frac{1}{2}}$, where d is the squarefree part of disc(f), in place of Δ in the construction, since the corresponding norm will have smaller coefficients and hence be easier to factor. Thus the polynomial g_d is reducible over $\mathbf{Q}$ iff $\mathrm{norm}_{\mathbf{Q}(\Delta)/\mathbf{Q}}(h(x,\Delta))$ is reducible over $\mathbf{Q}$ iff $h(x,\Delta)$ is reducible over $\mathbf{Q}(\Delta)$ iff $g(x)$ is reducible over $\mathbf{Q}(\sqrt{\mathrm{disc}(f)})$. Hence (3) is equivalent to (4). ■

Now suppose $n = 6$ and $\Re = \Re(x_1 + x_2 + x_3, f)$ of degree 20.

Suppose $Gal_{\mathbf{Q}}(f) = S_4/Z_4$ or G_{48}. Let g be the monic irreducible factor of degree 12 (how?) of $\Re$. Then $Gal_{\mathbf{Q}}(f) = S_4/Z_4$ if and only if g_d is reducible.

Suppose $Gal_{\mathbf{Q}}(f) = G^1_{36}$ or G_{72}. Let g be the monic irreducible factor of degree 2 of $\Re$. Then $Gal_{\mathbf{Q}}(f) = G^1_{36}$ if and only if g_d is reducible.

Suppose $Gal_{\mathbf{Q}}(f) = PGL_2(5)$ or S_6. Let $g = \Re$. Then $Gal_{\mathbf{Q}}(f) = PGL_2(5)$ if and only if g_d is reducible.

4.6 Application to the Inverse Galois Problem

"Given a permutation group G, to find polynomials over a certain field whose Galois group are G" is called the **Inverse Galois Problem.** There have been a huge amount of literature on this problem, cf. the citations in Appendix B.1 no.5&6 or simply search "inverse Galois" in the MathSci disc on the CD-rom.

It is an unsolved problem whether any permutation group can appear as the Galois group of a polynomial over $\mathbf{Q}$. For each solvable group G it is known that there exists a polynomial f such that $Gal_{\mathbf{Q}}(f) = G$, [Sha54]; however there has not yet appeared a practical general method of constructing an f from any given solvable group G.

For each transitive permutation group G of degree 3 to 7, Soicher and McKay have computed a polynomial $f(x)$ such that $Gal_{\mathbf{Q}}(f) = G$. These polynomials appear in Table 4.2 below where ω_n denotes a primitive nth root of unity. (For each polynomial f in Table 4.2 one can prove that $Gal_{\mathbf{Q}}(f)$ is the group indicated by the algorithms mentioned above.)

Many of the polynomials f are constructed so that $\mathrm{spl}_{\mathbf{Q}}(f)$ is contained in some known field. The methods of doing this include constructing f to be a resolvent polynomial, constructing f to be a composite polynomial, or if $Gal_{\mathbf{Q}}(f)$ is to be cyclic, by constructing f such that $\mathrm{spl}_{\mathbf{Q}}(f)$ is contained in $\mathbf{Q}(\omega_p)$, p prime. This knowledge about $\mathrm{spl}_{\mathbf{Q}}(f)$ is used to reduce or eliminate the work necessary to determine $Gal_{\mathbf{Q}}(f)$.

The only polynomials whose Galois groups are determined using other information than the splitting field, cycle types, or discriminant are those f with $Gal_{\mathbf{Q}}(f) = D_5$, D_7, F_{21}, or $PSL_3(2)$. These exceptions are proved to have the group indicated by using the factorization of appropriate linear resolvent polynomials, Table 4.1.

Given G, to find monic integral $f(x)$ such that $Gal_{\mathbf{Q}}(f) = G$, where it is nontrivial to construct an appropriate splitting field, we do computer searching. If G is a group of even permutations, we first seek f such that $\mathrm{disc}(f)$ is a square. We also search for f such that, for all primes p in a fixed set, either $p \mid \mathrm{disc}(f)$ or the degree partition of f mod p is the cycle type of some permutation in G.

There has been interest in polynomials with $PSL_3(2)$ (isomorphic to $PSL(2,7)$) as Galois group over $\mathbf{Q}$, cf. the citations in [SM85]. The new example in Table 4.2 has the property that its discriminant, $7^8 17^2$, is the smallest discriminant of any monic integral polynomial with Galois group $PSL_3(2)$ over the rationals of which Soicher and McKay were aware.

This table has also been used as the TEST DATA of the author's MAPLE program on computing Galois group of polynomials up to degree 7.

G	disc(f)	$f(x)$	Remarks
		Degree 3	
$+A_3$	7^2	x^3+x^2-2x-1	spl(f)=$\mathbf{Q}(\varpi_7+\varpi_7^{-1})$
S_3	$-2^2 3^3$	x^3+2	
		Degree 4	
Z_4	5^3	$x^4+x^3+x^2+x+1$	spl(f)=$\mathbf{Q}(\varpi_5)$
$+V_4$	2^8	x^4+1	spl(f)=$\mathbf{Q}(\varpi_8)$
D_4	-2^{11}	x^4-2	
$+A_4$	$2^{12}3^4$	$x^4+8x+12$	
S_4	229	x^4+x+1	
		Degree 5	
$+Z_5$	11^4	$x^5+x^4-4x^3-3x^2+3x+1$	spl(f)=$\mathbf{Q}(\varpi_{11}+\varpi_{11}^{-1})$
$+D_5$	$2^{12}5^6$	$x^5-5x+12$	
F_{20}	$2^4 5^5$	x^5+2	
$+A_5$	$2^{16}5^6$	$x^5+20x+16$	
S_5	$19\cdot151$	x^5-x+1	
		Degree 6	
Z_6	-7^5	$x^6+x^5+x^4+x^3+x^2+x+1$	spl(f)=$\mathbf{Q}(\varpi_7)$
S_3	$-2^{16}3^{21}$	x^6+108	spl(f)=spl(x^3+2)
D_6	$-2^{11}3^6$	x^6+2	
$+A_4$	$2^6 3^8$	x^6-3x^2-1	spl(f)=spl($x^4+8x+12$)
G_{18}	-3^{11}	x^6+3x^3+3	
G_{24}	$-2^6 3^8$	x^6-3x^2+1	$Gal_{\mathbf{Q}}(x^3-3x+1)=A_3$
$+S_4/V_4$	$2^6 229^2$	x^6-4x^2-1	spl(f)=spl(x^4+x+1)
S_4/Z_4	229^3	$x^6-3x^5+6x^4-7x^3+2x^2+x-4$	spl(f)=spl(x^4+x+1)
G_{36}^1	$2^8 3^9$	x^6+2x^3-2	
$+G_{36}^2$	$2^{10}3^6 5^4$	$x^6+6x^4+2x^3+9x^2+6x-4$	f(x)=$(x^3+3x+1)^2-5$
G_{48}	$-2^{11}5^2 7^2$	x^6+2x^2+2	
$+PSL_2(5)$	$2^{36}5^8$	$x^6+10x^5+55x^4+140x^3+175x^2+170x+25$	spl(f)=spl($x^5+20x+16$)
G_{72}	$-2^8 733$	$x^6+2x^4+2x^3+x^2+2x+2$	f(x)=$(x^3+x+1)^2+1$
$PGL_2(5)$	$5^{20}19^3 151^3$	$x^6+10x^5+55x^4+140x^3+175x^2-3019x+25$	spl(f)=spl(x^5-x+1)
$+A_6$	$2^{16}3^6 5^6$	$x^6+24x-20$	
S_6	$-101\cdot431$	x^6+x+1	
		Degree 7	
$+Z_7$	$17^2 29^6$	$x^7+x^6-12x^5-7x^4+28x^3+14x^2-9x+1$	spl(f)=$\mathbf{Q}(\varpi_{29}+\varpi_{29}^{12}+\varpi_{29}^{-1}+\varpi_{29}^{-12})$
D_7	$-3^6 7^9$	$x^7+7x^3+7x^2+7x-1$	
$+F_{21}$	$2^6 7^{10}$	$x^7-14x^5+56x^3-56x+22$	
F_{42}	$-2^6 7^7$	x^7+2	
$+PSL_3(2)$	$7^8 17^2$	$x^7-7x^3+14x^2-7x+1$	
$+A_7$	$3^6 7^8$	$x^7+7x^4+14x+3$	
S_7	$-11\cdot239\cdot331$	x^7+x+1	

Table 4.2: Polynomials $f(x)$ such that $Gal_{\mathbf{Q}}(f)=G$

4.7 Comment

- The resolvents are relatively simple and can be calculated symbolically. But rather than to test for a linear factor, as was the case in Stauduhar's method, the resolvents are completely factored (by the polynomial factorization algorithm in Chapter 3). Hence no round-off error problem exists.

- A few resolvents are needed to distinguish all the groups of a given degree, and for each group, its shape (cycle-structures of its elements and their number of occurrences) and orbit-length partitions of r-sets and 2-sequences under its action are all is required. So the storage requirements are low.

- Traversing down the subgroup lattice is not necessary in this method.

However, the degree of some resolvents are relatively higher than that of Stauduhar's method and factorization can take some time. We also observe that this method only give the Galois group of $f(x)$ to within conjugacy, while Stauduhar's method can output the exact permutations in the Galois group with respect to an ordering of the roots.

Appendix A

Demonstration of the MAPLE program

Example 1

```
> t1:=time();
> gal3_7(x^6-42*x^4+80*x^3+441*x^2-1680*x+4516); t2:=time()-t1;
                                   t1 := 0
  [stuff deleted]
            factored := (x - 6186) (x + 55050) (x + 66390)
                  3          2
                (x  - 159498 x  + 7638068124 x - 95259900647448)
                           olpart := [1, 1, 1, 3]
                 ~S3-, 6, {(1 3 5)(2 4 6), (1 6)(2 5)(3 4)}
                                 t2 := 2.000
```

Example 2

```
> t1:=time();
> gal3_7(x^6-32*x^4+160*x^3-320*x^2+384*x-256); t2:=time()-t1;
                                 t1 := 2.000
{--> enter gal3_7, args = x^6-32*x^4+160*x^3-320*x^2+384*x-256
                     6       4        3        2
               g := x  - 32 x  + 160 x  - 320 x  + 384 x - 256
                                  x := {x}
                                   x := x
                                   n := 6
                      6       4        3        2
                g := x  - 32 x  + 160 x  - 320 x  + 384 x - 256
{--> enter factor, args = x^6-32*x^4+160*x^3-320*x^2+384*x-256
                   6       4        3        2
                  x  - 32 x  + 160 x  - 320 x  + 384 x - 256
<-- exit factor (now in gal3_7) = x^6-32*x^4+160*x^3-320*x^2+384*x-256}
{--> enter whattype, args = x^6-32*x^4+160*x^3-320*x^2+384*x-256
                                       +
<-- exit whattype (now in gal3_7) = +}
                                   lc := 1
{--> enter gal6, args = x^6-32*x^4+160*x^3-320*x^2+384*x-256, x
                                 Digits := 67
                            D := 403780252137947136
                              is_sqrdisc := true
                      6       4        3        2
                h := x  - 32 x  + 160 x  - 320 x  + 384 x - 256
hordroots := [
-7.689949640495598501488201177134623564814635134931898297770890531635,
.5173040450083055256719890487905706579074948061084162141542651230727
    - 1.342241671626710863100292671217310039105979613780809918911617444338 I
   ,
.5173040450083055256719890487905706579074948061084162141542651230727
    + 1.342241671626710863100292671217310039105979613780809918911617444338 I
   ,
1.587201157033524323661010132917740178730405336185231560200216840147,
2.534070196722731563241606473317871035134620093264917154631071722671
    - 1.927373608040885986907299473213880287485276562886169097251622162908 I
   ,
2.534070196722731563241606473317871035134620093264917154631071722671
    + 1.927373608040885986907299473213880287485276562886169097251622162908 I
]
                  2   2                    2   2
         F := x1  x5  (x2 x4 + x3 x6) + x2  x4  (x5 x1 + x3 x6)
                  2   2                    2   2
             + x3  x6  (x5 x1 + x2 x4) + x1  x6  (x5 x2 + x3 x4)
                  2   2                    2   2
             + x2  x5  (x1 x6 + x3 x4) + x3  x4  (x1 x6 + x5 x2)
                  2   2                    2   2
             + x1  x3  (x2 x6 + x4 x5) + x2  x6  (x1 x3 + x4 x5)
                  2   2                    2   2
             + x4  x5  (x1 x3 + x2 x6) + x1  x4  (x2 x3 + x5 x6)
                  2   2                    2   2
             + x2  x3  (x4 x1 + x5 x6) + x5  x6  (x4 x1 + x2 x3)
                  2   2                    2   2
             + x1  x2  (x3 x5 + x4 x6) + x3  x5  (x1 x2 + x4 x6)
                  2   2
             + x4  x6  (x1 x2 + x3 x5)
    S6_PGL2_5 := [[], [[1, 2]], [[1, 3]], [[1, 4]], [[1, 5]], [[1, 6]]]
                                           5                 4                3
  R := - 756591543258710016 x - 15872 x  + 118030336 x  - 449696497664 x
                         2     6
     + 859354236452864 x  + x  + 247809443596154699776
    enter  tschirnhausen
----------  tschirnhausen succeeded
                                2          3        4       5    6   2
 t := 12122 x + 9121 - 1233 x  - 4020 x  + 207 x  - 70 x  + x , x  - x + 1
                                    2          3        4       5    6
        h := 12122 x + 9121 - 1233 x  - 4020 x  + 207 x  - 70 x  + x
hordroots := [
67.82527511385398313794128326451992353676532559733215490276600331461,
 - 1.051313275077617650255922167899218914481077053633602918684992434809
     - .046452420595703701277570891601329043317124383909884988452833171337 I
    ,
 - 1.051313275077617650255922167899218914481077053633602918684992434809
     + .046452420595703701277570891601329043317124383909884988452833171337 I
    ,
1.932006355855034015944909264870709980123515849342715652158401064852,
1.172672540223109073312825903203902156036656330296167641222790245078
    - 7.840826428131851752184377821475726915283115896754373692279610003821 I
    ,
1.172672540223109073312825903203902156036656330296167641222790245078
    + 7.840826428131851752184377821475726915283115896754373692279610003821 I
]
                  2   2                    2   2
         F := x1  x5  (x2 x4 + x3 x6) + x2  x4  (x5 x1 + x3 x6)
                  2   2                    2   2
             + x3  x6  (x5 x1 + x2 x4) + x1  x6  (x5 x2 + x3 x4)
                  2   2                    2   2
             + x2  x5  (x1 x6 + x3 x4) + x3  x4  (x1 x6 + x5 x2)
                  2   2                    2   2
             + x1  x3  (x2 x6 + x4 x5) + x2  x6  (x1 x3 + x4 x5)
                  2   2                    2   2
             + x4  x5  (x1 x3 + x2 x6) + x1  x4  (x2 x3 + x5 x6)
                  2   2                    2   2
             + x2  x3  (x4 x1 + x5 x6) + x5  x6  (x4 x1 + x2 x3)
                  2   2                    2   2
             + x1  x2  (x3 x5 + x4 x6) + x3  x5  (x1 x2 + x4 x6)
                  2   2
             + x4  x6  (x1 x2 + x3 x5)
    S6_PGL2_5 := [[], [[1, 2]], [[1, 3]], [[1, 4]], [[1, 5]], [[1, 6]]]
                                                                   5
         R := - 1098482633805498572208546614636217 6 x - 6442036 x
                                 4                            3
             + 115129608233404 x  - 54666423000593249315 2 x
                                              2    6
             + 341737456787647234957854641 6 x  + x
             + 109053689404991213261010585725583693066 88
               4            3                  2
 factored := (x  - 2253176 x  + 101339045990168 x  - 112362603136642134496 x
                                             2
     + 2505643050485262159681180688) (x  - 4188860 x + 4352323423876)
                               olpart := [2, 4]
 <-- exit gal6 (now in gal3_7) = +S4/V4  ~S4 , 24, {(1 3 5)(2 4 6),
(2 6)(3 5), (2 5 )(3 6)}}
        +S4/V4  ~S4 , 24, {(1 3 5)(2 4 6), (2 6)(3 5), (2 5 )(3 6)}
<-- exit gal3_7 (now at top level) = +S4/V4  ~S4 , 24, {(1 3 5)(2 4 6),
(2 6)(3 5), (2 5 )(3 6)}}
        +S4/V4  ~S4 , 24, {(1 3 5)(2 4 6), (2 6)(3 5), (2 5 )(3 6)}
                                   t2 := 2.000
```

Example 3

```
> t1:=time();
> gal3_7(x^7-14*x^5+56*x^3-56*x+22); t2:=time()-t1;
                                   t1 := 4.000
  [stuff deleted]
                +F21, 21, {(1 2 3 4 5 6 7), (2 3 5)(4 7 6)}
                                   t2 := 2.000
```

Example 4

```
> t1:=time();
> gal3_7(x^7-7*x^3+14*x^2-7*x+1); t2:=time()-t1;
                                   t1 := 5.000
{--> enter gal3_7, args = x^7-7*x^3+14*x^2-7*x+1
                                7      3       2
                          g := x  - 7 x  + 14 x  - 7 x + 1
                                      x := {x}
                                       x := x
                                       n := 7
                                7      3       2
                          g := x  - 7 x  + 14 x  - 7 x + 1
{--> enter factor, args = x^7-7*x^3+14*x^2-7*x+1
                               7      3       2
                              x  - 7 x  + 14 x  - 7 x + 1
<-- exit factor (now in gal3_7) = x^7-7*x^3+14*x^2-7*x+1}
{--> enter whattype, args = x^7-7*x^3+14*x^2-7*x+1
                                              +
<-- exit whattype (now in gal3_7) = +}
                                     lc := 1
{--> enter gal7, args = x^7-7*x^3+14*x^2-7*x+1, x
                                 D := 1666027489
                              is_sqrdisc := true
                              7      3       2
                         h := x  - 7 x  + 14 x  - 7 x + 1
     hordroots := [-1.9966549098101690469620688277997499939 14,
               - .40140619593665502438391926272462314 20387
                    - 1.76309647656123944665881256197315020 3443 I,
               - .40140619593665502438391926272462314 20387
                    + 1.76309647656123944665881256197315020 3443 I,
             .26310402678294471083363866241226747 81262,
             .40699794271912432449565553383749095 21301,
             1.06468266609061503020030657849961892 3868
                    - .54490170629706613688754271622303535 06364 I,
             1.06468266609061503020030657849961892 3868
                    + .54490170629706613688754271622303535 06364 I]
        t := [-.53970836509018533793419986303697880 59512,
              .92638049693699471665002597818726325 99555
                    + 2.30799818285830558354635527819618555 4079 I,
              - 2.13495707896378936051234942811210565 7827
                    + 1.76309647656123944665881256197315020 3443 I,
              .92638049693699471665002597818726325 99555
                    + 1.21819477026417330977126984575011485 2807 I,
              .26869577356550401094537493352513528 82176
                    + 1.76309647656123944665881256197315020 3443 I,
              .26187027421748498143246805305037263 97906
                    + .54490170629706613688754271622303535 0636 I,
              -2.79946730168329909572990735324899627 7991,
              .26187027421748498143246805305037263 97906
                    - .54490170629706613688754271622303535 0636 I,
              -.39581444915400572427218299161175533 19473,
              - 1.33337843965611904114568151202475421 2085
                    + 2.30799818285830558354635527819618555 4079 I,
              1.72795913624466503601669389427461470 5697
                    + 1.76309647656123944665881256197315020 3443 I,
              1.07027441287317433031204284961248673 3959
                    + 2.30799818285830558354635527819618555 4079 I,
            - 1.3333784396556119041145681512024754212085
                 + 1.2181947702641733097712698457501148852807 I,
            - 1.9910631630276097468503325566868882183823
                 + 1.7630964765561239446658812561973150203443 I,
            1.0702744128731743303120428496124867339959
                 + 1.2181947702641733097712698457501148852807 I,
            .9263804969369947166500259781872632599555
                 - 1.2181947702641733097712698457501148852807 I,
            - 2.1349570789637893605123494281121056578271
                 - 1.7630964765561239446658812561973150203443 I,
            .9263804969369947166500259781872632599555
                 - 2.3079981828583055835463552781961855554079 I,
            .2686957735655040109453749335251352882176
                 - 1.7630964765561239446658812561973150203443 I,
            - .668868216936609305928123586887863591920
                 + .5449017062970661368875427162230353506364 I,
            2.392469358964174771234251819411505325862,
            1.734784635592684065529600774749377354124
                 + .5449017062970661368875427162230353506364 I,
            - .668868216936609305928123586887863591920
                 - .5449017062970661368875427162230353506364 I,
            -1.326552940308100011632774631549991563658,
            1.734784635592684065529600774749377354124
                 - .5449017062970661368875427162230353506364 I,
            - 1.3333784396556119041145681512024754212085
                 - 1.2181947702641733097712698457501148852807 I,
            1.727959136244665036016693894274614705697
                 - 1.7630964765561239446658812561973150203443 I,
            1.0702744128731743303120428496124867339959
                 - 1.2181947702641733097712698457501148852807 I,
            - 1.3333784396556119041145681512024754212085
                 - 2.3079981828583055835463552781961855554079 I,
            - 1.9910631630276097468503325566868882183823
                 - 1.7630964765561239446658812561973150203443 I,
            1.0702744128731743303120428496124867339959
                 - 2.3079981828583055835463552781961855554079 I,
            .132710422371061013438544329199487853822,
            - .5249743010004296922661067154626401179159
                 + .5449017062970661368875427162230353506364 I,
            2.536363274900354384896268690836728799866,
            - .5249743010004296922661067154626401179159
                 - .5449017062970661368875427162230353506364 I]
                                   i := i
                                                 9  7          -33
R := .4576219190000000000000000000000000012*10   x   + .7*10     I
                                                   7
     + .4676223999999999999999999999999999931*10   x
                                                   9  5
     + .3191663999999999999999999999999999995*10   x
                                                   9  4
     + .1582566719999999999999999999999999996*10   x
                                                   9  3
     + .1189659519999999999999999999999999999*10   x
                                                   8  2
     + .5743919999999999999999999999999999996*10   x
                                                   9  6          -31    3
     + .5456530240000000000000000000000000010*10   x   + .5*10     I x
                   -29      20                                             8  8
     - .13646*10       I x    + .1969686599999999999999999999999999999521*10   x
                                                   8  9
     + .2260308399999999999999999999999999634*10   x
                                                   8  13
     + .3319044399999999999999999999999997825*10   x
                                                 21
     - 236272.0000000000000000000000000000007011 x
                                                   7  20
     + .2254098000000000000000000000000000001329*10   x
                                                   7  17
     + .3055527999999999999999999999999999991000*10   x
                                                   7  16
     + .3357760000000000000000000000000000001266*10   x
                                                   8  15
     - .2763514599999999999999999999999999998298*10   x
                                                   8  14
     + .3674891200000000000000000000000000000314*10   x
                                                   7  18
     + .1638363999999999999999999999999999998539*10   x
                                                   7  19
     - .3280990999999999999999999999999999997191*10   x
                                                   9  10
     + .2072131320000000000000000000000000000108*10   x
                                                   9  11
     + .1181074930000000000000000000000000000157*10   x
                                                   9  12
     - .1243717020000000000000000000000000000113*10   x
```

```
                                                       29
    - 279.99999999999999999999999999999999997423 x
                                                       28
    + 301.99999999999999999999999999999999994110 x
                                                       27
    + 294.00000000000000000000000000000000002833 x
                                                       26
    - 4900.0000000000000000000000000000000001114 x
                                                       25
    - 17052.000000000000000000000000000000003568 x
                                                       24
    + 60942.000000000000000000000000000000009582 x
                                                       23
    - 60773.999999999999999999999999999999977277 x
                                                       22          -38  34
    + 15357.999999999999999999999999999999937984 x     - .71*10     x
               -30    8            -29    9          -29    10            -29      7
    - .42*10      I x  + .794*10      I x  + .81*10     I x    - .233*10      I x
                  -36  33             -36  32            -30     4            -30    5
    + .1206*10       x    - .55511*10     x    + .14*10     I x   - .1*10      I x
               -30    6                                                          31
    - .29*10      I x  + 55.999999999999999999999999999999999998632 x
                                                       30         -32
    - 139.99999999999999999999999999999999999628 x      - .9*10     I x
              -31    2    35             -28     12             -29    13
    + .2*10      I x  + x    - .20101*10     I x    + .724*10      I x
                 -28     11              -28     16            -29    17
    - .1202*10      I x    - .13118*10      I x    - .1511*10     I x
                 -29     18             -28     14            -32    15
    + .5400*10      I x    + .2024*10      I x    + .504*10     I x
                  -35    31              -35    32            -37    33
    - .58782*10      I x    + .177587*10     I x    + .143*10     I x
                   -29     19              -29    21              -30    22
    + .144827*10      I x    - .102092*10      I x    + .40551*10      I x
                  -30     23            -32     26             -32    27
    + .28597*10      I x    + .76871*10     I x    - .11779*10     I x
                   -30     24               -31    25           -33    28
    - .104881*10      I x    - .2055784*10      I x    + .5383*10     I x
                  -33     29               -34   30
    + .17391*10      I x    - .7150755*10      I x
                                                              7          -38    34
    - .19758080000000000000000000000000000000016*10   - .20*10     I x
                   7                                        5                4
R := 457621919 x   + 4676224 x - 1975808 + 319166400 x  + 158256672 x
                  3              2               6              8                9
    + 118965952 x  + 57439200 x  + 545653024 x  + 19696866 x  + 22603084 x
                 13              21             20             17              16
    + 33190444 x   - 236272 x   + 2254098 x   + 3055528 x   + 3357760 x
                 15              14             18             19
    - 27635146 x   + 36748912 x   + 1638364 x   - 3280991 x
                  10               11                12            29          28
    + 207213132 x   + 118107493 x   - 124371702 x   - 280 x   + 302 x
            27          26            25            24            23            22
    + 294 x   - 4900 x   - 17052 x   + 60942 x   - 60774 x   + 15358 x
           31          30    35
    + 56 x   - 140 x   + x
                               olpart := [7, 28]
<-- exit gal7 (now in gal3_7) = +PSL2(F7) ~+PSL3(2), 168, {(1 2 3 4 5 6 7)
, (2 3)(4 7)}}
          +PSL2(F7) ~+PSL3(2), 168, {(1 2 3 4 5 6 7), (2 3)(4 7)}
<-- exit gal3_7 (now at top level) = +PSL2(F7) ~+PSL3(2), 168, {
(1 2 3 4 5 6 7), (2 3)(4 7)}}
          +PSL2(F7) ~+PSL3(2), 168, {(1 2 3 4 5 6 7), (2 3)(4 7)}
                               t2 := 2.000
```

Example 5

```
> t1:=time();
> gal3_7(x^5+x^4-4*x^3-3*x^2+3*x+1); t2:=time()-t1;
                                 t1 := 0
{--> enter gal3_7, args = x^5+x^4-4*x^3-3*x^2+3*x+1
                            5    4      3      2
                      g := x  + x  - 4 x  - 3 x  + 3 x + 1
                                x := {x}
                                x := x
                                n := 5
                            5    4      3      2
                      g := x  + x  - 4 x  - 3 x  + 3 x + 1
{--> enter factor, args = x^5+x^4-4*x^3-3*x^2+3*x+1
                       5    4      3      2
                      x  + x  - 4 x  - 3 x  + 3 x + 1
<-- exit factor (now in gal3_7) = x^5+x^4-4*x^3-3*x^2+3*x+1}
{--> enter whattype, args = x^5+x^4-4*x^3-3*x^2+3*x+1
                                                       +
<-- exit whattype (now in gal3_7) = +}
                                              lc := 1
{--> enter gal5, args = x^5+x^4-4*x^3-3*x^2+3*x+1, x
                                             D := 14641
                                         is_sqrdisc := true
                                  5     4       3       2
                             h := x  + x  - 4 x  - 3 x  + 3 x + 1
         hordroots := [-1.918985947228994779780736114132655398125,
              -1.309721467890570128113850144932587106368,
              -.2846296765465702808875853372327393375821,
              .8308300260037728510585482984592464070480,
              1.682507065662362337723623297838735435027]
 F := (x1 x2 + x2 x3 + x3 x4 + x4 x5 + x5 x1 - x1 x3 - x3 x5 - x5 x2 - x2 x4
      - x4 x1)^2
   S5_F20 := [[], [[1, 2, 3]], [[1, 3, 2]], [[1, 2]], [[2, 3]], [[1, 3]]]
               6         5           4             3              2
     t := x  - 264 x  + 25168 x  - 1022208 x  + 14992384 x  - 14992384 x,
        [31.78284454738512997240974738261312642005,
              1.077014593677007040114228609258101691156,
              95.66277585414092509135789505930621238498,
              64.55545037131997519140424134514375155263, 0,
              70.92191463347696270471388760367880795136]
                6         5           4             3              2
      R := x  - 264 x  + 25168 x  - 1022208 x  + 14992384 x  - 14992384 x
           Rordroots := [31.78284454738512997240974738261312642005,
                1.077014593677007040114228609258101691156,
                95.66277585414092509135789505930621238498,
                64.55545037131997519140424134514375155263, 0,
                70.92191463347696270471388760367880795136]
                        5           4             3              2
 factored := x (x  - 264 x  + 25168 x  - 1022208 x  + 14992384 x - 14992384)
                                         olpart := [1, 5]
                                            iroot := 0
                                         thisp := [[2, 3]]
         hordroots := [-1.918985947228994779780736114132655398125,
              -.2846296765465702808875853372327393375821,
              -1.309721467890570128113850144932587106368,
              .8308300260037728510585482984592464070480,
              1.682507065662362337723623297838735435027]
              t := [-1.918985947228994779780736114132655398125,
                   -.2846296765465702808875853372327393375821,
                   -1.309721467890570128113850144932587106368,
                   .8308300260037728510585482984592464070480,
                   1.682507065662362337723623297838735435027]
                 d := 121.00000000000000000000000000000000000
                                               d := 121
<-- exit gal5 (now in gal3_7) = +C5, 5, {(1 2 3 4 5)}}
                                       +C5, 5, {(1 2 3 4 5)}
<-- exit gal3_7 (now at top level) = +C5, 5, {(1 2 3 4 5)}}
                                       +C5, 5, {(1 2 3 4 5)}
                                             t2 := 1.000
```

Appendix B

Avenues for Further Exploration

B.1 Computational Galois Theory

1. John McKay and Richard Stauduhar, Finding Relations Among the Roots of an Irreducible Polynomial, in Proceedings ISSAC'97 (International Symposium on Symbolic and Algebraic Computation).

2. D. Casperson and J. McKay, "Symmetric Functions, m-sets, and Galois Groups", Mathematics of Computation, vol.63, no.208, 1994, pp.749-757.

3. Thomas W. Mattman, The computation of Galois groups over function fields, M.Sc. Mathematics Thesis, McGill University, Montréal, Canada, 1992.

4. T. W. Mattman and J. McKay, The computation of Galois groups over function fields, to appear in Mathematics of Computation.

5. Jean Pierre Serre, notes written by Henri Darmon, Topics in Galois Theory, Boston:Jones & Bartlett Publishers, 1992.

6. Bernd Heinrich Matzat, Konstruktive Galoistheorie, Lecture Notes in Mathematics, Band 1284, Springer, Heidelberg, 1987.

7. H. Darmon and D. Ford, Computational Verification of M_{11} and M_{12} as Galois Groups over $\mathbb{Q}$, Communications in Algebra, 17(12), pp.2941-2943, 1989.

8. Alexander Hulpke, Konstruktion transitiver Permutationsgruppen, Ph.D. Thesis, 1996, German, 165p, Verlag der Augustinus Buchhandlung Aachen, (ABM 18), Germany.

9. A. Hulpke, Block systems of a Galois group, Experimental Mathematics, volume 4, Number 1, p.1-9, 1995, English. (An algorithm to compute subfields as block systems of the Galois group.)

10. Jürgen Klüners, Über die Berechnung von Teilkörpern algebraischer Zahlkörper, Diplomarbeit, Technischen Universität Berlin, Berlin, Germany, 1994.

11. J. Klüners and M. Pohst, On Computing Subfields, to appear in Journal of Symbolic Computation.

12. John D. Dixon, Computing Subfields in Algebraic Number Fields, J. Australian Mathematical Society, Series A, 49, 1990, pp.434-448.

13. Yves Eichenlaub, Problèmes effectifs de théorie de Galois en degrés 8 à 11, Ph.D. Thèse, Université Bordeaux 1, Talence Cedex, France, 1996.

14. Frédéric Lehobey, Resolvent Computations by Resultants Without Extraneous Powers, in Proceedings ISSAC'97.

15. G. Kemper, Calculating Invariant Rings of Finite Groups over arbitrary Fields, Journal of Symbolic Computation 21, 1996, pp.351-366.

16. B. Sturmfels, Algorithms in Invariant Theory, Springer-Verlag, Wien, 1993.

17. William W. Adams and Philippe Loustaunau, An introduction to Gröbner bases, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, 1994.

18. K. Girstmair, On invariant Polynomials and their Application in Field Theory, Maths of Comp., vol. 48, no.178, 1987, pp.781-797.

19. K. Girstmair, On the computation of resolvents and Galois Groups, Manuscripta Mathematica, 43, pp.289-307, 1983.

20. Antoine Colin, "Théorie des invariants effective. Applications à la théorie de Galois et à la résolution de systèmes algébriques. Implantation en AXIOM," Thèse de doctorat (version provisoire), mai 1997, École Polytechnique, Palaiseau Cedex, France.

21. Antoine Colin, Théorie de Galois effective et implantation en AXIOM, Rapport de DEA, juillet 1994. Note Informelle de Calcul Formel numéro 66, École Polytechnique.

22. A. Colin, Relative resolvents and partition tables in Galois group computations, prépublication (version préliminaire, soumise à la conférence ISSAC'97). Note Informelle de Calcul Formel numéro 97.2, École Polytechnique, 1997.

23. A. Colin, An efficient symbolic algorithm to compute Lagrange resolvents for computational Galois theory, prépublication (version préliminaire, soumise à la conférence AAECC'12, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes). Note Informelle de Calcul Formel numéro 96.16, École Polytechnique, 1996.

24. A. Colin, Formal Computation of Galois Groups with Relative Resolvents, Actes de AAECC'11, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (Paris Juillet 1995), Lecture Notes in Computer Science 948, G. Cohen, M. Giusti, and T. Mora, Eds., Springer-Verlag, Berlin, 1995.

25. Annick Valibouze, Manipulations de fonctions symétriques, Thèse de l'Université Paris VI, 1988.

26. J.-M. Arnaudiès and A. Valibouze, Lagrange Resolvents, Conférence MEGA'96, (Effective Methods in Algebraic Geometry, Eindhoven, The Netherlands), à paraître dans Journal of Pure and Applied Algebra.

27. J.-M. Arnaudiès and A. Valibouze, Partial Computation for special Resolvents, soumis à Math. of Comp., 1996.

28. A. Valibouze, Computation of the Galois Groups of the Resolvent Factors for the Direct and Inverse Galois Problems, Actes de AAECC'11, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (Paris Juillet 1995), Lecture Notes in Computer Science 948, Springer-Verlag.

29. A. Valibouze, "Particular resolvents which are linear, symmetric or monomial", soumis à European Journal of Combinatorics, (1995).

30. A. Valibouze, "Modules de Cauchy, polynômes caractéristiques et résolvantes", Rapport interne LITP 95-62 (1995), soumis à Discrete Mathematics.

31. A. Valibouze, Sur l'arité des fonctions, European Journal of Combinatorics, 1993, Vol. 14, pp 359-372.

32. A. Valibouze, "Computing subfields: Reverse of the primitive element problem," MEGA'92, (Nice, France), published in Computational Algebraic Geometry, Progress in Mathematics vol.109, Birkhäuser, 1993, pp.163-176.

33. A. Valibouze, "Symbolic computation with symmetric polynomials, an extension to MACSYMA," Conférence Computers and Mathematics (1989, MIT, Cambridge, Mass.), Springer-Verlag, 308-320.

34. A. Valibouze, Résolvantes et fonctions symétriques, ISSAC'89 (Portland, Oregon), ACM Press, 390-399.

35. Marc Giusti, Daniel Lazard et A. Valibouze, "Algebraic transformations of polynomial equations, symmetric polynomials and elimination," ISSAC'88, (Roma, Italy), P. Gianni, ed., Lect. Notes in Comp. Sc. 358, 309-314.

36. A. Valibouze, Fonctions symétriques et changements de bases, EUROCAL'87, Leipzig, June 1987, published in Lecture Notes in Computer Science 378, Springer, Page 309-314.

37. J. Blömer, How to denest Ramanujan's nested radicals, Proc. 33rd Annual IEEE Symposium on Foundation of Computer Science, 1992, p.447-456.

38. A. Borodin, R. Fagin, J. Hopcroft, and M. Tompa, Decreasing the nesting depth of expressions involving square roots, J. Symbol. Comput., 1 (1985), pp.169-188.

39. R. Zippel, Simplification of expressions involving radicals, J. Symbol. Comput., 1 (1985), pp.189-210.

40. B. Caviness and R. Fateman, Simplification of radical expressions, in Proc. 1976 ACM Symposium on Symbolic and Algebraic Computation, ACM Press, New York, 1976, pp.329-338.

41. Susan Landau, Simplification of Nested Radicals, SIAM J. Comput., vol.21, no.1, 1992, pp.85-110.

42. Susan Landau and Gary Miller, Solvability by Radicals is in Polynomial Time, Journal of Computer and System Sciences, vol. 30, no. 2 (1985), pp. 179-208.

43. Ákos Seress, Introduction to computational group theory, Notices of the American Mathematical Society, June/July 1997.

44. Charles C. Sims, "Group-Theoretic Algorithms, A Survey", Proceedings of the International Congress of Mathematicians (Helsinki 1978), Acad. Sci. Fennica, Helsinki, 1980, pp. 979-985.

45. L. Babai, Computational complexity in finite groups, in Proceedings of the International Congress of Mathematicians (Kyoto, 1990), Math. Soc. Japan and Springer-Verlag, Toyko, Vol. I, II, pp.1479-1489.

46. C. C. Sims, Computation with permutation groups, Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles 1971 (S.R. Petrick, ed.), ACM Press, 1971, pp.23-28.

47. C. C. Sims, Determining the conjugacy classes of a permutation group, In Computers in Algebra and Number theory (Garrett Birkhoff and Marshall Hall, Jr, eds.), American Mathematical Society, Providence, RI, 1971, pp.191-195.

48. C. C. Sims, Computational methods in the study of permutation groups, In John Leech (ed.), Computational problems in abstract algebra, Pergamon press, 1970, pp.169-183.

49. M. Atkinson, An algorithm for finding the blocks of a permutation group, Math. Comp., July (1975), 911-13.

50. Martin Schönert and Ákos Seress, Finding blocks of imprimitivity in small base groups in nearly linear time, Proc. ISSAC'94, ACM Press, 1994.

51. M. Furst, J. Hopcroft, and E. Luks, Polynomial time algorithms for permutation groups, in Proc. Twenty-first Annu. IEEE Sympos. Found. Comput. Sci. 1980, pp.36-41.

52. Eugene M. Luks, Computing the composition factors of a permutation group in polynomial time, Combinatorica 7 (1987), 87-89.

53. Michael D. Atkinson (ed.), Computational group theory, Academic press, 1984.

54. Gregory Butler, Fundamental algorithms for permutation groups, Lecture Notes in Computer Science, Band 559, Springer-Verlag, 1991.

55. C. C. Sims, Computation with finitely presented groups, Cambridge University Press, 1994.

56. Susan Landau, Polynomial Time Algorithms for Galois Groups, in Proceedings of Eurosam'84, (J. Fitch, ed.), Springer-Verlag Lecture Notes in Computer Science no. 174, pp. 225-236.

57. James H. Davenport, Galois groups and the simplification of polynomials, to appear in Programmirovanie.

58. K.Yokoyama, A modular method for computing the Galois groups of polynomials, MEGA'96, (Effective Methods in Algebraic Geometry, at Eindhoven, The Netherlands), to appear in a special issue of the Journal of Pure and Applied Algebra.

59. H.Anai, M.Noro and K.Yokoyama, Computation of the splitting fields and the Galois groups of polynomials, MEGA'94, (U. Cantabria, Santander, Spain), published in Algorithms in Algebraic Geometry and Applications, Progress in Mathematics vol.143, Birkhäuser, 1996, pp.29-50.

60. H.Anai and K.Yokoyama, Radical representation of polynomial roots, submitted to J. Symbolic Computation. (Research Report ISIS-RR-94-13E, 1994, Institute for Social Information Science, Fujitsu Lab. Ltd., Japan.)

61. D. S. Dummit, Solving Solvable Quintics, Mathematics of Computation, vol.57, no.195, 1991, pp. 387-401.

62. K.Yokoyama, T.Takeshima and M.Noro, On determining the solvability of polynomials, ISSAC'90, published in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM PRESS, p.127-134, 1990.

63. K.Yokoyama, T.Takeshima and M.Noro, Computing primitive elements of extension fields, Journal of Symbolic Computation, vol.8, p.553-580, 1989.

64. Susan Landau, Factoring Polynomials over Algebraic Number Fields, SIAM J. Comput., vol.14, no.1, 1985, pp.184-195.

65. Mark J. Encarnación, "Faster Algorithms for Reconstructing Rationals, Computing Polynomial GCDs, and Factoring Polynomials", Ph.D. Thesis, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, 1995.

66. M. J. Encarnación, Factoring polynomials over algebraic number fields via norms, in Proceedings ISSAC'97.

67. M. J. Encarnación, The average number of modular factors in Trager's polynomial factorization algorithm, in Proceedings ISSAC'97.

68. M. J. Encarnación, On a modular algorithm for computing gcds of polynomials over algebraic number fields, in Proceedings ISSAC'94, pp.58-65, ACM Press, 1994.

69. M. J. Encarnación and G. E. Collins, Efficient rational number reconstruction, Journal of symbolic Computation 20, 287-297, 1995.

70. M. J. Encarnación and G. E. Collins, Improved techniques for factoring univariate polynomials, Journal of Symbolic Computation 21, 313-327, 1996.

71. P. Weinberger and L. Rothschild, Factoring Polynomials over Algebraic Number Fields, ACM Transactions on Mathematical Software, vol.2, no.4, 1976, pp.335-350.

72. A. K. Lenstra, Lattices and Factorization of Polynomials over algebraic number fields, in Proc. Eurocam 1982, Lecture Notes in Computer Science 144, Springer-Verlag, pp.32-39.

73. A. K. Lenstra, Factoring polynomials over algebraic number fields, in Proc. Eurocal 1983, Lecture Notes in Computer Science 162, Springer-Verlag, pp.458-465.

74. H. W. Lenstra, Algorithms in Algebraic Number Theory, Bulletin (New Series) of the American Mathematical Society, vol.26, no.2, 1992, pp. 211-244. (given as a Progress in Mathematics Lecture, August 8-10, 1991, meeting of American Mathematical Society in Orono, Maine.)

75. Harold M. Edwards, "Kronecker, Galois and Symbolic Computation", Invited lecture presented in ISSAC'96, International Symposium of Symbolic and Algebraic Computation, (at Eidgenössische Technische Hochschule Zürich.)

B.2 Notes on SAC—Symbolic and Algebraic Computation

SAC = Symbolic and Algebraic Computation = Computer Algebra = CA

Symbolic and Algebraic Computation, or Computer algebra or Computational algebra, is devoted to the investigation of algorithms, computational methods, software systems, and computer languages, oriented to scientific computations performed on *exact* and often *symbolic* data, by manipulating formal expressions by means of the algebraic rules they satisfy.

It studies such problems from three different but confluent viewpoints:
a) development and analysis of algebraic algorithms (from the viewpoints both of practical performance and of theoretical complexity);
b) design and analysis of software systems for symbolic manipulation;
c) applications of scientific and/or technological systems.

Its importance for *applications* has grown in recent years with the introduction of technological areas (related to information processing, software engineering, etc.) in which the symbolic nature of the objects studied makes the techniques of calculus and numerical analysis inapplicable. For these areas, algebra provides both a theoretical framework for the development of theories and algorithmic techniques for the concrete manipulation of objects. Traditional areas of science such as physics, chemistry and biology has found the method of computer algebra creeping in. For more information , see

- Computer algebra in science and engineering /editors, J. Fleischer, 1995.
- Computer algebra in industry 1,2: problem solving in practice 1991, 1995.

- Symbolic computation in undergraduate mathematics education / Zaven A. Karian, editor. Mathematical Association of America, 1992. Computer algebra systems in the classroom / edited by J. Monaghan and T.A. Etchells., 1993.

It is important to stress that the *mathematical* theories to which computer algebra applies are not necessarily only the algebraic ones: polynomial and differential equations, commutative algebra, real geometry, and group theory have a well-established research activity using symbolic computation techniques, and such activity necessarily *interweaves* mathematics, complexity theory, and software systems. For example, in the last few years extensive studies have been devoted to complexity in elimination theory, and to the importance of appropriate data structures for exhibiting efficient algorithms to solve polynomial systems.

Computer Algebra System General purpose computer algebra systems: MuPAD (free of charge), Maple(TM), Mathematica(TM), Macsyma(TM), Axiom(TM), Reduce(TM), Derive(TM).

Special purpose computer algebra systems (free of charge):
Computational Group Theory: GAP
Computational (Algebraic) Number Theory: PARI, KANT, Asir, SACLIB
Computational Algebraic Geometry: Casa, Gröbner, Macaulay
Mixed:Magma(TM)

The first beginnings of the development of program systems for computer algebar date back to the 1950s, but . . . . many of these programs can now run on my cheap Pentium-133. One can compute for instance integral bases, unit group generators and the class group structure of algebraic number fields, and even generators for the Mordell-Weil group.

SAC meetings ISSAC '9x International symposium on symbolic and algebraic computation (the primary international conference on SAC, the first meeting was on 1966 but with a different name—SYMSAC.)
ANTS Algorithmic number theory: international symposium, 199x.
MEGA '9x Effective Methods in algebraic geometry.
AAECC Applied Algebra, Algorithms and Error-Correcting Codes '198x '199x
DISCO '9x, Design and implementation of symbolic computation systems: International Symposium,
PASCO '9x International Symposium on Parallel Symbolic Computation.
AISMC Integrating symbolic mathematical computation and artificial intelligence : International Conference, 199x

General SAC Meetings in the East ASCM '95 Beijing, China. The First Asian Symposium on Computer Mathematics
ATCM '95 Singapore The First Asian Technology Conference in Mathematics
ASCM '96 Kobe, Japan The 2nd Asian Symposium on Computer Mathematics
ATCM '97 Penang, Malaysia The 2nd Asian Technology Conference in Mathematics on COMPUTER TECHNOLOGY IN MATHEMATICAL RESEARCH AND TEACHING

Periodical publications Journal of Symbolic Computation, editor B.F. Caviness (formerly B. Buchberger)
Applicable Algebra in Engineering, Communication, and Computing, editor J. Calmet
SIGSAM Bulletin (ACM special interest group on SAC)

Something more should be said from the angle of mathematics.

Computers have stretched the limits of what is possible in mathematics. More than that, they have given rise to new fields of mathematical study: the analysis of new and traditional algorithms; the creation of new paradigms for implementing computational methods; and the viewing of old techniques from a concrete algorithmic vantage point.

Being suspected of the validity of this statement, Tang Simon asked the following question in the USENET newsgroup **sci.math.symbolic** from April to August 1995:

> A Computer has nothing to do with Pure Mathematics. Discuss (100%) Not for homework and examination, simply because students are bewildered.

Here are some fragments of answers that I have collected:

On the style (school or philosophy) of Mathematics The word "algorithm" as well as the key word "algebra" in the title of this book come from the name and the work of the ninth-century scientist Mohammed ibn Musa al-Khowarizml, ... The word "algorithm" is actually a westernization of al-Khowarizml's name, while algebra derives from "al-jabr," a term that appears in the title of his book Kitab al-jabr wa'l muqabala, where he discusses symbolic methods for the solution of equations.

This *-close-* connection between algebra and algorithms !!!–lasted roughly up to the beginning of this century–!!!; until then, the primary goal of algebra was the design of *constructive* methods for solving equations by means of symbolic transformations.

During the !!!–second half of the nineteenth century–!!!, a new line of thought began to enter algebra from the realm of geometry, where it had been successful since Euclid's time, namely, the *—axiomatic method—*.

The starting point of the axiomatic approach to algebra is the question, What kind of object is a symbolic solution to an algebraic equation? To use a simple example, the question would be not only, What is a solution of ax+b = 0, but also, What are the properties of the objects a and b that allow us to form the object -b/a? The axiomatic point of view is that these are objects in a surrounding algebraic structure which determines their behavior. The algebraic structure in turn is described and determined by properties that are laid down in a set of axioms.

The foundations of this approach were laid by Richard Dedekind, Ernst Steinitz, David Hilbert, Emmy Noether, and many others. The *–axiomatic method–* favors *-abstract-*, **-non-constructive-** arguments over *-concrete-* algorithmic constructions. The former tend to be considerably *-shorter-* and more *-elegant-* than the latter.

—Before the arrival of computers—, this advantage more or less *-settled-* the question of *-which-* one of the two approaches was to be preferred: the algorithmic results of mathematicians like Leopold Kronecker and Paul Gordan were way beyond the scope of what could be done with pencil and paper, and so they had little to offer except being more tedious than their non-constructive counterparts.

On the other hand, it would be a *–mistake–* to construe the axiomatic and the algorithmic method as being irreconcilably *–opposed–* to each other. As a matter of fact, significant *-algorithmical-* results in algebra were proved by the very proponents of *-axiomatic-* thinking such as David Hilbert and Emmy Noether. Moreover, mathematical logic–a field that centers around the *-axiomatic-* method–made fundamental contributions to *-algorithmic-* mathematics in the 1930s. Alan Turing and Alonzo Church for the first time made precise the concept of computability in what is known as Church's thesis, or also as the Church-Turing thesis. Kurt Godel proved that certain problems inherently elude computability and decidability. This triggered a wave of new results by Alfred Tarski and other members of the Polish school of logicians on the algorithmic solvability or unsolvability of algebraic problem.

Again, because of their enormous complexity, these algorithms were of *–no practical–* significance whatsoever.

As a result, the !!–beginning second half of this century–!! saw an *—axiomatic and largely non-constructive—* approach to algebra firmly established in both research and teaching.

———The arrival of computers——— * * and their breathtaking development in the !!–last three decades–!! then prompted a *–renewed interest–* in the problem of **–effective constructions–** in algebra. Many constructive results from the past were unearthed, often after having been rediscovered independently.

Moreover the development of new concepts and results in the area has now established *–computer algebra–* as an independent discipline that extends **-deeply-** into *–both–* mathematics and computer science.

[oh! Pure Maths]

There are many good reasons for viewing computer algebra as an independent field. However, the fact that the *–mathematical part–* of it is some what separated from the work of *–pure algebraists–* is, in our opinion, rather unfortunate and not at all justified. We feel that this situation **–must and will change–** in the near future.

As a matter of fact, **–computational aspects–** are **–beginning–** to show up more and more in undergraduate-level textbook on abstract algebra. * *

[oh! Take a break.]

A case study: Gröbner bases There is, however, one particular contribution made by computational algebra that is in most dire need of being introduced in the *–mathematical mainstream–* namely, the theory of Grobner bases.

Grobner bases were introduced by Bruno Buchberger in 1965. The terminology acknowledges the influence of Wolfgang Grobner on Buchberger's work. .... Given a finite set of multivariate polynomials over a field, the Buchberger algorithm computes a new set of polynomials, called a Grobner basis, which generates the same ideal as the original one and is an analogue to the gcd of the univariate case ....

It leads to solutions to a large number of algorithmic problems that are related to polynomials in several variables. Most notably, algorithms that involve Grobner basis computation allow **–exact–** conclusions on the solutions of systems of non-linear equations, such as the (geometric) dimension of the solution set, the exact number of solutions in case there are finitely many, and their actual computation with arbitrary precision.

Most of the problems for which Grobner bases provide algorithmic solutions were already known to be solvable *-in principle-*. Grobner bases are a *-giant step-* forward insofar as actual implementations have *-become feasible-* and have actually provided answers to physicists and engineers.

[ie Pure Maths + Computer may–¿ applications of DEEP Mathematics. Is this a happy event?]

On the other hand, many problems of no more than moderate input size still defy computation. The *—mathematics behind the algorithms—*, as well as the *—hardware—* that performs them have a **-long way to go–** before these problems can be considered solved to the satisfaction of the user.

[ie full of opportunities.]

The purpose of this book is to give a self-contained, **mathematically sound** introduction to the theory of Grobner bases and to some of its applications, stressing both theoretical and computational aspects.

..... Chapter 6-10 cover a wide range of applications, ... Strong emphasis is placed on a mathematically sound verification of the algorithms.

[ie they are not careless or non-rigorous mathematics. Are they ugly?]

..:::: implementations of any practical value involve considerably more mathematics.

.... theoretically and algorithmically. .... absence of any complexity theory ...... On the contrary, we feel that complexity theory is too important an issue to be dealt with lightly.

{oh! Pure Maths & Computer Science}

... Furthermore, we demonstrate how Grobner bases can often be used to give *–elegant an enlightening proofs of classical results–*, ... This shows that Grobner bases are not only a powerful tool for actual computation, but also a cornerstone of commutative algebra.

[oh! an extra 'double oh'.]

SAC from the angle of Mathematics The mathematical uses of computers can be divided roughly into numeric and nonnumeric applications.

Numeric computation involves primarily calculations in which real numbers are approximated by elements from a fixed set of rational numbers, called floating-point numbers. Such computation is usually associated with the mathematical discipline NUMERICAL ANALYSIS.

One nonnumeric application of computers to mathematics is

_S Y M B O L I C_COMPUTATION_

Although it is impossible to give a precise definition, symbolic computation normally involves representing mathematical objects _exactly_ and perform _exact_ calculations with these representations. It includes efforts to automate many of the techniques taught to high school students and college undergraduates.

The term 'computer algebra' is frequently used as a syndnym for 'symbolic computation'. Although the term 'computer algebra' is well established, it conflicts somewhat with current usage within mathematics, where 'algebra' usually is used in the narrower sense of 'abstract algebra', the study of algebraic structures such as groups, rings, fields, and modules.

The word 'computer' in the phrase 'computer algebra' is also not quite accurate. It is true that much of what is done is motivated by the existence of computers.

Nevertheless, the algebraic algorithms which have been developed represent substantial _mathematical_ achievements, whose importance is not dependent entirely on their being incorporated into computer programs.

Within symbolic computation there is a rapidly expanding area of computational (abstract) algebra, which is the study of procedures for manipulating objects from abstract algebra with particular concern for practicality.
_Computational_Group_Theory_ is the part of **computational algebra** which considers problems related to groups. Other flourishing subfields of computational algebra are _Computational_(Algebraic)_Number_Theory, and _Computational_Algebraic_Geometry.

Symbolic computation is at the _border_ between mathematics and computer science. The objects being manipulated are _mathematical_. However, the algorithmic ideas often have come from _computer_science_, and individuals who identify themselves as computer scientists have made important contributions to the subject.

A point of continuing debate is the role of _complexity_theory_ in symbolic computation. The traditional complexity measure in theoretical computer science is asymptotic worst-case complexity. For users of symbolic software, worst-case analysis are often too pessimistic. Of much more relevance is average-case complexity. However, average-case analyses are lacking for many of the most important algebraic algorithms. Moreover, there are cases in which no agreement has been reached on what the average case is. [....] Frequently, all that we can do is apply competing methods to a selection of test problems and compare the results. Experimental evidence is better than nothing, but one must be very _careful_ about drawing conclusions from such evidence.

[....]

While the work of symbolic solutions for algebraic equations, from integrals to polynomials, has its roots in work of the Greeks (the Euclidean algorithm continues to be useful in computing the greatest common divisor in various domains), research began in earnest _thirty_years_ago_, with the development of Macsyma(TM) for solving polynomial and integral equations. Since that time a variety of system have appeared.

Much work in the development of algorithms remains to be done, from a very _practical_ systems development to _theoretical_ algorithms research (Mathematics?).

On the other hand, computer algebra systems have been a _useful_tool_ to teachers, scientists and engineers, from (pure) mathematical research to industry and business.

An article that looks like a conclusion Pure mathematics has resisted computers longer than most branches of science. The advent of computers is finally, fundamentally transforming the practice of mathematics. Both pure and applied mathematics and mathematicians are experiencing the upheaval. The changes include:

* The line between pure and applied mathematics is in flux.
* Computers spur new areas of research and revive languishing areas.
* Computers increase productivity or make productivity possible by providing new means for: generating examples and testing conjectures; recording and disseminating information; conducting joint research among far removed researchers.
* Algorithm development motivates theory.
* Computers enhance teaching by providing: vastly improved graphical presentation; ability to present non-trivial examples in class; ability to routinely assign non-trivial homework problems; tools for students to analyze, experiment and play with the subject matter.

Areas of mathematics such as applied logic, algebraic geometry, combinatorics, commutative algebra, dynamical systems, and many others owe their existence, *–rebirth–* or major new research initiatives to the influence of computers. Researchers in these areas use computers as an extension of hand calculation to compute examples which test conjectures and provide data to increase understanding and motivate theorems.

Beyond computation, computers exert a profound influence on mathematical research. Researchers delve into the algorithms underlying computation. They find motivation for pure mathematics from analyzing existing algorithms and developing new algorithms. Significant developments in the creation of *–algorithms–* require significant developments in *–theory–*. This is a fertile source of new mathematical theory.

Buchberger theory, other computational methods, and the related theory are becoming known to commutative algebraists and algebraic geometers because of their importance for computation and because they provide the basis for many new interesting problems. (Originally Galois theory was primarily computational. Now a days, proofs often make use of the fact that a Galois group exists and no computation is involved. The same can be done with Buchberger theory.)

The *–computer algebra–* community realized the significance of this work before the general academic pure mathematics community. The computer algebra community involves a full spectrum from engineers to theoreticians. The problem solvers using computer algebras, the implementors, the algorithm developers and the theory developers work side by side, motivating and benefiting from each other's developments and problems.

To be Continued ....Continued from above .....

At best, the result is a robust, stimulating, productive scientific environment.

****__**** At worst, over zealous theoreticians -Puritans- and over zealous algorithm developers -Algos- divide into narrow-minded, self-serving communities. Each thinks it is smarter and works harder than the other. The Algos believe that algorithmic solutions to problems require better theorems with more difficult proofs than non-algorithmic solutions. The Puritans believe that the Algos work on problems which have *–already been–* solved. ***__****

Purely existential results are fundamental to algorithm development. For in computer algebra systems, one wishes to use efficient algorithms as measured by the number of computational steps and memory usage. Determining bounds on algorithms and finding efficient algorithms requires and motivates further theory. This is *–why–* algorithmic approaches to problems can be more difficult and involve more theory than non-algorithmic approaches.

Suppose a new area of mathematics has just opened. Here is a list of new problems and their difficulties.

Problem:	*A*	*B*	*C*	*D*	*E*	*F*
Difficulty of non-algorithm approach:	5	10	15	20	25	30
Difficulty of algorithm approach:	6	12	18	24	30	36

At first the Puritans and Algos work on problem A. An existential solution is obtained before an algorithmic solution. Soon the Puritans are working on Problem F, while the Algos are working on Problem E. This is *why* the Puritans fell the Algos are working on problems which have already been solved. The Algos have contempt for problem F as being of only theoretical interest and irrelevant to the real world. The first Algos developing algorithmic approaches to F face this contempt from their Algo peers. Once F becomes established on an algorithmic basis, the Algos/Puritans act as if they always believed it had real world relevance.

Applications *–made possible–* by computers are part of the impetus for the study and development of modern commutative algebra and algebraic geometry. The book presents *–applications–* in chapters on Robotics, Invariant theory of Finite Groups and Automatic Geometric Theorem Proving. Automatic Geometric Theorem Proving is not an artificial intelligence attempt to duplicate the human theorem proving process. Instead, using analytic geometry, the theorem to be proved is reduced to a system of equations, possibly with some exceptional conditions or inequalities. One then uses algebraic means to verify the resulting algebraic system.

Bibliography

[Poh87] M. Pohst (ed.), *Algorithmic methods in algebra and number theory*, Academic Press, 1987, Volume 4, number 1 of Journal of Symbolic Computation.

[PZ89] M. Pohst, H. Zassenhaus, **Algorithmic algebraic number theory**, Cambridge University Press, 1989.

[Zas71] H. Zassenhaus, *On the group of an equation*, In Computers in Algebra and Number theory, G. Birkhoff and M. Hall, eds., SIAM and AMS Proceedings, 1971, pp.69-88.

[Coh93] H. Cohen, **A course in computational algebraic number theory**, Springer-Verlag, 1993.

[GAP] Martin Schönert et al., GAP 3.4, patchlevel 3, *Groups, Algorithms and Programming*, [Computational Group Theory Software System], Lehrstuhl D für Mathematik, Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, 1995. (gap@samson.math.rwth-aachen.de)

[MAPLE] B. W. Char, K.O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan and S. M. Watt, *Maple Library Reference Manual*, [General Purpose Computer Algebra System] Springer-Verlag, 1991.

[GaloisBks] (1) E. Artin, **Galois theory**, second edition, University of Notre Dame Press, 1944.
(2) I. Kaplansky, **Fields and rings**, second edition, The University of Chicago Press, 1972.
(3) J. Rotman, **Galois theory**, Springer-Verlag, 1990.
(4) B.L. van der Waerden, **Modern Algebra**, Vol.1.(transl. by F. Blum), Ugar, New York, 1953.
(5) N. Jacobson, **Basic algebra I**, second edition, W.H. Freeman and Company, 1985.
(6) S. Lang, **Algebra**, Addison Wesley, 1971.
(7) L. Gaal, **Classical Galois theory with examples**, 4th edition, Chelsea, 1988.
(8) H. M. Edwards, **Galois theory**, Springer, 1984. (Old-fashioned Galois Theory)
(9) J.-P. Tignol, **Galois theory of algebraic equations**, Wiley, 1988. (Old-fashioned Galois Theory)

[Wie64] H. Wielandt, **Finite Permutation Groups**, Academic Press, New York, 1964.

[Sta69] R. P. Stauduhar, *The automatic determination of Galois groups*, Ph.D. Dissertation, University of California, Berkeley, 1969.

[Sta73] R. P. Stauduhar, *The determination of Galois Groups*, Mathematics of Computation 27 (1973), 981-996.

[Mck79] John McKay, *Some remarks on computing Galois groups*, SIAM Journal of Computing, vol.8, no.3, 1979, 344-347.

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

[MR85] J. McKay and E. Regener, *Actions of permutation groups on r-sets*, Communications in Algebra 13 (1985), 619-630.

[Soi81] L. Soicher, *The computation of Galois groups*, M.Sc. Comp. Sci. Thesis, University of Concordia, Montréal, Québec, Canada, 1981.

[Soi84] L. Soicher, *An algorithm for computing Galois groups*, in Computational Group Theory (M. D. Atkinson, Ed.), pp. 291-296, Academic Press, 1984.

[SM85] L. Soicher and J. Mckay, *Computing Galois groups over the rationals*, Journal of Number Theory 20 (1985), 273-281.

[EFM79] D. W. Erbach, J. Fischer, and J. McKay, *Polynomials with $PSL(2,7)$ as Galois group*, Journal of Number theory 11, 1979, pp.69-75.

[LO77] J. C. Lagarias and A. M. Odlyzko, *Effective versions of the Chebotarev density theorem*, in Algebraic Number fields (L-functions and Galois properties), A. Frohlich, Ed., pp.409-464, Academic Press, London, 1977.

[Sha54] I. R. Shafarevich, *Construction of fields of algebraic numbers with given solvable Galois group*, Izv. Akad. Nauk USSR Ser. Mat. 18, 1954, pp.525-578.

[CAbks] (1) F. Winkler, Polynomial algorithms for computer algebra, Springer-Verlag, 1996.
(2) R. Zippel, Effective polynomial computation, Kluwer Academic Publishers Group, 1993.
(3) B. Mishra, Algorithmic algebra, Texts and Monographs in Computer Science, Springer-Verlag, 1993.
(4) K.O. Geddes, S.R. Czapor, G. Labahn, Algorithms for computer algebra, Kluwer Academic Publishers Group, 1993.
(5) M. Mignotte, Mathematics for computer algebra, Springer-Verlag, 1992.
(6) J. H. Davenport, Y. Siret and E. Tournier, Computer algebra, Systems and algorithms for algebraic computation, Second edition, Academic Press, 1993.
(7) A.G. Akritas, Elements of computer algebra with applications, John Wiley & Sons, Inc., 1989.
(8) T. Becker, V. Weispfenning and H. Kredel, "Gröbner Bases, A computational approach to commutative algebra", Graduate Texts in Mathematics 141, Springer-Verlag, 1993.
(9) D. Cox, J. Little, D. O'Shea, "Ideals, Varieties, and Algorithms, An introduction to computational algebraic geometry and commutative algebra", second edition, Springer undergraduate texts in Mathematics, Springer-Verlag, 1997.
(10) R. Lidl and H. Niederreiter, Introduction to Finite Fields and their Applications, Cambridge University Press, 1986.

[Lan87] Susan Landau, *Factoring polynomials quickly*, Notices of the American Mathematical Society, [Special Article Series], Vol. 34, No. 1 (1987), pp. 3-8.

[Ber68] E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.
See also E. R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Technical Journal, vol.46, 1967, pp.1853-1859.

[Zas69] H. Zassenhaus, *On Hensel factorization I*, Journal of Number Theory 1 (1969), 291-311.

[Tra76] B. M. Trager, *Algebraic factoring and rational function integration*, In Proceedings of the 1976 Symposium on Symbolic and Algebraic Computation, ACM Press, 1976, pp.219-226.

[LLL82] A. K. Lenstra, H.W. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen 261, 1982, pp.515-534.

The End