The Long Shadow of Évariste Galois

Robert L. Miller

December 7, 2007

Contents

Chapter 1

Introduction

There are several spots in this paper where I would have said more, had I the time to chase out all the details. These spots are indicated by the label MISSING.

MISSING- a nice biographical sketch of Évariste.

Chapter 2

Galois Representations

2.1 Representation Theory

In general, mathematics benefits whenever two different types of objects are related by some kind of action. For example, studying the action of a group G on the set underlying the group gives rise to Cayley's theorem, which says that every finite group is isomorphic to a subgroup of the symmetric group $S_\infty = \varinjlim_n S_n$. Many of the results from Sylow theory come from studying the action of a group on some set, usually cosets of one group in another. Representation theory is the study of groups acting (linearly) on vector spaces.

Definition 2.1.1. A (linear) *representation* of a group G is a group homomorphism $\rho : G \to GL(V)$, where V is a vector space over a field F; in other words, a vector space V together with a G-action that acts linearly. Equivalently, a representation of a group G is a module V over the group ring FG, in which case we say that V *affords* the representation ρ. The *degree* of the representation is the dimension of V. The representation is *faithful* if the homomorphism is injective.

Definition 2.1.2. Fix a group G and a field F. Two representations are *equivalent* if the FG-modules affording them are isomorphic as modules. Terms such as *reducible, decomposable,* and *semisimple* are inherited from the corresponding modules.

One consequence of the following theorem is that every representation of a finite group over $\mathbb{Q}$ is semisimple.

Theorem 2.1.3 (Maschke's Theorem)**.** *If G is a finite group and F is a field of characteristic not dividing $|G|$, and V is an FG-module with a submodule U, then $V = U \oplus W$ for some submodule W.*

Proof. By assumption, $n = |G|$ is invertible in F. First, we know that there is a complement W_0 such that $V = U \oplus W_0$ as vector spaces. Let $\pi_0 : V \to U$ be the associated projection, and define $\pi : V \to U$ by

$$\pi = \frac{1}{n} \sum_{g \in G} g \pi_0 g^{-1},$$

the average of π_0 over G. U is an FG-module, so if $u \in U$, $g^{-1}(u) \in U$, and by definition $\pi_0(g^{-1}(u)) = g^{-1}(u)$. Therefore $\pi(u) = u$ for all $u \in U$, and it is easy to check that $\pi^2(v) = v$ for all $v \in V$. In fact, taking the average over G makes π an FG-module homomorphism, since

$$\pi(hv) = \frac{1}{n} \sum_{g \in G} g \pi_0(g^{-1}hv) = h \frac{1}{n} \sum_{h^{-1}g \in G} (h^{-1}g) \pi_0((g^{-1}h)v) = h\pi(v).$$

Therefore $V \cong U \oplus \ker \pi$ as FG-modules. □

The whole notion of averaging in the proof of Maschke's theorem depends crucially on the hypothesis that $|G|$ is invertible in F. This is meaningless when the characteristic of F divides the order of the group (see section 2.4).

Definition 2.1.4. *A character* of a group G over a field K is a one-dimensional representation $\chi : G \to GL_1(K) = K^\times$. The characters $\chi_1, ..., \chi_n$ are said to be *linearly independent* over K if they are independent as functions:

$$k_1\chi_1 + \cdots + k_n\chi_n = 0 : G \to K \Rightarrow k_1 = \cdots = k_n = 0.$$

Definition 2.1.5. *The character* χ_ρ of a representation $\rho : G \to \mathrm{GL}_n(K)$ is defined by $\chi_\rho(g) = \mathrm{tr}(\rho(g))$, where tr is the trace.

Every character value $\chi(g)$ is a sum of n^{th} roots of unity, as long as G contains no elements of infinite order. If $K = \mathbb{C}$ or $\overline{\mathbb{Q}}$, this implies that $\chi(g) \in \overline{\mathbb{Z}}$.

Theorem 2.1.6 (Linear Independence of Characters)**.** *If $\chi_1, ..., \chi_n$ are distinct characters of a group G over K, then they are linearly independent over K.*

Proof. If we have a nontrivial relation, then at least one k_i is nonzero, and after reordering, we have $k_1\chi_1(g) + \cdots + k_m\chi_m(g) = 0$ for all $g \in G$, with all of the k_i nonzero and $m \leq n$. Pick an element $g_0 \in G$ such that $\chi_1(g_0) \neq \chi_m(g_0)$. First, the relation with $g_0 g$ gives $\sum_{i=1}^m k_i\chi_i(g_0)\chi_i(g) = 0$. Next, we can multiply the relation with g by $\chi_m(g_0)$ to get $\sum_{i=1}^m k_i\chi_m(g_0)\chi_i(g) = 0$. Subtracting the one from the other, we obtain $\sum_{i=1}^{m-1}(\chi_m(g_0) - \chi_i(g_0))k_i\chi_i(g) = 0$. Since $\chi_m(g_0) - \chi_1(g_0) \neq 0$, this is a nontrivial relation involving one less character. We can repeat the process, eventually obtaining a nontrivial relation involving one character, $k_1'\chi_1(g) = 0$ for all $g \in G$. This implies $\chi_1(g) = 0$, a contradiction. □

2.2 Algebraic Field Extensions

From the very beginning, representation theory gives remarkable results in field theory. First, recall that any ring homomorphism $\sigma : F \to K$ between fields is either zero, or injective, since the kernel must be an ideal. Thus any nontrivial map of fields is an embedding.

Corollary 2.2.1 (to Theorem 2.1.6)**.** *If $\sigma_1, ..., \sigma_n : F \hookrightarrow K$ are distinct embeddings of fields, then they are linearly independent over K.*

Proof. Consider the group homomorphisms $\sigma_i|_{F^\times} : F^\times \to K^\times$. $\square$

Proposition 2.2.2. *Suppose $G \leq \mathrm{Aut}(K)$ is a finite subgroup of the set of field automorphisms of K, and F is the fixed field K^G. Then $[K : F] = |G|$.*

Proof. Denote $G = \{\sigma_1, ..., \sigma_n\}$, where $\sigma_1 = \mathrm{id}$. If $b_1, ..., b_m$ is a basis for K over F, then the system of equations

$$\sigma_1(b_i)x_1 + \cdots + \sigma_n(b_i)x_n = 0;\ i = 1, ..., m$$

will have a nontrivial solution $(x_1, ..., x_n) = (\beta_1, ..., \beta_n)$ if there are less equations than unknowns, i.e. if $m < n$. If this is the case, setting $a_1, ..., a_m \in F$, we have that $\sigma_j(a_i) = a_i$, since the σ_j fix F. Multiplying by a_i, we get the system of relations

$$\sigma_1(a_ib_i)\beta_1 + \cdots + \sigma_n(a_ib_i)\beta_n = 0;\ i = 1, ..., m,$$

which we can sum to obtain, with $b = a_1b_1 + \cdots a_mb_m$, the relation

$$\sigma_1(b)\beta_1 + \cdots + \sigma_n(b)\beta_n = 0,$$

a contradiction to Corollary 2.2.1. Thus we have $[K : F] = m \geq n$.

Suppose there are $n + 1$ F-independent elements of K, say $b_1, \cdots, b_{n+1}$. Then the system of n equations in $n + 1$ variables

$$\sigma_i(b_1)x_1 + \cdots + \sigma_i(b_{n+1})x_{n+1} = 0;\ i = 1, ..., n$$

has a nontrivial solution $(x_1, ..., x_{n+1}) = (\beta_1, ..., \beta_{n+1}) \in K^{n+1}$. Assume that the solution we have chosen has the minimal number s of zero entries, and that these are the first s entries, by reordering. Thus there are nonzero $\beta_1, ..., \beta_s \in K$ such that

$$\sigma_i(b_1)\beta_1 + \cdots + \sigma_i(b_s)\beta_s = 0;\ i = 1, ..., n,$$

and since β_s is nonzero, we can divide through, relabeling to obtain

$$\sigma_i(b_1)\beta_1 + \cdots + \sigma_i(b_{s-1})\beta_{s-1} + \sigma_i(b_s) = 0;\ i = 1, ..., n. \tag{2.2.1}$$

If all the β_j are in F, then since $\sigma_1 = \mathrm{id}$, the equation

$$\sigma_1(b_1)\beta_1 + \cdots + \sigma_1(b_{s-1})\beta_{s-1} + \sigma_1(b_s) = 0$$

is a contradiction, since the b_j are linearly independent over F. So at least one of the β_j is in $K \setminus F$: assume $\beta_1 \notin F$. Then there is an automorphism $\sigma_r, r \in \{1, ..., s\}$, such that $\sigma_r(\beta_1) \neq \beta_1$.

Since $G = \{\sigma_1, ..., \sigma_n\}$ is a group, we have $\{\sigma_1, ..., \sigma_n\} = \{\sigma_r\sigma_1, ..., \sigma_r\sigma_n\}$, so if we apply σ_r to the above equations, we can rewrite the result as

$$\sigma_j(b_1)\sigma_r(\beta_1) + \cdots + \sigma_j(b_{s-1})\sigma_r(\beta_{s-1}) + \sigma_j(b_s) = 0;\ j = 1, ..., n. \qquad (2.2.2)$$

Combining the two equation systems by subtracting equation 2.2.2 from equation 2.2.1 gives

$$\sigma_j(b_1)(\beta_1 - \sigma_r(\beta_1)) + \cdots + \sigma_j(b_{s-1})(\beta_{s-1} - \sigma_r(\beta_{s-1})) = 0;\ j = 1, ..., n.$$

However, this is a nontrivial $(\beta_1 - \sigma_r(\beta_1) \neq 0)$ solution to the original system of equations with fewer nonzero entries, a contradiction. Thus $[K : F] \leq n$. □

Corollary 2.2.3. *Given any finite extension K/F,*

$$|\text{Aut}(K/F)| \Big| [K : F].$$

Proof. Since the extension is finite, $K = F(\alpha_1, ..., \alpha_n)$. Each α_i must still satisfy its minimal polynomial after an automorphism, so each α_i has finitely many places to go. Therefore $\text{Aut}(K/F)$ is a finite group. The fixed field L of $\text{Aut}(K/F)$ lies somewhere between K and F, and $[K : F] = [K : L][L : F] = |\text{Aut}(K/F)|[L : F]$. □

Proposition 2.2.4. *Suppose K/F is an extension of fields, and $\alpha \in K$. Either $F(\alpha) \cong F(x)$, the field of rational functions over F, or there exists a unique monic irreducible polynomial $m_\alpha(x) \in F[x]$ such that $f(\alpha) = 0$ iff $m_\alpha | f$. Further, in this case, $F(\alpha) = F[\alpha]$, and $[F(\alpha) : F]$ equals the degree of m_α.*

Proof. If there is no nonzero polynomial in $F[x]$ with α as a root, then there is a ring isomorphism $F(x) \to F(\alpha)$ given by $f(x)/g(x) \mapsto f(\alpha)/g(\alpha)$. If there is some nonzero polynomial with α as a root, then suppose $m_\alpha(x)$ is such a polynomial of least degree, and by dividing through by the leading term, suppose $m_\alpha(x)$ is monic. Since $m_\alpha(\alpha) = 0$, $m_\alpha | f \Rightarrow f(\alpha) = 0$. Now if $f(\alpha) = 0$, divide f by m_α to obtain $0 = f(\alpha) = m_\alpha(\alpha)q(\alpha) + r(\alpha)$, where the degree of r is less than that of m_α, whence $r(\alpha) = 0$ implies $r = 0$. Thus $m_\alpha | f$. Then m_α is unique, since any other candidate would be monic, divide, and be divisible by m_α. Since $F[x]$ is a domain, m_α must be irreducible. The last facts come from examining the isomorphism $F(\alpha) \cong F[x]/(m_\alpha(x))$. □

Definition 2.2.5. In the first case, α is *transcendental* over F, and in the second, α is *algebraic* over F.

Corollary 2.2.6. *If K/F is an extension and $\alpha \in K$, then $F(\alpha)/F$ is a finite extension if and only if α is algebraic.*

Proof. Note that the extension $F(x)/F$ is infinite, since $\{1, x, x^2, ...\}$ is an independent set over F, and that $[F[x]/(m(x)) : F] = \deg m < \infty$. $\square$

Definition 2.2.7. A field extension K/F is *algebraic* if every element of K is algebraic over F.

Proposition 2.2.8. *(1) If K/L is a finite extension, then it is algebraic. (2) If K/L and L/F are algebraic extensions, then K/F is algebraic. (3) If $\{K_\alpha/F\}$ is a family of algebraic extensions, then $(\prod_\alpha K_\alpha)/F$ is an algebraic extension.*

Proof. (1) If K/L were finite but not algebraic, then there would be a transcendental element $t \in K$, and $1, t, t^2, ...$ would all be linearly independent, a contradiction.

(2) If $\beta \in K$, then since it is algebraic over L, it is the root of a polynomial $\beta_n x^n + \cdots + \beta_1 x + \beta_0$ with coefficients β_i in L. Indeed, β is algebraic over $F(\beta_0, \beta_1, ..., \beta_n)$, a finite extension of F, hence the extension $F(\beta_0, \beta_1, ..., \beta_n, \beta)/F$ is finite. In particular, $F(\beta)/F$ is also finite, hence β is algebraic over F.

(3) If $\beta \in \bigcup_\alpha K_\alpha$, then $\beta \in K_{\alpha_0}$ for some α_0. Then β is algebraic over F. $\square$

Example 2.2.9. Even though finite extensions are algebraic, consider the infinite sequence of extensions

$$\mathbb{Q} \subsetneq \mathbb{Q}(\sqrt{2}) \subsetneq \mathbb{Q}(\sqrt[4]{2}) \subsetneq \cdots .$$

The union of all of these is an algebraic extension, since any element lives in some $\mathbb{Q}(\sqrt[2^k]{2})$.

Any finite extension is algebraic, but algebraic extensions K/F in general need not be finite. Given an element $\alpha \in K$, it must be algebraic, so the extension $F(\alpha)/F$ is finite. Since $F(\alpha) \subset K$, we have that $K = \bigcup_{\alpha \in K} F(\alpha)$. This gives us at least some way of dealing with infinite algebraic extensions in terms of finite ones.

Definition 2.2.10. The field K is called an *algebraic closure* of F if K is algebraic over F and if every polynomial of $F[x]$ splits completely over K.

Definition 2.2.11. The field K is called *algebraically closed* if every polynomial with coefficients in K has a root in K. In particular, this implies that the polynomial will have all its roots in K.

It comes straight from the definitions that an algebraic closure is algebraically closed: a root α of a polynomial $f(x) \in K[x]$ generates an algebraic extension $K(\alpha)/K$, and $K(\alpha)/F$ is also algebraic. Then α is algebraic over F, and $\alpha \in K$.

Proposition 2.2.12. *Zorn's Lemma implies that any field F has an algebraic closure.*

Proof. Form the ring $B = F[..., x_f, ...]$ with indeterminates indexed by all the nonconstant monic polynomials $f(x) \in F[x]$ (B is for big). Let I be the ideal generated by $f(x_f)$ for all the $f(x)$. If this ideal is not a proper subset of B, then there are $g_1, ..., g_n \in B$ such that $g_1 f_1(x_{f_1}) + \cdots + g_n f_n(x_{f_n}) = 1$. There are only finitely many g_i, in finitely many variables involved in the f_i, so assume $x_{f_i} = x_i$ and define $x_{n+1}, ..., x_m$ to be the rest of them. There is a finite extension of F which contains roots α_i of the f_i, and in this extension, plugging in those roots gives the equation $0 = 1$, a contradiction. Therefore I is a proper ideal.

Zorn's Lemma implies that I is contained in some maximal ideal M, and since $I \subset M$, the field B/M contains at least one root of every polynomial in $F[x]$. Since each x_f is the root of some polynomial, the new field is algebraic. If we repeat this process a countable number of times, we get an increasing chain of algebraic extensions, and a polynomial of degree N must have split after only N extensions. The union of all these is again an algebraic extension over F, and every polynomial splits completely, so we have an algebraic closure. $\square$

The proof of the following fact is similar to that of 2.2.12, along with using extension of isomorphisms. Although it does not directly follow from the statement, it is still in some sense a

Corollary 2.2.13. *Two algebraic closures of the same field F are isomorphic.*

In light of the above, we often denote an algebraic closure of F by $\overline{F}$, but note that since $\mathrm{Aut}\left(\overline{F}\right)$ is (highly) nontrivial, $\overline{F}$ is only defined up to isomorphism. Often when $F = \mathbb{Q}$, $\overline{\mathbb{Q}}$ denotes the subset of $\mathbb{C}$ consisting of algebraic numbers.

2.3 Galois Theory

Suppose K/F is an algebraic field extension. Consider the group $G = \mathrm{Aut}(K/F)$ of field automorphisms of K that fix F. Denote the fixed set of σ by $K^\sigma = \{k \in K : \sigma(k) = k\}$, and the fixed set of G by $K^G = \bigcap_{\sigma \in G} K^\sigma$. Note that $F \subseteq K^G$. In terms of the group G, the more natural base of the extension seems to be the field K^G.

Definition 2.3.1. We say that the extension is *Galois* if the fixed field K^G is exactly F.

Example 2.3.2. Suppose $F = \mathbb{F}_p(t)$ is the field of rational functions over the finite field $\mathbb{F}_p$, and $K = \mathbb{F}_p(t^{1/p})$. The minimal polynomial of $t^{1/p}$ is $x^p - t = (x - t^{1/p})^p$, which has only one root. Thus any automorphism $\sigma \in \mathrm{Aut}(K/F)$ must fix $t^{1/p}$, hence all of K. Because $\mathrm{Aut}(K/F)$ is trivial, K/F is not a Galois extension.

Although the irreducible polynomial $x^p - t$ splits completely over K, it contains only one repeated root, and the automorphism group cannot detect the field F. Recall that a polynomial is *separable* if it splits into distinct linear factors in its splitting field.

Definition 2.3.3. An extension K/F is *separable* if every element of K is a root of a separable polynomial over F. In particular, every separable extension is algebraic.

Proposition 2.3.4. *The composite L_1L_2 of two separable extensions L_1/F and L_2/F is separable.*

Proof. The composite L_1L_2 is isomorphic to $L_1 \otimes_F L_2$ mod a maximal ideal M, as F-algebras: given any composite K in any field, we can map $L_1 \otimes_F L_2 \to K$ by multiplication. Note that since K is a field, the kernel is some maximal ideal. Also, by minimality, the map must be surjective. This proves that each K is isomorphic to L_1L_2. □

Example 2.3.5. Suppose $F = \mathbb{Q}$ is the rational numbers, and $K = \mathbb{Q}\left(\sqrt[3]{2}\right)$. First, suppose $\sigma : K \to K$ is an automorphism of K fixing $\mathbb{Q}$, and denote $\alpha = \sqrt[3]{2}$. Since $\alpha^3 = 2$, we have $\sigma(\alpha^3) = \sigma(\alpha)^3 = 2$, so σ sends α to another cube root of 2. In particular, $\sigma(\alpha) \in \{\alpha, \zeta\alpha, \zeta^2\alpha\}$, where ζ is a primitive 3rd root of unity. However, $\zeta \notin K$, so $\sigma(\alpha) = \alpha$. This implies that $\mathrm{Aut}(K/F)$ is the trivial group, and its fixed field is K, hence K/F is not Galois.

The irreducible polynomial $x^3 - 2$ has a root in K, but does not split completely, and again the automorphism group fixes a field larger than F.

Definition 2.3.6. An algebraic extension K/F is *normal* if every irreducible polynomial in $F[x]$ which has a root in K splits into linear factors over K. In other words, K is the splitting field of some collection of polynomials in $F[x]$.

These two properties characterize all Galois extensions.

Proposition 2.3.7. *If K/F is a finite extension, then it is Galois if and only if it is separable and normal.*

Proof. ([3], Theorem 13, Chapter 14. Also useful, Theorem 27, Chapter 13) □

Lemma 2.3.8. *If L/F is a finite separable extension, then there is a unique field L' such that L'/F is a Galois extension, $L' \supseteq L$, and any other Galois extension L''/F containing L also contains L'. Furthermore, L'/F is finite.*

Proof. Take the composite of the splitting fields of the minimal polynomials of a basis for L over F. Each of these splitting fields is separable, since L/F is, hence there is a finite Galois extension of F containing L. Now simply take the intersection of all such extensions. □

Definition 2.3.9. The field L' is called the *Galois closure* of F in L.

Proposition 2.3.10. *K/F is Galois if and only if it is separable and normal.*

Proof. $\Rightarrow$ If $\alpha \in K$, then the Galois closure of $F(\alpha)$ contains α, hence the minimal polynomial of α over F must be separable, by 2.3.7. If $f(x) \in F[x]$ is an irreducible polynomial with root $\alpha \in K$, then the minimal polynomial of α

is just $f(x)$ divided by its leading coefficient. Thus again by 2.3.7, $f(x)$ splits into linear factors over the Galois closure of $F(\alpha)$, which is contained in K.

$\Leftarrow$ We must show that for any $\alpha \in K \setminus F$, there is an automorphism $\sigma \in \mathrm{Aut}(K/F)$ such that $\sigma(\alpha) \neq \alpha$. Define L to be the splitting field of the minimal polynomial of α over F. By hypothesis, $F \subset L \subset K$, and since L/F is a Galois extension, there is an automorphism of L fixing F that does not fix α. Then the idea is to extend this automorphism to all of K. □

Definition 2.3.11. The *absolute Galois group* G_k of a field k is the Galois group $\mathrm{Gal}(k^{\mathrm{sep}}/k)$, where k^{sep} is a separable closure of k. If k is a perfect field, then $k^{\mathrm{sep}} \cong \overline{k}$, where $\overline{k}$ is any algebraic closure of k, and in this case $G_k = \mathrm{Gal}(\overline{k}/k)$. Often we will be interested in the most perfect field, $\mathbb{Q}$, and if the field is not specified, the absolute Galois group, and sometimes just the Galois group, means $G_{\mathbb{Q}}$.

The algebraic closure $\overline{\mathbb{Q}}$ of $\mathbb{Q}$ is equal to the union of all number fields, and for any two such fields $\mathbb{F} \subset \mathbb{F}'$ and $\sigma \in G_{\mathbb{Q}}$, we have that $\sigma|_{\mathbb{F}} = (\sigma|_{\mathbb{F}'})|_{\mathbb{F}}$. Thus we can write the absolute galois group as an inverse limit over number fields $\mathbb{F}$,

$$G_{\mathbb{Q}} = \varprojlim_{\mathbb{F}} \mathrm{Gal}\left(\mathbb{F}/\mathbb{Q}\right).$$

Note that every number field $\mathbb{F}$ is contained in a Galois closure $\mathbb{G}$ inside of $\overline{\mathbb{Q}} \supset \mathbb{G} \supseteq \mathbb{F}$. Hence we could also write the absolute galois group as an inverse limit over *Galois* number fields $\mathbb{G}$,

$$G_{\mathbb{Q}} = \varprojlim_{\mathbb{G}} \mathrm{Gal}\left(\mathbb{G}/\mathbb{Q}\right).$$

In general, when we are dealing with an inverse limit of finite groups, called a *profinite group* , there is a natural topology on the inverse limit. If $G = \varprojlim_{\alpha} G_\alpha$ is a profinite group, for each α there is a projection $G \twoheadrightarrow G_\alpha$. We can define a topology on G by requiring the kernels of these maps to be a basis of the identity, and assuming that multiplication is continuous. Reader beware! The same profinite group may end up having different topologies depending on which index set is used.

Definition 2.3.12. When considering the Galois group as an inverse limit of all number fields, the topology that arises is called the *Krull topology.* When considering only those number fields which are Galois over $\mathbb{Q}$, the topology that arises is called the *Galois topology.*

Often this distinction is confused, and terminology is not standard. For example, [2] and [4] both use the term Krull topology to refer to the Galois topology, whereas [3] defines the Krull topology the way we have. The term Galois topology is introduced here simply to clarify the situation. More generally, suppose L/F is a Galois extension. We can define the Galois and Krull topologies in exactly the same way.

Note that we *always* take the Galois topology.

Theorem 2.3.13 (The Fundamental Theorem of Galois Theory)**.** *Suppose L/F is a Galois extension. The correspondence*

$$K \mapsto \mathrm{Gal}(L/K)$$

where K is an intermediate field extension $F \subset K \subset L$, is an inclusion reversing bijection between the set $\{K : F \subset K \subset L\}$ of all intermediate field extensions and the set of all closed subgroups of $\mathrm{Gal}(L/F)$*. Normal closed subgroups correspond to Galois extensions, and open subgroups correspond to finite extensions.*

Proof. See [4], page 3. □

Proposition 2.3.14. *If $L \subset \overline{\mathbb{Q}}$ is a Galois extension of $\mathbb{Q}$, then every map $L \longrightarrow \overline{\mathbb{Q}}$ has its image in L.*

Proof. A nonzero map $L \longrightarrow \overline{\mathbb{Q}}$ is an isomorphism onto its image, and this isomorphism can be extended to all of $\overline{\mathbb{Q}}$, and L is a Galois conjugate of its image. Since L is Galois, they are the same. □

2.4 Modular Representation Theory

Example 2.4.1. Suppose we have an elliptic curve E defined over $\mathbb{Q}$. E is an example of an abelian variety, which is a projective algebraic variety, together with an algebraic group structure. Group operation and taking inverses act as regular functions, and projectivity implies (nontrivially) that the group is abelian (so denote the group law by $+$). The complex points $E(\mathbb{C})$ of E are isomorphic to the quotient of $\mathbb{C}$ by a rank 2 lattice Λ, with complex addition as the group law. This isomorphism is defined in terms of the Weierstrass elliptic function $\wp$, by

$$\mathbb{C}/\Lambda \longrightarrow E(\mathbb{C}) : z \mapsto \begin{cases} [\wp(z) : \wp'(z) : 1] & z \notin \Lambda \\ [0:1:0] & z \in \Lambda \end{cases}$$

Consider the subgroup $E[p] = \{Q \in E : pQ = 0\}$, where $pQ = Q + \cdots + Q$, p times, called the *p-torsion* subgroup. This group is abstractly isomorphic to $(\mathbb{Z}/p\mathbb{Z})^2$, since Λ is a full rank sublattice of $\wp^{-1}(E[p])$. $E[p]$ is a vector space over $\mathbb{F}_p$ of dimension 2. The coordinates of its points generate a field, denoted $\mathbb{Q}(E[p])$. This field sits inside of $\overline{\mathbb{Q}}$, since the defining equation of E is a polynomial over $\mathbb{Q}$. The absolute Galois group $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$ acts on the coordinates of $E[p]$, since the coefficients of the defining equation are fixed. The action of this group on $E[p]$ factors through a finite Galois group $\mathrm{Gal}(K/\mathbb{Q})$ for some number field K. In fact, K is just the Galois closure of $\mathbb{Q}(E[p])$.

So we have a mod p representation of the finite group $G = \mathrm{Gal}(K/\mathbb{Q})$:

$$G \xrightarrow{\rho} \mathrm{GL}_2(\mathbb{F}_p) \cong \mathrm{Aut}(E[p])$$

This is called a modular Galois representation, since the base field is $\mathbb{F}_p$. Two good references for this are [5] and [6].

In general, if the characteristic p of a field F divides $|G|$, then Maschke's theorem does not hold, and the group algebra is not generally semisimple. This belongs to the domain of modular representation theory, which was essentially founded by Richard Brauer.

A mod p ℓ-isogeny is a surjective morphism $\varphi : E[p] \longrightarrow F[p]$ with finite kernel of order ℓ. The Fourier coefficients of an elliptic curve are defined by

$$a_p(E) = p + 1 - \#E(\mathbb{F}_p),$$

and as it turns out they are the traces of the Frobenius endomorphism, which gives them the special role of being Brauer characters. Thus, it is a consequence of the Brauer-Nesbitt theorem (see [1]) that

Proposition 2.4.2. *If*

$$a_q(E) \equiv a_q(F) \mod p$$

for all primes $q \nmid pN_EN_F\ell$ where N_E is the conductor of E and the same for F, then there exists a mod p ℓ-isogeny between E and F. In particular, $E[p] \cong F[p]$ as Galois representations.

MISSING - A better understanding of how this works.

2.5 p-adic Galois Representations

One can also consider the full set of maps given by multiplication by p:

$$\begin{tikzcd} E[p] & E[p^2] \arrow[l, two heads, "{[p]}"', "\subset"] & E[p^3] \arrow[l, two heads, "{[p]}"', "\subset"] & \cdots \arrow[l, two heads, "{[p]}"', "\subset"] \end{tikzcd}$$

Since the embedding $\overline{\mathbb{Q}} \hookrightarrow \mathbb{C}$ is fixed, so are the isomorphisms $E[p^k] \cong (\mathbb{Z}/p^k\mathbb{Z})^2$ (and they respect the maps $[p]$), yielding that the p-adic Tate module

$$\mathrm{Ta}_p(E) := \varprojlim_k E[p^k] \cong \mathbb{Z}_p^2$$

is group-isomorphic to two copies of the p-adic integers $\mathbb{Z}_p$. Each $\mathbb{Q}(E[p^k])$ is Galois over $\mathbb{Q}$, and restriction commutes, in the sense of the following diagram.

$$\begin{tikzcd} & G_{\mathbb{Q}} \arrow[dl, "\cdot|_{\mathbb{Q}(E[p^k])}"'] \arrow[dr, "\cdot|_{\mathbb{Q}(E[p^{k+1}])}"] & \\ \mathrm{Gal}(\mathbb{Q}(E[p^k])/\mathbb{Q}) & & \mathrm{Gal}(\mathbb{Q}(E[p^{k+1}])/\mathbb{Q}). \arrow[ll, "{[p] \circ \cdot}"'] \end{tikzcd}$$

In other words, the Tate module is actually a $G_{\mathbb{Q}}$-module, and hence defines a representation

$$\rho_{E,p} : G_{\mathbb{Q}} \longrightarrow GL_2(\mathbb{Z}_p).$$

It turns out that this representation is irreducible, and is usually called the p-adic Galois representation arising from the elliptic curve E.

Chapter 3

Galois Cohomology

3.1 Derived Functors

MISSING - but for a good reference, see [10].

3.2 Group Cohomology

For this section and the next, one can follow along with the treatment in chapter VII of [7].

Definition 3.2.1. If G is a group acting on an abelian group A, then this action affords a module structure on A over the group algebra $\mathbb{Z}[G]$. Since A is an abelian group, it is already a $\mathbb{Z}$-module. We define the $\mathbb{Z}[G]$-module structure by $\left(\sum_{g\in G} n_g g\right) \cdot a = \sum_{g\in G} n_g(g \cdot a)$. We call A a *G-module.*

Definition 3.2.2. A sequence of G-module homomorphisms

$$\begin{tikzcd} A_1 \arrow[r, "f_1"] & A_2 \arrow[r, "f_2"] & \cdots \arrow[r, "f_{n-1}"] & A_n \end{tikzcd}$$

is said to be *exact* if $\ker f_{i+1} = \operatorname{im} f_i$ for each $i = 1, 2, \ldots, n-2$. Often $n = 5$ and $A_1 = A_5 = 0$. In this case we have a *short exact sequence*

$$\begin{tikzcd} 0 \arrow[r] & A_2 \arrow[r, "f_2"] & A_3 \arrow[r, "f_3"] & A_4 \arrow[r] & 0, \end{tikzcd}$$

and it is a trivial consequence of the definition that in this case, f_2 is injective and f_3 surjective.

Example 3.2.3. Any surjective map $A \twoheadrightarrow B$ can be extended to a short exact sequence, by the first isomorphism theorem:

$$\begin{tikzcd} 0 \arrow[r] & \ker(A \twoheadrightarrow B) \arrow[r] & A \arrow[r] & B \arrow[r] & 0. \end{tikzcd}$$

Letting A^G denote the subset of A fixed by the action of G, notice that if $f : A \longrightarrow B$ is a homomorphism of G-modules, then $f(A^G) \subset B^G$. Thus there is a functor $A \mapsto A^G$.

Definition 3.2.4. A functor F is called *left exact* if whenever

$$0 \longrightarrow A \longrightarrow B \longrightarrow C$$

is exact, then so is

$$0 \longrightarrow F(A) \longrightarrow F(B) \longrightarrow F(C).$$

Proposition 3.2.5. *The functor $A \mapsto A^G$ is left exact.*

Proof. Suppose

$$0 \longrightarrow A \xrightarrow{f} B \xrightarrow{g} C$$

is an exact sequence of G-modules. To show that

$$0 \longrightarrow A^G \xrightarrow{f} B^G \xrightarrow{g} C^G$$

is exact, we just need to establish that the map $A^G \longrightarrow B^G$ is injective and that $\text{im}(A^G \longrightarrow B^G) = \ker(B^G \longrightarrow C^G)$. Thinking of $A \longrightarrow B$ as an inclusion should make the prior clear. To show the latter, suppose $a \in A^G$. Since the original sequence was exact, $g(f(a)) = 0$. On the other hand if $b \in B^G$ such that $g(b) = 0$, then there is an $a \in A$ such that $f(a) = b$. We must only show that $g \cdot a = a$ for all $g \in G$. Noting that $f(g \cdot a) = g \cdot f(a) = g \cdot b = b = f(a)$, since f is injective, $a \in A^G$ as desired. □

If $\mathbb{Z}$ is the G-module with trivial action $g \cdot n = n$, then $A^G = \text{Hom}_G(\mathbb{Z}, A)$ in the following sense. An element of A^G is the image $f(1)$ of 1 under some map $f : \mathbb{Z} \longrightarrow A$, since the G-action has been trivialized: $g \cdot f(1) = f(g \cdot 1) = f(1)$. Further, the elements of A^G are the only possible images of such homomorphisms. This allows for the interpretation of the cohomology $H^q(G, A)$ as the derived functors $\text{Ext}^q(\mathbb{Z}, A)$ of the functor Hom_G.

Given a short exact sequence of G-modules

$$0 \longrightarrow A \longrightarrow B \longrightarrow C \longrightarrow 0,$$

there is a connecting homomorphism $\delta : H^q(G, C) \longrightarrow H^{q+1}(G, A)$ such that

$$\cdots \longrightarrow H^q(G, B) \longrightarrow H^q(G, C) \xrightarrow{\delta} H^{q+1}(G, A) \longrightarrow h^{q+1}(G, B) \longrightarrow \cdots$$

is a long exact sequence.

For an explicit construction of H^*, we first use a specific free resolution of $\mathbb{Z}$. Let P_i be the free $\mathbb{Z}$-module generated by $(g_0, ..., g_i)$ for $g_j \in G$, where G acts on all components: $g \cdot (g_0, ..., g_i) = (gg_0, ..., gg_i)$. Define $d : P_i \longrightarrow P_{i-1}$ by

$$d(g_0, ..., g_i) = \sum_{j=0}^{i} (-1)^j (g_0, ..., \hat{g_j}, ..., g_i),$$

and define $P_0 \longrightarrow \mathbb{Z}$ by mapping each g_j onto $1 \in \mathbb{Z}$. We now have the exact sequence of projective (except for $\mathbb{Z}$) G-modules

$$\cdots \longrightarrow P_i \longrightarrow P_{i-1} \longrightarrow \cdots \longrightarrow P_1 \longrightarrow P_0 \longrightarrow \mathbb{Z} \longrightarrow 0.$$

As $\mathrm{Hom}_G(P_i, A)$ ranges over i, we have a cochain complex, and we can define

$$\mathrm{H}^q(G, A) = H^q(\mathrm{Hom}_G(P_i, A)).$$

In this cochain complex, elements of $\mathrm{Hom}_G(P_i, A)$ are just homomorphisms $f : P_i \longrightarrow A$, and the coboundary map is given by

$$\begin{aligned} df(g_0, ..., g_i) = & g_0 \cdot f(g_1, ..., g_i) \\ & + \sum_{j=0}^{i-1} (-1)^j f(g_1, ..., g_j g_{j+1}, ..., g_i) \\ & + (-1)^i f(g_0, ..., g_{i-1}). \end{aligned} \qquad (*)$$

3.3 Changing Groups

A group homomorphism $f : G \longrightarrow H$ can be used to define a G-module structure on an H-module A by

$$g \cdot a = f(g) \cdot A.$$

To indicate that we are pulling back the group module structure by f, we write the resulting G-module as f^*A. This provides a morphism of functors from $H^0(H, A)$ to $H^0(G, f^*A)$. By the universal property of derived functors, this extends to a morphism of cohomology functors, denoted

$$f_q^* : H^q(H, A) \longrightarrow H^q(G, f^*A).$$

Example 3.3.1. If $f : G \hookrightarrow H$ is the inclusion of a subgroup G into H, the resulting morphism is denoted

$$\mathrm{Res} : H^q(H, A) \longrightarrow H^q(G, A),$$

for *restriction.*

Definition 3.3.2. Given an additive map $g : A \longrightarrow B$ between an H-module A and a G-module B, we say that f and g are *compatible* if g is a G-module homomorphism from f^*A to B.

In this case, g gives a homomorphism $H^q(G, f^*A) \longrightarrow H^q(G, B)$, which when composed with the above map gives the homorphism *associated* to the pair (f, g)

$$(f, g)_q^* : H^q(H, A) \longrightarrow H^q(G, B).$$

Example 3.3.3. Suppose H is a normal subgroup of G. Letting $f : G \longrightarrow G/H$ be the quotient map and $g : A^H \longrightarrow A$ be the inclusion, it is easy to show they are compatible. The resulting morphism is denoted

$$\mathrm{Inf} : H^q(G/H, A^H) \longrightarrow H^q(G, A)$$

for *inflation.*

3.4 Elliptic Curves

Recall 2.4.1. For p a prime and ℓ another prime, it is easy to check for ℓ-isogenous congruences $E[p] \cong F[p]$ (see section 2.4), which imply that the Galois representations are equivalent. If this is the case, consider the following commutative diagram:

$$\begin{tikzcd}
0 \arrow[r] & E[p] \arrow[r] \arrow[d, "\cong"'] & E(\overline{\mathbb{Q}}) \arrow[r, "{[p]}"] & E(\overline{\mathbb{Q}}) \arrow[r] & 0 \\
0 \arrow[r] & F[p] \arrow[r] & F(\overline{\mathbb{Q}}) \arrow[r, "{[p]}"] & F(\overline{\mathbb{Q}}) \arrow[r] & 0
\end{tikzcd}$$

Applying the functor H^* we obtain the long diagram (recall that $H^0(G, A) = A^G$):

$$\begin{tikzcd}
\cdots \arrow[r] & E(\overline{\mathbb{Q}})^{G_{\mathbb{Q}}} \arrow[r, "{[p]}"] & E(\overline{\mathbb{Q}})^{G_{\mathbb{Q}}} \arrow[r, "\delta"] & H^1(G_{\mathbb{Q}}, E[p]) \arrow[r] \arrow[d, "\cong"'] & \cdots \\
\cdots \arrow[r] & F(\overline{\mathbb{Q}})^{G_{\mathbb{Q}}} \arrow[r, "{[p]}"] & F(\overline{\mathbb{Q}})^{G_{\mathbb{Q}}} \arrow[r, "\delta"] & H^1(G_{\mathbb{Q}}, F[p]) \arrow[r] & \cdots
\end{tikzcd}$$

Note that $E(\overline{\mathbb{Q}})^{G_{\mathbb{Q}}}$ is just $E(\mathbb{Q})$, so we can rewrite the above diagram in terms of the cokernel of $[p]$:

$$\begin{tikzcd}
0 \arrow[r] & E(\mathbb{Q})/pE(\mathbb{Q}) \arrow[r, "\delta"] & H^1(G_{\mathbb{Q}}, E[p]) \arrow[d, "\cong"'] & (**) \\
0 \arrow[r] & F(\mathbb{Q})/pF(\mathbb{Q}) \arrow[r, "\delta"] & H^1(G_{\mathbb{Q}}, F[p]) &
\end{tikzcd}$$

The construction (*) gives us formulas for working with this diagram. Any cocyle of $H^1(G_{\mathbb{Q}}, E[p])$ can be expressed as a crossed homomorphism, which is a homomorphism $f : G_{\mathbb{Q}} \longrightarrow E[p]$ satisfying

$$f(\sigma\sigma') = \sigma \cdot f(\sigma') + f(\sigma).$$

To get the full picture, we must mod out by coboundaries, which are functions of the form

$$f(\sigma) = \sigma(S) - S$$

for some $S \in E[p]$.

Given a point $\overline{P} \in E(\mathbb{Q})/pE(\mathbb{Q})$, we can express its image as a homomorphism

$$\delta(\overline{P}) : G_{\mathbb{Q}} \longrightarrow E[p] : \sigma \mapsto \sigma(R) - R$$

for R such that $pR = P$. Notice that $\delta(\overline{P})$ is trivial iff $R \in E[p]$ iff $P = O$ is the identity.

Lemma 3.4.1. *If $E[p] \cong F[p]$ as Galois representations, then*

$$\mathbb{Q}(E[p]) = \mathbb{Q}(F[p]).$$

Proof. First note that the field $\mathbb{Q}(E[p])$ is Galois over $\mathbb{Q}$. To obtain it, you first split the division polynomial, whose roots α_i are the x-coordinates of $E[p]$. To obtain the y-coordinates, you plug α_i into x in the defining equation for E, and solve for y. This is also a polynomial you are splitting, so in the end the extension is Galois. The same goes for $\mathbb{Q}(F[p])$.

Since $E[p] \cong F[p]$ as Galois representations, this isomorphism extends to an isomorphism $\mathbb{Q}(E[p]) \cong \mathbb{Q}(F[p])$, which by Proposition 2.3.14 must be equality. □

Note that this implies that the representations factor through the same Galois extension:

$$\begin{tikzcd}
G_{\mathbb{Q}} \arrow[rr] \arrow[dr] & & \mathrm{Aut}(E[p]) \\
& \mathrm{Gal}(\mathbb{Q}(E[p])/\mathbb{Q}) \arrow[ur] \arrow[d, "="'] & \\
& \mathrm{Gal}(\mathbb{Q}(F[p])/\mathbb{Q}) \arrow[dr] & \\
G_{\mathbb{Q}} \arrow[rr] \arrow[ur] & & \mathrm{Aut}(F[p])
\end{tikzcd}$$

The particular case of interest is when E is a curve of rank 2, and F is a curve of rank 1. Further, we will assume that the representations arising from these curves are surjective, e.g. the homomorphism

$$\rho : G_{\mathbb{Q}} \longrightarrow \mathrm{Aut}(E[p])$$

is surjective. Let $G_K := \mathrm{Gal}(\overline{\mathbb{Q}}/K)$ for K an intermediate extension of $\overline{\mathbb{Q}}/\mathbb{Q}$. Note that this is a generalization of our notation for $G_{\mathbb{Q}}$.

Suppose the Mordell-Weil groups are presented as $E(\mathbb{Q}) = \langle P_1, P_2\rangle \times E(\mathbb{Q})_{\mathrm{tor}}$ and $F(\mathbb{Q}) = \langle R\rangle \times F(\mathbb{Q})_{\mathrm{tor}}$. Let

$$K = \mathbb{Q}\left(\frac{1}{p}\cdot R\right)$$

be the extension of $\mathbb{Q}$ by a fixed p^{th} root of R, and for $a, b \in \mathbb{Z}$, similarly let

$$K_{a,b} = \mathbb{Q}\left(\frac{1}{p}\cdot (a\cdot P_1 + b\cdot P_2)\right)$$

be the extension of $\mathbb{Q}$ by a fixed p^{th} root of $P_{a,b} := a\cdot P_1 + b\cdot P_2$.

In the diagram (**), we are interested to know whether $\delta(E(\mathbb{Q})/pE(\mathbb{Q}))$ and $\delta(F(\mathbb{Q})/pF(\mathbb{Q}))$ are related. If $\delta(\overline{P}_{a,b})$ and $\delta(\overline{R})$ are scalar multiples, then their kernels, as crossed homomorphisms, are the same. Recall that

$$\delta(\overline{P}_{a,b}) : \sigma \mapsto \sigma\left(\frac{1}{p}\cdot P_{a,b}\right) - \frac{1}{p}\cdot P_{a,b}.$$

The kernel of this is exactly those $\sigma \in G_{\mathbb{Q}}$ fixing $\frac{1}{p}\cdot P_{a,b}$, i.e. $G_{K_{a,b}}$. Thus by the Galois correspondence, if $\delta(\overline{P}_{a,b})$ and $\delta(\overline{R})$ are scalar multiples, then $K = K_{a,b}$.

On the other hand, suppose the compositums $K.\mathbb{Q}(F[p]) = K_{a,b}.\mathbb{Q}(E[p])$ are equal. We begin by examining the isomorphism

$$H^1(G_{\mathbb{Q}}, E[p]) \xrightarrow{\cong} H^1(G_{\mathbb{Q}}, F[p]).$$

Applying the restriction map Res coming from the inclusion $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q}(E[p])) \hookrightarrow \mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$, and similarly for F, the isomorphism extends:

$$\begin{tikzcd}
H^1(G_{\mathbb{Q}}, E[p]) \arrow[r, "\cong"] \arrow[d, "\mathrm{Res}"'] & H^1(G_{\mathbb{Q}}, F[p]) \arrow[d, "\mathrm{Res}"'] \\
H^1(G_{\mathbb{Q}(E[p])}, E[p]) \arrow[r, "\cong"] & H^1(G_{\mathbb{Q}(F[p])}, F[p]).
\end{tikzcd}$$

The kernel of

$$\mathrm{Res} : H^1(\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q}), E[p]) \longrightarrow H^1(\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q}(E[p])), E[p])$$

is

$$H^1(\mathrm{Gal}(\mathbb{Q}(E[p])/\mathbb{Q}), E[p]) \approx H^1(\mathrm{GL}_2(\mathbb{F}_p), \mathbb{F}_p^2),$$

since the representations are surjective. The claim is that this kernel is zero, so that Res is injective.

If $p = 2$, then $\mathrm{GL}_2(\mathbb{F}_p) \cong S_3$. Suppose $\sigma : S_3 \longrightarrow \mathbb{Z}/2\mathbb{Z}^2$ is a group homomorphism. By order, $\sigma((1\ 2\ 3)) = 0$, hence $\sigma(a) + \sigma(b) = \sigma(ab) = 0$ for every $a, b \in S_3$. Further, they must be crossed homomorphisms, so $0 = \sigma(ab) = a\sigma(b) + \sigma(a)$ for all $a, b \in S_3$. Since $p = 2$, this means $a\sigma(b) = \sigma(a)$ for all a, b, so $b = 0$ gives that $\sigma(a) = 0$ for all $a \in S_3$. Thus the kernel is trivial. If $p \neq 2$, the argument at the end of the proof of Proposition 5.1 in [9] shows that the kernel must still be zero.

Therefore the Res maps are injective. Since the action of $G_{\mathbb{Q}(E[p])} = \mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q}(E[p]))$ on $E[p]$ is trivial, we have that

$$H^1(G_{\mathbb{Q}(E[p])}, E[p]) \cong \mathrm{Hom}(G_{\mathbb{Q}(E[p])}, E[p]),$$

which means we can write the diagram as

$$\begin{tikzcd}
H^1(G_{\mathbb{Q}}, E[p]) \arrow[r, "\cong"] \arrow[d, hook, "\mathrm{Res}"'] & H^1(G_{\mathbb{Q}}, F[p]) \arrow[d, hook, "\mathrm{Res}"'] \\
\mathrm{Hom}(G_{\mathbb{Q}(E[p])}, E[p]) \arrow[r, "\cong"] & \mathrm{Hom}(G_{\mathbb{Q}(F[p])}, F[p]).
\end{tikzcd}$$

By injectivity, to compare $\delta(\overline{R})$ and $\delta(\overline{P_{a,b}})$ in $H^1(G_{\mathbb{Q}}, E[p]) \cong H^1(G_{\mathbb{Q}}, F[p])$, we can compare $\delta(\overline{R})$ to $\delta(\overline{P_{a,b}})$ in

$$\mathrm{Hom}(G_{\mathbb{Q}(E[p])}, E[p]) \cong \mathrm{Hom}(G_{\mathbb{Q}(F[p])}, F[p]).$$

Recall from Proposition 2.3.14 that $\mathbb{Q}(E[p]) = \mathbb{Q}(F[p])$, and that we have assumed that $K.\mathbb{Q}(F[p]) = K_{a,b}.\mathbb{Q}(E[p])$. Now, we have concrete descriptions of the maps, by

$$\delta(\overline{P_{a,b}}) : \sigma \mapsto \sigma\left(\frac{1}{p} \cdot P_{a,b}\right) - \frac{1}{p} \cdot P_{a,b}$$
$$\delta(\overline{R}) : \sigma \mapsto \sigma\left(\frac{1}{p} \cdot R\right) - \frac{1}{p} \cdot R,$$

and the Galois correspondence implies that the subgroups $G_{K.\mathbb{Q}(F[p])}$ and $G_{K_{a,b}.\mathbb{Q}(E[p])}$ of $G_{\mathbb{Q}(E[p])}(= G_{\mathbb{Q}(F[p])})$ are equal. These are the subgroups fixing $\frac{1}{p} \cdot R$ and $\frac{1}{p} \cdot P_{a,b}$, respectively. As elements of $\mathrm{Hom}(G_{\mathbb{Q}(F[p])}, F[p])$, the maps $\delta(\overline{P_{a,b}})$ and $\delta(\overline{R})$ are both functions on the cosets of $G_{K.\mathbb{Q}(F[p])}$. MISSING - This seems pretty strong, but I'm not quite sure where to go from here. However, this is pretty good cause to investigate whether it is possible that $K.\mathbb{Q}(F[p]) = K_{a,b}.\mathbb{Q}(E[p])$.

The primes ramifying in $\mathbb{Q}(E[p])$, i.e. those dividing Np, will ramify in the compositums, so if the primes not dividing Np that ramify in K are the same as those that ramify in $K_{a,b}$, this is strong evidence that the compositums may be equal.

Chapter 4

Computations and Code

4.1 Computing $E[n]$

To compute $E[n]$, the main tool is the division polynomial, which is defined in terms of E and n, and whose roots are the x-coordinates of the points of $E[n]$. Given an x-coordinate, the corresponding y-coordinate is one of two solutions to a quadratic equation. For example, if the elliptic curve is presented as $y^2 = x^3 + ax + b$, then the y coordinate is just one of the two square roots of the right side when the x coordinate is plugged in. That is exactly what the following function, using the new `QQbar` Sage class. For the group law functions, see the appendix.

```
def compute_e_bracket_n(E, n):
    f = E.division_polynomial(n)
    x_coords = f.roots(ring=QQbar)
    g = E.defining_polynomial()
    y = polygen(QQbar,'y')
    points = []
    for x in x_coords:
        h = g(x[0],y,1)
        rootsh = h.roots(ring=QQbar)
        for root in rootsh:
            points.append([x[0], root[0], QQbar(1)])
    row1 = [[QQbar(0),QQbar(1),QQbar(0)]]
    gen1 = points[0]
    for i in xrange(1,n):
        row1.append(group_law(E, row1[i-1], gen1))
    output = [row1]
    gen2 = [p for p in points if p not in row1][0]
    for i in xrange(1,n):
        row_i = []
        summer = group_law_times_n(E, gen2, i)
        for j in xrange(n):
            row_i.append(group_law(E, summer, row1[j]))
        output.append(row_i)
    return output
```

4.2 Visualizing $E[n]$

$E[n]$ is a set of points in projective space $\mathbb{CP}^2$, which can be difficult to visualize. However each individual coordinate lives in the complex plane, and one of the coordinates can be taken to be 1. Since the last coordinate is always 1 unless the point is the identity of the curve, we plot the x coordinate in red, the y coordinate in cyan, and connect the coordinates of the same curve by an edge. It seems ad hoc, but the resulting plots can be enlightening. For example, the plots at `http://wiki.rlmiller.org/ModularIsogenies` seem to indicate a limiting process in the structure of $E[n]$ for a fixed E as n increases.

The following implements that scheme.

```
def plot_e_bracket_n(E, n):
    G = Graph()
    positions = {}
    xes = []
    ys = []
    En = []
    for row in compute_e_bracket_n(E, n):
        En += row
    for i in xrange(len(En)):
        p = En[i]
        if p[2] == 0:
            continue
        G.add_edge(((i,0), (i,1)))
        xes.append((i,0))
        ys.append((i,1))
        positions[(i,0)] = [p[0].real().n(),p[0].imag().n()]
        positions[(i,1)] = [p[1].real().n(),p[1].imag().n()]
    return G.plot(pos=positions, partition=[xes,ys], vertex_labels=False, vertex_size=10)
```

4.3 Detecting the Isomorphism

This is relatively simple code, since the Brauer-Nesbitt theorem does all the lifting.

```
def exists_isogeny(E, F, ell, p):
    N = lcm(E.conductor(), F.conductor())
    for q in prime_range(2, N*ell/6+1):
        if mod(p*N*ell, q) == 0:
            continue
        if mod(E.ap(q) - F.ap(q), p) != 0:
            return False
    return True
```

4.4 Computing the Isomorphism

This is probably the most interesting computation. Given a point P of $E[n]$, we can form the number field $\mathbb{Q}(P)$, which has several embeddings $\mathbb{Q}(P) \hookrightarrow \overline{\mathbb{Q}}$. Each of these embeddings corresponds to an actual choice of coordinates for P— abstractly, the number field is just the quotient of a polynomial ring by a polynomial, and the roots of that polynomial are algebraically indistinguishable. If Q is a point of $F[n]$ that P maps to, then the isomorphism $\mathbb{Q}(E[n]) \cong \mathbb{Q}(F[n])$ restricts to an isomorphism $\mathbb{Q}(P) \cong \mathbb{Q}(Q)$, which we can exploit in the following diagram:

$$\begin{tikzcd}
\overline{\mathbb{Q}} & \overline{\mathbb{Q}} \\
\mathbb{Q}(P) \arrow[u, hook, shift left] \arrow[u, hook, shift right] \arrow[r, "\cong"] & \mathbb{Q}(Q) \arrow[u, hook, shift left] \arrow[u, hook, shift right]
\end{tikzcd}$$

The first thing to do is actually construct $\mathbb{Q}(P)$. We do so in a similar manner to finding $E[n]$, by utilizing the division polynomial, but notice the flaw in the function: we only ever consider the first factor. This was an intentional dodge, but in the computations I was working on, it never came up, so it never got fixed.

```
def QQ_adjoin_P(E, n):
    f = E.division_polynomial(n)
    if not f.is_irreducible():
        f = f.factor()[0][0]
    f *= lcm([a.denom() for a in f.coeffs()])
    x = polygen(QQ)
    f_n = f.leading_coefficient()
    d_f = f.degree()
    f = (f_n)^(d_f-1)*f(x/f_n)
    l.<x> = NumberField(f)
    y = polygen(l)
    g = E.defining_polynomial()(x/f_n,y,1)
    if g.is_irreducible():
        g *= lcm([a.denominator() for a in g.coeffs()])
        g_n = g.leading_coefficient()
        g = g_n*g(y/g_n) # degree 2
        L.<y> = l.extension(g)
        return L, x/f_n, y/g_n
    else:
        g = g.factor()[0][0]
        return l, x/f_n, g.roots()[0][0]
```

Then we have the main function. Note again the ad hoc choice of when two embeddings are equal, since we are actually working in $\mathbb{C}$. This is like using floating points instead of pairs of integers to express rational numbers– the times involved with exact computations is prohibitive.

```
def find_the_map(E, F, n):
    isogeny_known = False
    for ell in prime_range(24):
        if exists_isogeny(E, F, ell, n):
            isogeny_known = True
            break
    if not isogeny_known:
        return None
    LE, xE, yE = QQ_adjoin_P(E, n)
    LF, xF, yF = QQ_adjoin_P(F, n)
    isomorphism = LE.embeddings(LF)[0]
    E_points = LE.embeddings(CC)
    F_points = LF.embeddings(CC)
    map = []
    for E_embedding in E_points:
        # print (E_embedding(xE), E_embedding(yE)) # the "E-point"
        # see where isomorphism sends this point
        # by seeing which embedding of F matches up
        for F_embedding in F_points:
            if abs(F_embedding(isomorphism(xE)) - E_embedding(xE)) < 0.01\
           and abs(F_embedding(isomorphism(yE)) - E_embedding(yE)) < 0.01:
                map.append (((E_embedding(xE), E_embedding(yE)),\
                             (F_embedding(xF), F_embedding(yF))))
    return map
```

4.5 Visualizing the Isomorphism

Given the ability to in some way visualize the structures $E[n]$, visualizing the isomorphisms is essentially trivial. There are two examples at
`http://wiki.rlmiller.org/VisualizingIsogenies`

```
def animate_the_map(E, F, n):
    map = find_the_map(E, F, n)
    start = {}
    end = {}
    for i in xrange(len(map)):
        start[i] = map[i][0]
        end[i] = map[i][1]
    plots = []
    frames=60.0
    for j in xrange(frames):
        G = Graph()
        xes = []
        ys = []
        pos = {}
        for i in xrange(len(map)):
            c = []
            c.append((start[i][0]*j + end[i][0]*(frames-j-1))/(frames-1))
            c.append((start[i][1]*j + end[i][1]*(frames-j-1))/(frames-1))
            G.add_edge(((i,0), (i,1)))
            xes.append((i,0))
            ys.append((i,1))
            pos[(i,0)] = [c[0].real(), c[0].imag()]
            pos[(i,1)] = [c[1].real(), c[1].imag()]
        plots.append(G.plot(pos=pos, partition=[xes,ys], \
                      vertex_labels=False, vertex_size=20))
    a = animate([plots[0]]*5 + plots + [plots[-1]]*5, axes=True)
    return a
```

4.6 Specific Isomorphisms

```
p = 3
for E in cremona_optimal_curves(range(389,600)):
    #if mod(E.ap(l)**2 - (l+1)**2, p) == 0:
    if E.rank() == 2:
        print E.cremona_label()
        N = E.conductor()
        for l in prime_range(24):
            if mod(E.ap(l)**2 - (l+1)**2, p) == 0:
                print 'searching for l =',l,'isogenous curves'
                for F in cremona_optimal_curves(N*l):
                    if F.rank() == 1 and exists_isogeny(E, F, l, p):
                        print F.cremona_label()
                        for arrow in find_the_map(E, F, p):
                            print 'x:', arrow[0][0]
                            print '    -->', arrow[1][0]
                            print 'y:', arrow[0][1]
                            print '    -->', arrow[1][1]
                            print '---'
```

```
389a1
```

```
searching for l = 3 isogenous curves
searching for l = 5 isogenous curves
searching for l = 7 isogenous curves
searching for l = 17 isogenous curves
searching for l = 19 isogenous curves
433a1
searching for l = 3 isogenous curves
1299b1
x: -0.0832972061521252 - 4.89065187323140e-16*I
    --> 26.9947984375809 + 3.62493168359436e-13*I
y: -0.958929557168226 + 4.42184597612172e-16*I
    --> 94.5494283783160 + 2.18571840113652e-14*I
---
x: 0.715034173184315 - 1.36768528783611*I
    --> -5.82274094491875 + 4.14500682085519*I
y: -0.365737680861901 + 2.41209641252603*I
    --> 24.4805983223508 - 18.4436099598329*I
---
x: 0.715034173184306 + 1.36768528783611*I
    --> -5.82274094491877 - 4.14500682085519*I
y: -0.365737680861900 - 2.41209641252603*I
    --> 24.4805983223508 + 18.4436099598329*I
---
x: 0.715034173184316 + 1.36768528783609*I
    --> -5.82274094491877 - 4.14500682085519*I
y: -0.349296492322414 + 1.04441112468994*I
    --> -18.6578573774320 - 14.2986031389777*I
---
x: 0.715034173184314 - 1.36768528783609*I
    --> -5.82274094491877 + 4.14500682085519*I
y: -0.349296492322414 - 1.04441112468994*I
    --> -18.6578573774320 + 14.2986031389777*I
---
x: -1.68010447354982 - 3.17523785042795e-14*I
    --> -15.6826498810727 + 3.77475828372553e-14*I
y: 0.840052236774915 + 1.74264996282730*I
    --> 7.84132494053640 + 34.9365587005177*I
---
x: -1.68010447354985 - 5.88418203051333e-15*I
    --> -15.6826498810727 + 1.19904086659517e-14*I
y: 0.840052236774919 - 1.74264996282730*I
    --> 7.84132494053635 - 34.9365587005177*I
---
x: -0.0832972061521240 + 3.13640467093313e-15*I
    --> 26.9947984375775 + 7.07432214568246e-13*I
y: 1.04222676332035 - 1.07260429134786e-15*I
```

```
    --> -121.544226815893 - 7.67122990644701e-14*I
---
searching for l = 13 isogenous curves
searching for l = 17 isogenous curves
searching for l = 19 isogenous curves
446d1
searching for l = 7 isogenous curves
3122a1
x: 2.20842029485001 + 1.86281924050492e-15*I
    --> 70.2915304558367 - 2.05823029771960e-10*I
y: -2.61391538316307 - 6.68995682128181e-16*I
    --> 416.066770190395 - 3.45764569239556e-13*I
---
x: 1.02048677414838 + 0.509149810472889*I
    --> -21.0936474236108 + 3.19552539475431*I
y: -1.22276581148438 + 0.697874682450763*I
    --> 37.2904371881605 - 18.3107685349351*I
---
x: 1.02048677414838 - 0.509149810472890*I
    --> -21.0936474236108 - 3.19552539475430*I
y: -1.22276581148438 - 0.697874682450764*I
    --> 37.2904371881605 + 18.3107685349351*I
---
x: 1.02048677414838 + 0.509149810472891*I
    --> -21.0936474236108 + 3.19552539475433*I
y: 0.202279037335998 - 1.20702449292365*I
    --> -16.1967897645497 + 15.1152431401808*I
---
x: 1.02048677414838 - 0.509149810472894*I
    --> -21.0936474236108 - 3.19552539475433*I
y: 0.202279037335998 + 1.20702449292366*I
    --> -16.1967897645497 - 15.1152431401808*I
---
x: 2.20842029485002
    --> 70.2915304567137 - 9.00893804403036e-11*I
y: 0.405495088313050
    --> -486.358300646657 + 1.68424944772764e-13*I
---
x: -3.24939384314601 - 1.17439391544849e-12*I
    --> -27.1042356090396 - 1.08801856413265e-14*I
y: 1.62469692157339 + 5.02297380793674*I
    --> 13.5521178045198 - 33.5900984533761*I
---
x: -3.24939384314650 - 1.37401201527609e-12*I
    --> -27.1042356090397 + 1.82076576038526e-14*I
y: 1.62469692157339 - 5.02297380793675*I
```

```
    --> 13.5521178045198 + 33.5900984533761*I
---
563a1
searching for l = 3 isogenous curves
searching for l = 7 isogenous curves
searching for l = 13 isogenous curves
searching for l = 17 isogenous curves
searching for l = 23 isogenous curves
571b1
searching for l = 3 isogenous curves
searching for l = 7 isogenous curves
searching for l = 11 isogenous curves
6281a1
x: 0.373214754019828 + 3.94823306886335e-15*I
    --> 36.1881265717571 - 3.44405452995275e-12*I
y: -1.47386602810255 + 1.98768473663071e-15*I
    --> -170.606789622641 - 6.63173220042760e-14*I
---
x: 1.12810213627080 - 0.552334409759726*I
    --> -14.8201889628355 + 4.95934380473844*I
y: -0.964272502698682 + 1.05225417093902*I
    --> 29.4707962665893 + 13.6248876911133*I
---
x: 1.12810213627080 + 0.552334409759728*I
    --> -14.8201889628355 - 4.95934380473843*I
y: -0.964272502698681 - 1.05225417093902*I
    --> 29.4707962665893 - 13.6248876911133*I
---
x: -3.96275235989491 - 4.58799664926346e-13*I
    --> -7.88108197938353 - 2.95596880306448e-15*I
y: -0.499999999999996 + 5.33144394446369*I
    --> -0.500000000000005 - 28.3102089388883*I
---
x: -3.96275235989491 + 4.57689441901721e-13*I
    --> -7.88108197938352 + 6.93889390390723e-16*I
y: -0.499999999999996 - 5.33144394446369*I
    --> -0.500000000000001 + 28.3102089388883*I
---
x: 1.12810213627080 + 0.552334409759727*I
    --> -14.8201889628354 - 4.95934380473839*I
y: -0.0357274973013203 + 1.05225417093902*I
    --> -30.4707962665893 + 13.6248876911133*I
---
x: 1.12810213627080 - 0.552334409759727*I
    --> -14.8201889628355 + 4.95934380473842*I
y: -0.0357274973013198 - 1.05225417093902*I
```

```
    --> -30.4707962665893 - 13.6248876911133*I
---
x: 0.373214754019830
    --> 36.1881265717551 - 1.37346836080256e-12*I
y: 0.473866028102545
    --> 169.606789622641 + 2.64469516245661e-14*I
---
searching for l = 13 isogenous curves
searching for l = 19 isogenous curves
```

Chapter 5

Appendix

5.1 A Conversation with Carl Witty

```
[8:11pm] cwitty: You can’t do roots(ring=QQbar) yet, no matter what
the base ring is.
[8:11pm] cwitty: I would have thought that roots(ring=CIF) would
work, but it seems not to.  Let me try to figure out why.
[8:12pm] cwitty: OK, simple workaround.
[8:13pm] cwitty: (Simple for me, a little complicated for you.
[8:13pm] cwitty: from sage.rings.polynomial.complex_roots import
complex_roots
[8:13pm] cwitty: complex_roots(p, min_prec=53)
[8:13pm] cwitty: That should work if p is in QQbar[’x’].
[8:13pm] cwitty: Then you can take the results and construct a new
QQbar element.
[8:14pm] rlm: does the constructor use LLL?
[8:14pm] cwitty: No.
[8:14pm] rlm: what does complex_roots return?
[8:15pm] cwitty: complex_roots returns the same thing that
.roots(ring=CIF) does: roots as complex intervals, with
multiplicities.
[8:15pm] rlm: oh, so it’s exact the whole time
[8:15pm] cwitty: It’ll be slow on QQbar inputs because it tries to
compute a squarefree decomposition.
[8:16pm] cwitty: If you know that p is already squarefree, you can
speed it up by adding skip_squarefree=True.
[8:16pm] cwitty: But if you say skip_squarefree=True and the
polynomial is not squarefree, then complex_roots will loop forever.

...
```

```
[8:21pm] cwitty: BTW, what polynomial degrees and coefficient
bitsizes are you dealing with?
[8:21pm] rlm: i start by finding the roots of a degree 4 polynomial
[8:21pm] rlm: then these become part of the coefficients on a
quadratic
[8:22pm] rlm: and solving the quadratic is taking a long time
[8:22pm] cwitty: Really?  I would have thought that would be a
reasonable thing to do.
[8:23pm] cwitty: If you put an example script on sage.math (or paste
it in IRC, if it’s small enough) I’ll take a look.
[8:24pm] rlm: sage: E = EllipticCurve(’389a’)
[8:24pm] rlm: sage: f = E.division_polynomial(3); f
[8:24pm] rlm: 3*x^4 + 4*x^3 - 12*x^2 + 3*x - 3
[8:24pm] rlm: sage: roots = f.roots(ring=CIF)
[8:24pm] rlm: sage: f = E.defining_polynomial()
[8:24pm] rlm: sage: y = polygen(QQbar)
[8:24pm] rlm: sage: g = f(x_coord[0],y,3)
[8:24pm] rlm: i would like to factor g

...

[8:30pm] cwitty: sage: x_coord = QQbar.polynomial_root(f, roots[2][0])
[8:30pm] cwitty: sage: f = E.defining_polynomial()
[8:30pm] cwitty: sage: y = polygen(QQbar)
[8:30pm] cwitty: sage: g = f(x_coord,y,3)
[8:31pm] cwitty: sage: rootsg = complex_roots(g)
[8:31pm] cwitty: sage: y_coord = QQbar.polynomial_root(g, rootsg[1][0])
[8:32pm] cwitty: And just to show off that the result really is an
exact number:
[8:32pm] cwitty: sage: y_coord.minpoly()
[8:32pm] cwitty: x^8 + 12*x^7 + 3511/81*x^6 + 109/9*x^5 - 8435/27*x^4 - 8299/9*x^3 - 32456/
[8:32pm] rlm: hooray
[8:33pm] cwitty: sage: \%timeit complex_roots(g)
[8:33pm] cwitty: 10 loops, best of 3: 19.8 ms per loop
[8:33pm] cwitty: sage: \%timeit complex_roots(g, skip_squarefree=True)
[8:33pm] cwitty: 100 loops, best of 3: 2.58 ms per loop
[8:34pm] cwitty: sage: ComplexField(500)(y_coord)
[8:34pm] cwitty: 0.051710268491798811792983643952680797906902843334
751631130827660731238158814702944380954999682236194233911538078772
32888715311138011312101881282680264077 -
0.9398446925171662144546142797979621127747806957807649583336804748
743027172916820660586860179164010564009223515765494548833791366177
59554970424388611456*I
[8:34pm] rlm: applause
[8:35pm] cwitty: Thank you, thank you.
```

```
...

[8:39pm] cwitty: Is that the highest polynomial degree you need?
Unfortunately, it’s likely to be much slower with slightly higher
degrees.
[8:42pm] rlm: actually, there is no need to be perfectly exact
[8:42pm] rlm: i just want to study a group action on a finite set of
points
[8:43pm] cwitty: So you don’t need to test for equality?
[8:43pm] rlm: nope
[8:44pm] cwitty: Then AFAIK, you should be able to avoid exact
computation altogether.  Just be careful about which operations you
use.
[8:44pm] rlm: do everything over CIF?
[8:44pm] cwitty: Always use skip_squarefree=True, for example (and
hope that the polynomials really are squarefree.)
[8:45pm] cwitty: Unfortunately, complex_roots() doesn’t work with
CIF inputs.

...

[8:47pm] cwitty: Well, CIF and QQbar both have guarantees that if
you use them right, you’re guaranteed to get a right answer.
[8:47pm] cwitty: With CIF the guarantees are much weaker, because
the "right answer" might be a really huge interval.
[8:48pm] cwitty: But with CC, you need to understand floating-point
rounding and the numpy and Pari root-finding routines to be sure
that your answer is meaningful.
[8:48pm] rlm: well, i need to know what the galois group does to
this finite set of coordinates
[8:48pm] rlm: it’ll permute them, so...
[8:49pm] rlm: seems like I could just find the closest point
[8:49pm] cwitty: Probably.
[8:49pm] cwitty: Do you know the coordinates now?  (Do you know that
the coordinates never have a pair that’s very close together?)
[8:50pm] rlm: well, i’m going to be doing this for a whole
collection of examples, so it is probably something where they may
sometimes get close, but usually they won’t
[8:51pm] cwitty: You could try something like finding the closest
point, and then checking the ratio of (distance to closest point) /
(distance to second-closest point).
[8:51pm] cwitty: And give up with an error if that number isn’t
tiny.
[8:52pm] cwitty: That should be pretty safe; and if you ever get an
error, then you redo that particular example with CIF or with QQbar.
[8:53pm] rlm: What’s the precision of a root of a polynomial over
```

```
CC?
[8:53pm] cwitty: CC = ComplexField(53), so about 53 bits.  But you
could also use ComplexField(200), or whatever.
[8:54pm] cwitty: For precision 53 or less (so including CC), we use
numpy's root-finding routines.
[8:54pm] cwitty: For higher precision, we use Pari.
[8:54pm] rlm: I thought we had a discussion before about roots of a
polynomial losing a lot of precision?
[8:54pm] rlm: Was that in the case of multiplicities?
[8:54pm] cwitty: The example that people were using was in case of
multiplicities, yes.
[8:55pm] cwitty: It depends on the polynomial.  High-degree
polynomials are bad; multiple roots are bad; ...
[8:56pm] cwitty: Wilkinson's polynomial (http://en.wikipedia.org
wiki/Wilkinson\%27s_polynomial) is bad.
[8:56pm] rlm: A bad boy, like Z/2Z.
[8:56pm] rlm: So for a separable polynomial, is there an estimate of
how bad, in terms of the degree?
[8:57pm] cwitty: In any of these cases, tiny errors in the
polynomial coefficients (which are inevitable, when dealing with
floating point) can make large errors in the final root locations.
[8:58pm] cwitty: You can't just look at the degree; you also need to
look at how close the roots are to each other, and in what sort of
pattern.
[8:58pm] rlm: by the way, do you mind if I include some of this
discussion in a number theory project i'm doing for william?
[8:58pm] cwitty: There's actually a discussion of the topic in that
Wikipedia article on Wilkinson's polynomial.
[8:59pm] cwitty: Go ahead.
[8:59pm] rlm: That is a good article.
[9:02pm] rlm: Let's say I have a polynomial defined over CC, and I
find its roots. Is there any way I can get a guarantee on how
accurate it is?
[9:02pm] cwitty: For my current project, I want guaranteed answers;
which is why I did the real interval wrapper, and AA (the real
counterpart to QQbar).
[9:03pm] cwitty: I don't know of a good way to do that.  (So I
implemented a really cheesy, fairly inefficient method for
complex_roots.py.)
[9:04pm] cwitty: By CC, do you mean the mathematical domain, or do
you mean the Sage implementation (using finite-precision floating
point)?
[9:04pm] rlm: The implementation, with precision n.
[9:04pm] cwitty: If the latter, you would need at the very least
some sort of bound on how far away your coefficients were from the
true coefficients.
```

```
[9:05pm] rlm: The coefficients start out exact, and then their
accuracy would depend on the root finding.
[9:06pm] cwitty: roots(ring=ComplexField(n)), for n>53, uses Pari's
polroots.
[9:06pm] cwitty: The polroots documentation says:
[9:06pm] cwitty: "Barring bugs, it
[9:06pm] cwitty: is guaranteed to converge and to give the roots to
the required accuracy."
[9:06pm] cwitty: But I don't know what that means.  Are you supposed
to be able to trust every bit of the result?
[9:06pm] cwitty: Because I don't believe that.
[9:07pm] rlm: "It is better not to question the internals of a
program as complicated as Mathematica."
[9:07pm] cwitty: You know, I really don't know that much about how
to guarantee the accuracy of floating-point computations.
[9:08pm] cwitty: (If you've read the Wikipedia article on
Wilkinson's polynomial, then you know about as much as I do about
numerical stability for medium-degree polynomial root finding, for
instance.)
[9:09pm] cwitty: That's why I use intervals, is so I don't have to
think about proving bounds on rounding errors.
[9:09pm] rlm: Actually I misspoke before.
[9:10pm] rlm: I'll mostly be working with p=3, so I think that my
polynomials will be mostly degree 8 or 9.
[9:10pm] rlm: And then solving quadratics defined in terms of their
roots.
[9:11pm] cwitty: In that case, QQbar will probably be fast enough;
and if not, a careful mix of QQbar and CIF probably will be.
```

5.2 New and Improved, Wheel

These functions were written, based on the rules in [8], before the author became familiar with the native syntax:

```
sage: E = EllipticCurve('389a1')
sage: P1, P2 = E.gens()
sage: 3*P1 - 2*P2
(-21866/22801 : -6834807/3442951 : 1)

sage: def group_law_minus(E, P):
...       x, y, z = P
...       if z == 0: return [x, y, z]
...       return [x, -y - E.a1()*x - E.a3(), 1]
sage: def group_law(E, P1, P2):
...       x_1, y_1, z_1 = P1; x_2, y_2, z_2 = P2
...       if P1 == [0,1,0]: return P2
...       if P2 == [0,1,0]: return P1
...       if x_1 == x_2:
...           if y_1 + y_2 + E.a1()*x_2 + E.a3() == 0:
...               return [0,1,0]
...           lamb = (3*x_1*x_1 + 2*E.a2()*x_1 + E.a4() - E.a1()*y_1)/\
...                  (2*y_1 + E.a1()*x_1 + E.a3())
...           nu = (-x_1*x_1*x_1 + E.a4()*x_1 + 2*E.a6() - E.a3()*y_1)/\
...                (2*y_1 + E.a1()*x_1 + E.a3())
...       else:
...           lamb = (y_2 - y_1) / (x_2 - x_1)
...           nu = (y_1*x_2 - x_1*y_2) / (x_2 - x_1)
...       x_3 = lamb*lamb + E.a1()*lamb - E.a2() - x_1 - x_2
...       y_3 = -(lamb + E.a1())*x_3 - nu - E.a3()
...       return [x_3, y_3, 1]
sage: def group_law_times_n(E, P, n):
...       Q = P
...       if n == 0: return [0,1,0]
...       for i in xrange(abs(n)-1):
...           Q = group_law(E, Q, P)
...       if n > 0: return Q
...       if n < 0: return group_law_minus(E, Q)
```

Bibliography

[1] Charles W. Curtis and Irving Reiner. *Representation Theory of Finite Groups and Associative Algebras.* Interscience Publishers, 1966.

[2] Fred Diamond and Jerry Shurman. *A First Course in Modular Forms.* Springer, 2005.

[3] David S. Dummit and Richard M. Foote. *Abstract Algebra.* John Wiley & Sons, Inc., 2004.

[4] Jürgen Neukirch. *Class Field Theory.* Springer-Verlag, 1986.

[5] Kenneth A. Ribet. Galois representations and modular forms. *Bulletin of the AMS*, 32(4):375–402, 1995.

[6] Kenneth A. Ribet and William A. Stein. Lectures on serre's conjectures. *Arithmetic Algebraic Geometry*, 2001.

[7] Jean-Pierre Serre. *Local Fields.* Spinger-Verlag, 1979.

[8] Joseph Silverman. *Arithmetic of Elliptic Curves.* Springer-Verlag, 1986.

[9] William A. Stein. There are genus one curves over $\mathbb{Q}$ of every odd index. *Journal für die reine und angewandte Mathematik*, 2002.

[10] Charles A. Weibel. *An Introduction to Homological Algebra.* Cambridge University Press, 1995.