Algorithms for Galois Theory

Ben Kuehnert
Advised by Amanda Tucker

May 14, 2021

1 Introduction

Galois theory is an essential tool in algebra — it is famously used to prove that you can't trisect an angle or double a cube, to prove that there is no formula for the roots of a quintic equation under radicals, and even that Fermat's Last Theorem holds. The central object in the subject, the Galois group, is difficult to compute by hand, and exercises to compute it are ubiquitous in undergraduate algebra courses. Hence, it is important to have an effective method to solve this problem.

2 Preliminaries

In the context of this paper, Galois groups will always be defined over $\mathbb{Q}$, shown below.

Definition 2.1 (Galois group of a polynomial) Let $f(x) \in \mathbb{Z}[x]$. Let K be the splitting field of f. Then, the Galois group of f is

$$G = \{\sigma \in \text{Aut}(K) : \forall x \in \mathbb{Q}, \sigma(x) = x\}$$

where $\text{Aut}(K)$ denotes the group of automorphisms of K. As our eventual goal is to actually compute Galois groups, it is important to have a simple way to represent them. The following theorem helps with this.

Theorem 2.1 Let f be monic and irreducible. Let $X = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be the roots of f, which lie in some fixed algebraic closure of $\mathbb{Q}$. Let G be the

Galois group of f. Then, G acts freely and transitively on X via the action

$$(\sigma, \alpha) \mapsto \sigma(\alpha).$$

where $\sigma \in G$ and $\alpha \in X$.

This means that G is isomorphic to a transitive subgroup of $\mathrm{Sym}(X)$. Each element $\sigma \in G$ can be identified by where it takes each root of f.

Moving forward, $f(x) \in \mathbb{Z}[x]$ will be assumed to be square-free, which is a simpler class of polynomials. We can make this assumption because we have a polynomial-time algorithm to compute the square-free factorization of a polynomial f, and this factorization necessarily has the same roots as f, and thus the same Galois group. To do this, note that α is a multiple root of f if and only if it is also a root of f'. Thus, if we divide f by the greatest common divisor of f with f', denoted (f, f'), our result will be the square-free factorization of f. Computing derivatives of polynomials and dividing them are easy, so the only roadblock is computing (f, f'). This is done with the "Sub-resultant Algorithm" and is described on page 122 of [2].

The algorithm to compute the Galois group of f is done in three steps. The first step is to factor f into a product of irreducibles, so $f = f_1 \cdot f_2 \cdots f_l$ where each f_i is irreducible over $\mathbb{Q}$. The next step is to compute the Galois group G_i of each f_i. The final step is to use the G_i to compute the Galois group G of f.

3 Root-finding algorithms

Algorithms to compute the roots of a polynomial will be used as a subroutine in the Galois group computation.

3.1 Over the integers

In this section, we will describe the method for finding integer roots for monic polynomials in $\mathbb{Z}[x]$. This problem can be reduced to integer factorization via the following theorem:

Theorem 3.1 (Rational Root Theorem) Let $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ have $a_i \in \mathbb{Z}$, $a_0, a_n \neq 0$. Then, if $p/q \in \mathbb{Q}$ satisfies $f(p/q) = 0$, then p divides a_0 and q divides a_n.

An immediate corollary is that all integer roots must divide a_0. So, to find all integer roots, just compute all factors of a_0 and test each for being a root. Algorithms such as the Sieve of Eratosthenes exist to find factors of a number, although in general integer factorization is hard for large numbers. However, we expect polynomials to have small coefficients so this is not an issue.

3.2 Over the complex numbers

Finding roots in general is more complicated. For this, we will use Newton's method. This process works by iteratively improving an initial guess for a root, until the root is within the desired accuracy.

The process will be described recursively: Let f be a a square-free polynomial in $\mathbb{Z}$. Let x_0 be an initial "guess". It does not necessarily need to be a good guess, and in fact it could be almost any complex number and the process will still converge. Now the recursive step: suppose we have some guess x_n. Then, we can improve this to a guess x_{n+1} via

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

The convergence of this method is due to Taylor's theorem. Let x_n be the approximation obtained at the n-th step of the process, and define $\epsilon_n = |\alpha - x_n|$ where α is the root of f that we are approximating. Taylor expanding about α gives

$$0 = f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(\xi_n)(\alpha - x_n)^2$$

for some $\xi_n \in I$ where I is the interval between x_n and α. Assuming $f'(x_n) \neq 0$ (which is usually OK to do since f is square-free – it doesn't have multiple roots, hence $f'(\alpha) \neq 0$) we have:

$$\frac{f(x_n)}{f'(x_n)} + (\alpha - x_n) = \frac{-f''(\xi_n)}{2f'(x_n)}(\alpha - x_n)^2.$$

Then, since $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, we have

$$\alpha - x_{n+1} = \frac{-f''(\xi_n)}{2f'(x_n)}(\alpha - x_n)^2$$

taking an absolute value gives

$$\epsilon_{n+1} = \frac{|f''(\xi_n)|}{2|f'(x_n)|} \cdot \epsilon_n^2.$$

which shows that the error term is strictly decreasing if

$$\sup_{\xi_n \in I} \frac{1}{2} \left| \frac{f''(\xi_n)}{f'(x_n)} \right| < 1$$

which occurs in the vast majority of cases. To find other roots of f, compute $\widetilde{f(x)} = f(x)/(x - \widetilde{\alpha})$ where $\widetilde{\alpha}$ is obtained approximate root, and then re-run the algorithm on this new function. Care must be taken to not propagate error, though. One way to avoid this is to run the iterative step with f rather that $\widetilde{f}$ for subsequent roots at the end, to ensure that the acquired roots are indeed approximate roots of f. Pseudocode for this algorithm, along with some parameters and extra steps tuned for real world examples can be found in [2].

4 Factoring polynomials over $\mathbb{Z}$

The first step in our algorithm is to factor $f(x) \in \mathbb{Z}[x]$ into a product of irreducibles over $\mathbb{Z}$. In 1982, A. Lenstra, H. Lenstra, and L. Lovász devised the LLL algorithm, which solved this problem in polynomial time [6]. However, for our purposes we will be using a slightly different LLL-based algorithm. This algorithm proceeds by computing the minimal polynomial of an approximation of an algebraic integer α, described in [5]. Paired with the above algorithm for computing approximate roots of a polynomial, this turns into a polynomial-time algorithm for factorizing a polynomial.

To describe this algorithm, we must first define the LLL algorithm.

4.1 The LLL algorithm

Before giving the LLL algorithm, a few definitions are necessary:

Definition 4.1 Let $b_1, \ldots, b_k$ be linearly independent vectors in $\mathbb{R}^n$. The *lattice generated by* $b_1, \ldots, b_k$ is the set

$$L = \mathcal{L}(b_1, \ldots, b_k) = \left\{ \sum_{i=1}^{k} \lambda_i b_i : \lambda_i \in \mathbb{Z} \right\}$$

we call $b_1, \ldots, b_k$ a *basis* for L.

Note that in general there are multiple bases for a particular lattice L. Next,

Definition 4.2 Let $b_1, \ldots, b_k$ be a basis, let $b_1^*, \ldots, b_k^*$ be the Gram-Schmidt orthogonolization of the basis. We call $b_1, \ldots, b_k$ *LLL reduced* if

- For all $1 \leq j < i \leq n$,
$$\left| \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right| \leq \frac{1}{2}$$
- For $i = 2, 3, \ldots, n$
$$\frac{3}{4} \|b_{i-1}^*\|^2 \leq \|b_k^*\|^2 + \left| \frac{\langle b_k, b_{k-1}^* \rangle}{\langle b_{k-1}^*, b_{k-1}^* \rangle} \right| \cdot \|b_{k-1}^*\|^2$$

The goal of the LLL algorithm is to produce short vectors for a lattice. The problem of finding the shortest vector in a lattice is known as the Shortest Vector Problem (SVP), and is known to be intractable in general. The idea behind LLL is to find a basis for the lattice that is close to orthogonal. This is because the shortest vector in an orthogonal basis is the shortest vector in the lattice. Intuitively, the shortest vector in a nearly orthogonal basis is nearly the shortest vector in the lattice. This is formalized in the following theorem.

Theorem 4.1 Suppose $b_1, \ldots, b_k \in \mathbb{R}^n$ is an LLL reduced basis for a lattice L. Then,
$$\|b_1\|^2 \leq 2^{(n-1)} \cdot \lambda(L)^2$$
where $\lambda(L)$ is the length of the shortest vector in L.

Proof. Proposition 1.11 of [6] □

Finally, we can define the LLL algorithm:

Theorem 4.2 Let $b_1, \ldots, b_k \in \mathbb{R}^n$ be a basis for a lattice L. Then, there is an algorithm which terminates and produces an LLL-reduced basis $v_1, \ldots, v_k$ for L in $O(n \cdot k^3 \cdot \log B)$ arithmetic operations where $B = \max\{\|b1\|, \ldots, \|b_k\|\}$.

Proof. Proposition 1.26 of [6] □

4.2 Computing minimal polynomials

Now, we can describe the minimal polynomial algorithm. The basic idea is to take some algebraic integer α, and compute an integral relation among $1, \alpha, \alpha^2, \ldots, \alpha^d$ for $0 \leq d \leq n$. If d is minimal such that there is an integral relation $c_0+c_1\alpha+c_2\alpha^2+\cdots c_d\alpha^d = 0$, then the polynomial $h(x) = \sum_{i=0}^{d} c_i x^i$ is the minimal polynomial for α. In practice, however, we do not have access to α^i to full precision. So, moving forward, $\overline{\alpha_i}$ will represent an approximation of α^i. If $f(x) = \sum_{i=0}^{n} c_i x^i$, then $f_{\overline{\alpha}} = \sum_{i=0}^{n} c_i \overline{\alpha_i}$.

Let $f(x) \in \mathbb{Z}[x]$ have degree n. Let α be a root of f. Define L_s as $\mathcal{L}(b_1, \ldots, b_{n+1}$ where the b_i are the rows of the following matrix:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 2^s\Re(\overline{\alpha_0}) & 2^s\Im(\overline{\alpha_0}) \\ 0 & 1 & 0 & \cdots & 0 & 2^s\Re(\overline{\alpha_1}) & 2^s\Im(\overline{\alpha_1}) \\ 0 & 0 & 1 & \cdots & 0 & 2^s\Re(\overline{\alpha_2}) & 2^s\Im(\overline{\alpha_2}) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 2^s\Re(\overline{\alpha_n}) & 2^s\Im(\overline{\alpha_n}) \end{bmatrix}$$

Then, $\mathbb{Z}[x]$ and L_s are in one-to-one correspondence via

$$g(x) = \sum_{i=0}^{n} a_i x^i \mapsto \sum_{i=0}^{n} a_i b_i = \widetilde{g} \in L_s$$

The variable s is an integer parameter that will be used to guarantee that the resulting short vector is indeed the minimal polynomial of α. This is done by the following theorem:

Theorem 4.3 Let α be an algebraic integer, and $h(x)$ be its minimal polynomial. Let H be the maximum among absolute values of h's coefficients, and d be the degree of h. Then, if s is the smallest integer such that

$$2^s \geq 2^{d^2/2} \cdot (d+1)^{(3d+4)/2} \cdot H^{2d}$$

and if $|\alpha^i - \overline{\alpha_i}| \leq 2^{-s}$ for $1 \leq i \leq d$. Then, for any $g \neq h$ with degree n or less, we have

$$\|\widetilde{g}\|^2 > 2^n \|\widetilde{h}\|^2$$

where $\widetilde{g}$ and $\widetilde{h}$ are the images of g and h in L_s.

Proof. Lemma 1.9 of [5] □

This gives us the following algorithm. Suppose s and the $\overline{\alpha_i}$ satisfy the above conditions (H and d can be replaced with the height and degree of f, respectively). Run LLL on the lattice L_s, and let $\widetilde{v}$ be the first vector in the resulting LLL-reduced basis. Then,

$$\|\widetilde{v}\|^2 \leq 2^n \|\widetilde{h}\|^2$$

by theorem 4.1. Now, suppose $v(x)$ is not equal to the minimal polynomial of α, $h(x)$. Then, by theorem 4.3

$$\|\widetilde{v}\|^2 > 2^n \|\widetilde{h}\|^2$$

which contradicts the above inequality. Thus, $v(x) = h(x)$.

5 Galois groups of irreducible polynomials

Now that we can factor f into a product of irreducibles, we give an algorithm to compute the Galois group for each irreducible factor.

5.1 Invariants

In a previous section, we showed that for an irreducible monic polynomial $f \in \mathbb{Z}[x]$, its Galois group G is isomorphic to a transitive subgroup of S_n where $n = \deg(f)$. The task of the algorithm will be to look through the transitive subgroups in an efficient way, and throw out subgroups which are not isomorphic to G. This theory of invariants will give us tools to do that. Define $\mathcal{R} = \mathbb{Z}[x_1, \ldots, x_n]$. Then, $\sigma \in S_n$ acts on $h \in \mathcal{R}$ via

$$(\sigma, h) = h(\sigma(x_1), \ldots, \sigma(x_n))$$

when σ is considered as a permutation of the x_i.

Definition 5.1 (Invariant) $h \in \mathcal{R}$ is an *invariant for* $U \subseteq S_n$ if

$$(\sigma, h) = h \qquad \forall \sigma \in U$$

If $V \subseteq U$ and h is an invariant for V but not invariant for U, then h is called a *U-relative invariant for V*

This leads into the main theorem for invariants:

Theorem 5.1 Let $V \subsetneq U \subseteq S_n$ where V is a maximal subgroup of U. Let $h \in \mathcal{R}$ be an U-relative invariant for V. Then, if $W \subseteq U$ then

$$W \subseteq V \iff h \text{ is an invariant for } W.$$

Proof. If $W \subseteq V$, then clearly h is an invariant for W as h is fixed by every $\sigma \in V$, hence it is fixed by all $\sigma \in W$.

Next, suppose that h is an invariant for W. Suppose there is some $\sigma \in W$ which is not in V. Consider the group $X = \langle V, \sigma \rangle$. Since V is a maximal subgroup of U, then X must contain U as a subgroup. Let $\tau \in X$. Then, τ is some composition of elements in V and σ, all of which fix h. Thus, $(\tau, h) = h$. Hence, h is an invariant for X. Thus, U is invariant for h as it is a subgroup of X. This is contradicts the hypothesis of h being a U-relative invariant for V. Thus, each element of W must lie in V, hence $W \subseteq V$. □

5.2 Resolvent descent method

This section closely follows [4], which gives a general algorithm for computing Galois groups. Throughout this section, $f \in \mathbb{Z}[x]$ is monic and irreducible and G is the Galois group of f. Define $\varphi : \mathcal{R} \to K$ via $h \mapsto h(\alpha_1, \ldots, \alpha_n)$ where K is the splitting field of f, and the α_i are the roots of f lying in K. Note that φ is a homomorphism. Specifically,

Lemma 5.2 Let $h \in \mathcal{R}, \sigma \in G$. Then,

$$\varphi((\sigma, h)) = (\sigma, \varphi(h))$$

Proof. This is clear since multiplication and addition are commutative in $\mathbb{C}$. □

This homomorphism will be important in deciding if a given element in $\mathcal{R}$ is invariant for G.

Lemma 5.3 If

$$\varphi((\sigma, h)) \neq \varphi(h) \qquad \text{for all } \sigma \in S_n \setminus \{\text{id}\}$$

then h is G-invariant if and only if $\varphi(h) \in \mathbb{Z}$.

Proof. Suppose h is G invariant. So,

$$\varphi(h) = \varphi((\sigma, h)) = (\sigma, \varphi(h))$$

for each $\sigma \in G$. Since $\varphi(h)$ is fixed by all Galois automorphisms, it lies in the base field, which is $\mathbb{Q}$. Yet, $\varphi(h)$ is an polynomial in α_i, which are algebraic integers. Thus, it lies in the intersection of algebraic integers and $\mathbb{Q}$, which is $\mathbb{Z}$.

Conversely, suppose $\varphi(h) \in \mathbb{Z}$, and suppose h is not G invariant. Then, there exists some $\sigma \in G$ such that $(\sigma, h) \neq h$. By the assumption, $\varphi(h) \neq \varphi((\sigma, h))$. Yet,

$$\varphi((\sigma, h)) = (\sigma, \varphi(h)) = \varphi(h)$$

which is a contradiction. The first equality is by lemma 5.2. The second equality is since $\varphi(h) \in \mathbb{Z}$, hence is fixed by any $\sigma \in G$. □

As mentioned in [4], for each h, there exists an $\hat{f}$ such that this property holds and the splitting field of $\hat{f}$ is K.

Next, we form the lattice of transitive subgroups of S_n. Forming this lattice is extremely computationally intensive, as the order of S_n is $n!$. At this point, we could proceed by computing a U-relative V invariant h for each pair (U, V) where V is a maximal subgroup of U, and test for containment of G by checking if $\varphi(h) \in \mathbb{Z}$. However, this is unnecessarily slow, as we end up considering conjugate subgroups multiple times. The following polynomial will be useful in allowing us to check for containment of G for all conjugates at once.

Definition 5.2 Let $V \subsetneq U \subseteq S_n$ be groups where V is a maximal subgroup of U. Let h be a U-relative invariant for V. The *relative resolvent* for V is defined as

$$R(x) = \prod_{\sigma \in U} (x - \varphi((\sigma, h)))$$

Then,

Theorem 5.4 Let V be a maximal subgroup of U, with $G \subseteq U$. Let $R(x)$ be the relative resolvent of V. Then,

(1) $R(x)$ has integer coefficients.

(2) z is a root of R, so $z = \varphi((\sigma, h))$ for some σ if and only if G is a subgroup of $\sigma V \sigma^{-1}$ for some σ.

Proof. To prove (1): this follows from the coefficients of R being symmetric functions in the α_i, so they are fixed by all Galois automorphisms hence they are in $\mathbb{Q}$. At the same time, since they are combinations of the α_i, they are algebraic integers. Hence, they are integral.

To prove (2): suppose $z \in \mathbb{Z}$ and $\sigma \in U$ satisfies $z - \varphi((\sigma, h)) = 0$, so $\varphi((\sigma, h)) = z$. By Lemma 5.3, this happens if and only if (σ, h) is invariant for G. Next, (σ, h) is a U-relative invariant for $\sigma V \sigma^{-1}$. Let $\sigma\tau\sigma^{-1} \in \sigma V \sigma^{-1}$. Then,

$$(\sigma\tau\sigma^{-1}, (\sigma, h)) = (\sigma\tau, h) = (\sigma, (\tau, h)) = (\sigma, h)$$

as $\tau \in V$ and h is invariant for V. By theorem 5.1 this occurs if and only if $G \subseteq \sigma V \sigma^{-1}$. □

So, we continue as follows: Start at $U = S_n$, and for each pair (U, V) where V is a maximal subgroup of U, compute h a U-relative invariant for V. An algorithm which does this is described in [3]. Next, check if h satisfies the condition in lemma 5.3. If not, apply a Tschirnhaus transformation to f to obtain a $\hat{f}$ such that it holds. With this $\hat{f}$, compute the relative resolvent for V. Next, invoke the algorithm in section 3.1 to check the resolvent for integral roots. If it does, then we know that $G \subseteq \sigma V \sigma^{-1}$ for some $\sigma \in U$. Recursively run the algorithm on all maximal subgroups of $\sigma V \sigma^{-1}$. At the base case, if we know that $G \subseteq U$ and the relative resolvents for each maximal subgroup of U do not have integral roots, then we can conclude that $G = U$.

6 Galois groups of reducible polynomials

At this point, for any polynomial f, we can factor f as $f = f_1 \cdot f_2 \cdots f_k$ and compute G_i, the Galois group of f_i. It is known that the Galois group G of f is isomorphic to a subgroup of $\prod G_i$, and our goal is to pick which one it is. However, the information we have alone is not enough.

For example, if $f = f_1 f_2$ where $f_1(x) = x^4 - 2$ and $f_2(x) = x^2 + 1$, then the splitting field for f_1 is $K_1 = \mathbb{Q}(i, \sqrt[4]{2})$ which has Galois group D_8 over $\mathbb{Q}$. Meanwhile, the splitting field for f_2 is $K_2 = \mathbb{Q}(i)$ which has Galois group $\mathbb{Z}/2\mathbb{Z}$ over $\mathbb{Q}$. Note that $K_2 \subseteq K_1$, hence $K_1 K_2 = K_1$ so the Galois group for f is just D_8. Alternatively, if $f = f_1 f_3$ where $f_3(x) = x^2 - 3$, then the splitting field for f_3 is $K_3 = \mathbb{Q}(\sqrt{3})$ which has trivial intersection with K_1, hence the Galois group for K_1K_3 is $D_8 \times (\mathbb{Z}/2\mathbb{Z})$. Critically, information about how the splitting fields interact is necessary to find the Galois group of the compositum field.

A polynomial called the "compositum polynomial" will help in low-degree cases. First, we define the resultant.

Definition 6.1 Let f_1 and f_2 be monic polynomials with coefficients in some integral domain R. Suppose f_1 has roots $\alpha_1, \ldots, \alpha_m$ and f_2 has roots $\beta_1, \ldots, \beta_n$ in a fixed algebraic closure of the quotient field of R. Then, the *resultant* of f_1 and f_2 is

$$\mathrm{Res}(f_1, f_2) = \prod_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} (\alpha_i - \beta_j)$$

Theorem 6.1 Let R be an integral domain, and $f_1, f_2 \in R[x]$. Then, $\mathrm{Res}(f_1, f_2) \in R$.

Proof. Corollary of Lemma 3.3.4 of [2]. □

Now, we can define the compositum polynomial, as defined in [1].

Definition 6.2 Let f_1, f_2 be irreducible monic polynomials with coefficients in $\mathbb{Z}$. Let

$$r(x) = \mathrm{Res}(f_1(y), f_2(x - y))$$

Then, the *compositum* of f_1 and f_2, denoted $\mathrm{comp}(f_1, f_2)$, is the largest irreducible factor of $r(x)$.

Since $f_1(y)$ and $f_2(x-y)$ can be regarded as polynomials with coefficients in $\mathbb{Z}[x]$. Theorem 6.1 tells us that $r(x)$, and hence $\mathrm{comp}(f_1, f_2)$, is a member of $\mathbb{Z}[x]$. The compositum has the property that it is the minimal polynomial of $\alpha_i + \beta_j$ for some i and j. This is because if $\alpha_1, \ldots, \alpha_m$ are the roots of f_1, then

$$f_1(y) = (y - \alpha_1) \cdots (y - \alpha_m)$$

and if $\beta_1, \ldots, \beta_n$ are the roots of f_2, then $x - \beta_1, \ldots, x - \beta_n$ are the roots of $f_2(x - y)$, so

$$f_2(x - y) = (y - (x - \beta_1)) \cdots (y - (x - \beta_n))$$

then, applying the resultant formula,

$$\operatorname{Res}(f_1(y), f_2(x-y)) = \prod_{\substack{1\le i\le m\\1\le j\le n}} ((x-\beta_j)-\alpha_i) = \prod_{\substack{1\le i\le m\\1\le j\le n}} (x-(\beta_j+\alpha_i))$$

So, if f_1 and f_2 are both quadratic, and have splitting fields $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$ respectively, then the splitting field for $\operatorname{comp}(f_1, f_2)$ is $\mathbb{Q}(\alpha+\beta)$. Thus, the degree of $\operatorname{comp}(f_1, f_2)$ is equal to the degree of $\mathbb{Q}(\alpha, \beta)$ over $\mathbb{Q}$. At this point, we can now give an algorithm to compute the Galois group of reducible polynomials up to degree 5.

6.1 Degree 2 and 3

If f has degree 2 or 3, then f has at most one irreducible factor of degree 2 or more. If f does indeed have one irreducible factor of degree 2 or more, say f_1, then this means that the roots of f consist of the roots of f_1 together with some roots lying in $\mathbb{Z}$. Thus, the Galois group of f is the Galois group of f_1, G_1. If f does not have an irreducible factor of degree 2 or more, then it necessarily factors over $\mathbb{Z}$, hence its Galois group is trivial.

6.2 Degree 4

If f has degree 4, then f is either irreducible, a product of two quadratics, or it reduces to a previous case. The only non-trivial one is when $f = f_1f_2$ where f_1 and f_2 are irreducible quadratics, in which case G is isomorphic to a subgroup of $(\mathbb{Z}/2\mathbb{Z})^2$.

To decide which subgroup G is, compute $g = \operatorname{comp}(f_1, f_2)$. If $\deg(g) = 4$, then the splitting fields of f_1 and f_2 intersect trivially, hence $G \cong (\mathbb{Z}/2\mathbb{Z})^2$. If $\deg(g) = 2$, then the splitting fields of f_1 and f_2 are equal, so $G \cong \mathbb{Z}/2\mathbb{Z}$.

6.3 Degree 5

If f has degree 5, then the only case that isn't covered in previous sections is when $f = f_1f_2$ when $\deg(f_1) = 3$ and $\deg(f_2) = 2$. Let K_1 and K_2 be the splitting fields for f_1 and f_2 respectively, and let G_1 and G_2 be the Galois groups of K_1 and K_2 over $\mathbb{Q}$. A priori, we know that G_1 is isomorphic to some transitive subgroup of S_3, so either $\mathbb{Z}/3\mathbb{Z}$ or S_3 itself. We also know that $G_2 \cong \mathbb{Z}/2\mathbb{Z}$, as it is the only transitive subgroup of S_2.

If $G_1 \cong \mathbb{Z}/3\mathbb{Z}$, then $G \cong G_1 \times G_2 \cong \mathbb{Z}/6\mathbb{Z}$. This is because the degrees of K_1 and K_2 are coprime, hence $K_1 \cap K_2 = \mathbb{Q}$. If $G_1 \cong S_3$, then G is isomorphic to a subgroup of $S_3 \times (\mathbb{Z}/2\mathbb{Z})$ which necessarily contains subgroups isomorphic to G_1 (S_3) and G_2 ($\mathbb{Z}/2\mathbb{Z}$). The only possible options for G are then $S_3 \times (\mathbb{Z}/2\mathbb{Z})$ or S_3. To decide these cases, we look at the intersection between K_1 and K_2. Clearly $G \cong S_3 \times (\mathbb{Z}/2\mathbb{Z})$ if and only if $K_1 \cap K_2 = \mathbb{Q}$. On the other hand, $G \cong S_3$ if and only if $K_2 \subseteq K_1$. We will use the theory of cubic fields to decide which case we are in.

Let d_1 be the discriminant of f_1. Then, d_1 can be written uniquely as $d_1 = ef^2$ where e is either 1 or a fundamental discriminant. Additionally, the splitting field for f_1 (which is K_1) contains a unique quadratic subfield $\mathbb{Q}(\sqrt{e})$ (Section 6.4.5 of [2]). Since this analysis takes place only when $G_1 \cong S_3$, then we know that d_1 is not a square, hence e is indeed a fundamental discriminant. Thus, $K_2 \subseteq K_1$ if and only if the discriminant of f_2 is equal to e. If the discriminant of f_2 is not equal to e, then $K_1 \cap K_2 = \mathbb{Q}$ and the Galois group is $S_3 \times (\mathbb{Z}/2\mathbb{Z})$. This leads to the following algorithm for computing the Galois group of $f_1 f_2$:

1. Compute G_1 and G_2, the Galois groups of f_1 and f_2 respectively.
2. If $G_1 \cong \mathbb{Z}/3\mathbb{Z}$, output $\mathbb{Z}/6\mathbb{Z}$.
3. Compute $\operatorname{disc}(f_1) \cdot \operatorname{disc}(f_2)$. If the result is square-free then return $S_3 \times (\mathbb{Z}/2\mathbb{Z})$. If the result is a square, then output S_3.

Algorithms to compute the Galois group for reducible polynomials of degree 6 and 7 are given in [1].

7 Future considerations

The clear bottleneck in the algorithm is section 6, as the algorithms in the previous sections are tractable when run on polynomials with degrees much higher than 7. So, any work to expand section 6 to higher degree would directly translate to making the algorithm more useful overall. This requires having a general algorithm to answer the following question: Given two fields

K_1 and K_2 which are Galois over $\mathbb{Q}$, what is the field K_1K_2? What is $K_1 \cap K_2$?

One idea for solving this is to make use of the LLL algorithm. If $K_1 = \mathbb{Q}(\alpha)$ and $K_2 = \mathbb{Q}(\beta)$, then we can use LLL to solve for the minimal polynomial of β over K_1 (and vice versa). This gives us the degree of K_1K_2 over K_1. In low-degree cases, this could yield enough information to decide the Galois group of K_1K_2, but in general it is not enough.

8 Acknowledgements

I would like to thank my advisor Amanda Tucker for her help in writing this thesis by providing feedback and helping me understand this topic. I would also like to thank Professor Haessig and Professor Jochnowitz for being on my thesis committee.

References

[1] C. Awtrey, T. Cesarski, and P. Jakes. Determining galois groups of reducible polynomials via discriminants and linear resolvents. *JP Journal of Algebra Number Theory and Applications*, 39(5):685–702, 2017.

[2] H. Cohen. A course in computational algebraic number theory. *Graduate Texts in Mathematics*, 1993.

[3] K. Girstmair. On invariant polynomials and their application in field theory. *Mathematics of Computation*, 48(178):781–797, 1987.

[4] A. Hulpke. Techniques for the computation of galois groups. In B. H. Matzat, G.-M. Greuel, and G. Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 65–77, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[5] R. Kannan, A. K. Lenstra, and L. Lovász. Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, page 191–200, New York, NY, USA, 1984. Association for Computing Machinery.

[6] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261, 12 1982.