RUDZINSKI, SANDI, M.A. Symbolic Computation of Resolvents. (2017)
Directed by Dr. Sebastian Pauli. 46 pp.

Resolvent polynomials are used in the determination of Galois groups of polynomials. The computation of the resolvent usually relies on root approximations requiring a high degree of precision. Leonard Soicher developed a method to compute absolute linear resolvents symbolically without the need for root approximations. This thesis details that method and expands it to compute relative linear resolvents symbolically with respect to the wreath product of S_n and a transitive permutation group G.

SYMBOLIC COMPUTATION OF RESOLVENTS

by

Sandi Rudzinski

A Thesis Submitted to
the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Arts

Greensboro
2017

Approved by

Committee Chair

APPROVAL PAGE

This thesis written by Sandi Rudzinski has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Chair	Sebastian Pauli
Committee Members	Chad Awtrey
	Brett Tangedal
	Dan Yasaki

Date of Acceptance by Committee

Date of Final Oral Examination

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Pauli, for his support, patience and kindness, and for understanding my questions. Dr. Pauli has been consistently supportive and helpful, and I am extremely grateful for his commitment and guidance. I would like to thank my committee, Dr. Yasaki, Dr. Awtrey, and Dr. Tangedal, for their time and assistance. Dr. Yasaki has been especially helpful and willing to provide advice and suggestions as I was writing. I would also like to thank Jonathan Milstead for providing some of the proofs for Chapter V and some of the code used in generating examples.

These acknowledgments are insufficient to express my gratitude. My experience at UNCG has been exceptional. The faculty have encouraged me at every step. Thank you.

TABLE OF CONTENTS

LIST OF ALGORITHMS

CHAPTER I

INTRODUCTION

Resolvent polynomials are used in the determination of Galois groups of polynomials. The computation of the resolvent usually relies on root approximations requiring a high degree of precision. Leonard Soicher developed a method to compute absolute linear resolvents symbolically without the need for root approximations. This thesis details that method and expands it to compute relative linear resolvents symbolically with respect to the wreath product of S_n and a transitive permutation group G.

In Chapter II, we cover the basic definitions from algebra and set up some of the notation. Chapter III explains the symbolic computation of absolute linear resolvent polynomials as developed by Leonard Soicher. In Chapter IV, a new algorithm for computing relative linear resolvents with respect to $S_m \wr S_l$ is given. Chapter V outlines the theorems for the use of the relative resolvent polynomial in determining Galois groups and gives some examples of the algorithm.

All algorithms presented in this thesis have been adapted for efficiency and implemented in the MAGMA computer algebra system. The implementations were used in the construction of the examples in Chapter V.

CHAPTER II

FOUNDATIONS

This chapter contains the necessary background and definitions. These definitions are compiled primarily from [DF04] and [Gei97].

2.1 Permutation Groups

A permutation of a set is a rearrangement of the set, a bijection from the set to itself. Galois groups are often thought of as permutation groups of the roots of a polynomial; so, we begin by defining the group of all permutations of a set, the symmetric group.

Definition 2.1 (Symmetric Group, S_n)**.** Let Ω be any nonempty set, and let S_Ω be the set of all bijections from Ω to itself (i.e., the set of all permutations of Ω). The set S_Ω is a group under function composition: $\circ$. This group is called the *symmetric group on the set* Ω.
In the special case when $\Omega = \{1, 2, 3, \ldots, n\}$, the symmetric group on Ω is denoted S_n, the *symmetric group of degree n.*

Often, we are interested not in the entire symmetric group, but a subgroup of it.

Definition 2.2 (Permutation Group)**.** Let Ω be any nonempty set and S_Ω the symmetric group on the set Ω. If G is any subgroup of S_Ω, then G is called a *permutation group* on Ω, and we denote this group by (G, Ω).

Definition 2.3 (Group Action)**.** A *(right) group action* of a group G on a set A is a map from $A \times G$ to A (written as $a \cdot g$ for all $g \in G$ and $a \in A$) satisfying the following properties:

(1) $(a \cdot g_1) \cdot g_2 = a \cdot (g_1 g_2)$, for all $g_1, g_2 \in G, a \in A$, and

(2) $a \cdot 1 = a$ for all $a \in A$, where 1 is the identity element of G.

All actions in this thesis are right group actions unless otherwise stated.

Definition 2.4 (Permutation Representation)**.** Let G be a group acting on a set A. Define $\phi : G \to S_A$ by $g \mapsto \sigma_g$ where $\sigma_g(a) = a \cdot g$ for any $a \in A$. Then ϕ is a homomorphism called the *permutation representation of* G with respect to the action on A.

Definition 2.5 (Invariant)**.** Let G be a permutation group acting on a set A, an element $a \in A$ is said to be *invariant* under G if $a \cdot g = a$ for all $g \in G$.

Definition 2.6 (Stabilizer)**.** Let G be a permutation group acting on a set A. Let $B \subseteq A$. The *stabilizer of B in G* is $\text{Stab}_G(B) = \{\sigma \in G \mid \sigma(b) \in B \text{ for all } b \in B\}$. $\text{Stab}_G(B) \leq G$ for all $B \subseteq A$.

Definition 2.7 (Orbit, Transitive)**.** Let G be a group acting on a nonempty set A. The equivalence class $\{a \cdot g \mid g \in G\}$ is called the *orbit of G containing a*. The action of G on A is called *transitive* if there is only one orbit, i.e., given any two elements $a, b \in A$, there is some $g \in G$ such that $a = b \cdot g$.

Definition 2.8 (Block and Block System)**.** Let G be a transitive permutation group acting on a set X.

(1) A subset B of X is called a *block* of G if for all $g \in G$, we have:

$B \cdot g = B$ or $B \cdot g \cap B = \emptyset$.

The number of elements in a block is called the *length* of the block.

(2) If $B_1, \ldots, B_m$ are blocks of G, we call $\mathfrak{B} = \{B_1, \ldots, B_m\}$ a *block system* of G if

(a) $\bigcup_{1 \leq i \leq m} B_i = X$.

(b) $B_i \cap B_j = \emptyset$ for $i \neq j$.

(c) All blocks have the same length.

$\mathfrak{B} = \{X\}$ and $\mathfrak{B} = \{\{x\} \mid x \in X\}$ are block systems for any transitive group (G, X). These are the trivial block systems.

Definition 2.9 (Primitive / Imprimitive)**.** The action of the transitive group G is called *primitive* if there is no nontrivial block system. Otherwise, the action of G is called *imprimitive.*

Definition 2.10 (Cosets)**.** Let G be a group. For any $N \leq G$ and any $g \in G$ let

$$gN = \{gn \mid n \in N\} \qquad Ng = \{ng \mid n \in N\}$$

called respectively a *left coset* and a *right coset* of N in G. Any element of a coset is called a *representative* for the coset.

Definition 2.11 (Transversal)**.** Let H be a subgroup of a group G. Then a subset S of G is termed a *right (left) transversal* of H in G if S intersects every right (left) coset of H at exactly one element.

S is also termed a system of right coset representatives of H. We denote a right transversal of H in G as $G//H$.

2.2 Galois Groups

Definition 2.12 (Automorphism Group). Let G be a group. An isomorphism σ of G with itself is called an *automorphism* of G. The collection of automorphisms of G is a group denoted $\mathrm{Aut}(G)$.

Definition 2.13 (Field Extension). If L is a field containing the subfield K, then L is said to be an *extension field* (or simply *extension*) of K, denoted L/K. The field K is sometimes called the *base field* of the extension.

Definition 2.14 (Degree of an Extension). The *degree* of a field extension L/K, denoted $[L : K]$, is the dimension of L as a vector space over K. The extension is said to be *finite* if $[L : K]$ is finite and is said to be *infinite* otherwise.

Definition 2.15 (Splitting Field). The extension field L of K is called a *splitting field* for the polynomial $f(x) \in K[x]$ if $f(x)$ factors completely into linear factors (or *splits completely*) in $L[x]$ and $f(x)$ does not factor completely into linear factors over any proper subfield of L containing K.

Definition 2.16 (Algebraic Extension). A field extension L/K is called *algebraic* if every element of L is a root of some non-zero polynomial with coefficients in K.

Definition 2.17 (Algebraic Closure). The field $\overline{K}$ is called an *algebraic closure* of K if $\overline{K}$ is algebraic over K and if every polynomial $\in K[x]$ splits completely over $\overline{K}$ (so that $\overline{K}$ can be said to contain all the elements algebraic over K).

Definition 2.18 (Normal Extension). An algebraic field extension L/K is *normal* if every irreducible polynomial with coefficients in K that has at least one root in L factors completely into linear factors in $L[x]$.

Definition 2.19 (Normal Closure)**.** If K is a field and L is an algebraic extension of K, then there is some algebraic extension M of L such that M is a normal extension of K. A minimal subfield (by inclusion) of M which contains L and is a normal extension of K is called a *normal closure* of the extension L over K.

Definition 2.20 (Separable)**.** A polynomial over a field K is called *separable* if it has no multiple roots (i.e., all its roots are distinct). A polynomial which is not separable is called *inseparable.*

A field L is said to *separable* over K if every element of L is a root of a separable polynomial over K. A field which is not separable is *inseparable.*

Definition 2.21 (Automorphism Group of a Field Extension)**.** Let K be a field, and let L/K be an extension of fields.

(1) An automorphism $\sigma \in \operatorname{Aut}(K)$ is said to *fix* an element $\alpha \in K$ if $\sigma(\alpha) = \alpha$. If F is a subset of K, then an automorphism is said to *fix* F if it fixes all of the elements of F.

(2) $\operatorname{Aut}(L/K)$ is the collection of automorphisms of L which fix K. $\operatorname{Aut}(L/K) \leq \operatorname{Aut}(K)$.

Definition 2.22 (Galois Group)**.** Let L/K be a finite extension. Then L is said to be *Galois* over K and L/K is a *Galois extension* if $|\operatorname{Aut}(L/K)| = [L : K]$. If L/K is Galois, the group of automorphisms $\operatorname{Aut}(L/K)$ is called the *Galois group of* L/K, denoted $\operatorname{Gal}(L/K)$.

Definition 2.23 (Galois Group of a Polynomial)**.** If $f(x)$ is a separable polynomial over K, then the *Galois group of* $f(x)$ *over* K denoted $\operatorname{Gal}(f)$ is the Galois group of the splitting field of $f(x)$ over K.

CHAPTER III
ALGORITHMS FOR THE SYMBOLIC COMPUTATION OF ABSOLUTE LINEAR RESOLVENTS

In this chapter we explore the work done by Leonard Soicher in his 1981 thesis [Soi81], which has been extended by this thesis. Soicher developed a method to compute linear resolvents without root approximations by using resultants. Throughout the remainder of this thesis K will be a field unless otherwise stated. In our presentation, we often give polynomials in terms of their roots in an algebraic closure $\overline{K}$ of K (that is, in factored form) (see Definition 2.17). At no point do any of the algorithms given require knowledge of these roots. We only present the polynomials in this form to facilitate understanding. In order to discuss the main algorithm, we need to define three auxiliary functions, MultiplyZeros, SumZeros, and PolyRoot. We denote the output of Algorithm 1 (MultiplyZeros) given the input f and k by MultiplyZeros(f, k) and use analogous notation for all other algorithms. We begin by defining the resolvent polynomial and then discuss some preliminaries involving multisets and each of the auxiliary functions before discussing the main algorithm, Algorithm 4 (LinResolv).

3.1 Resolvents and Resultants

The resolvent is a polynomial whose computation relies on two other polynomials. Informally, the roots of a resolvent polynomial are the result of evaluating variations of a multivariate polynomial at the roots of a second polynomial. The variations of the multivariate polynomial are determined by the action of a group G as defined below.

Definition 3.1. Let $G \leq S_n$ be a permutation group. Let $F(x_1, \ldots, x_n) \in K[x_1, \ldots, x_n]$ be a multivariate polynomial and let $\sigma \in G$. There is a right action of G on $K[x_1, \ldots, x_n]$ defined by

$$F \cdot \sigma = F^{\sigma}(x_1, x_2, \ldots, x_n) = F(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}).$$

Definition 3.2 (G-relative H-invariant Polynomial)**.** Let $F \in K[x_1, \ldots, x_n]$ be a multivariate polynomial and let $H < G \leq S_n$ be permutation groups such that $\mathrm{Stab}_G(F) = H$. (See Definition 3.1 and Definition 2.6). The polynomial F is called a *G-relative, H-invariant polynomial.*

Definition 3.3 (Resolvent Polynomial)**.** Let $H < G \leq S_n$, and $F \in K[x_1, \ldots, x_n]$, a G-relative, H-invariant polynomial. Let $f \in K[x]$ of degree n such that the roots of f in an algebraic closure of K are $\alpha_1, \ldots, \alpha_n$. Then, the resolvent polynomial $R(G, H, F, f)$ is

$$R(G, H, F, f) = \prod_{\sigma \in G//H} \left(x - F\left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}\right)\right)$$

- If $G = S_n$, then $R(G, H, F, f)$ is called an *absolute resolvent*, if G is a proper subgroup of S_n, then $R(G, H, F, f)$ is called a *relative resolvent.*
- The resolvent polynomial $R(G, H, F, f)$ is called a *linear resolvent* if $F(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n$ for some $a_1, \ldots, a_n \in K$.

In order to compute the resolvent without the need for root approximations, Soicher uses resultants. We now discuss the resultant and its computation.

Definition 3.4 (Resultant)**.** Let $\overline{K}$ be an algebraic closure of K. Let $f, g \in K[x]$ with $g \neq f$ and

$$f(x) = \sum_{i=0}^{m} f_i x^i = f_m \prod_{i=1}^{m} (x - \alpha_i)$$

for $\alpha_i \in \overline{K}$, $1 \leq i \leq m$. Then the resultant of f and g with respect to x is

$$\mathrm{Res}_x(f, g) = \begin{cases} f_m^{\deg g} \prod\limits_{i=1}^{m} g(\alpha_i) & \text{if } m \geq 1 \\ f_0^{\deg g} & \text{if } m = 0 \end{cases}$$

The x in Res_x is necessary because we also consider the resultant with respect to multivariate polynomials, in which case the output is a polynomial instead of a value in K.

Example 3.5. Let $f, g \in K[x]$ be monic polynomials with $f(x) = \prod\limits_{i=1}^{m}(x - \alpha_i)$ and $g(x) = \prod\limits_{i=1}^{n}(x - \beta_i)$ as in Definition 3.4 and consider

$$\mathrm{Res}_y(f(y), g(x - y)) = \prod_{i=1}^{m} g(x - \alpha_i) = \prod_{j=1}^{n} \prod_{i=1}^{m} (x - (\alpha_i + \beta_j))$$

So that $\mathrm{Res}_y(f(y), g(x - y))$ is a polynomial in x of degree mn. This example is important for the auxiliary functions that we discuss in the next section.

In order to compute the resultant without the use of root approximations, we refer to the following proposition. A proof can be found in [Coh93].

Proposition 3.6. *Let* $f, g \in K[x]$ *with* $f(x) = \sum_{i=0}^{m} f_i x^i$ *and* $g(x) = \sum_{i=0}^{n} g_i x^i$ *then* $\mathrm{Res}_x(f,g) = \det(\mathrm{Syl}(f,g))$. $\mathrm{Syl}(f,g)$ *is the Sylvester matrix of* f *and* g *given by*

$$\begin{bmatrix}
f_m & f_{m-1} & f_{m-2} & \dots & f_1 & f_0 & 0 & 0 & \dots & 0 \\
0 & f_m & f_{m-1} & f_{m-2} & \dots & f_1 & f_0 & 0 & \dots & 0 \\
0 & 0 & f_m & f_{m-1} & f_{m-2} & \dots & f_1 & f_0 & \dots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \dots & 0 & f_m & f_{m-1} & f_{m-2} & \dots & f_1 & f_0 \\
g_n & g_{n-1} & \dots & g_2 & g_1 & g_0 & 0 & 0 & \dots & 0 \\
0 & g_n & g_{n-1} & \dots & g_2 & g_1 & g_0 & 0 & \dots & 0 \\
0 & 0 & g_n & g_{n-1} & \dots & g_2 & g_1 & g_0 & \dots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \dots & 0 & g_n & g_{n-1} & \dots & g_2 & g_1 & g_0
\end{bmatrix}$$

where the coefficients of f *are repeated on* $n = \deg(g)$ *rows and the coefficients of* g *are repeated on* $m = \deg(f)$ *rows.*

This shows that we do not need root approximations to compute the resultant and this allows the computation of the absolute resolvent without root approximations as well.

3.2 Auxiliary Functions

Three auxiliary functions are used in the computation of the absolute resolvent. Algorithm 1 (MultiplyZeros) and Algorithm 2 (SumZeros) create new polynomials from input polynomials. Algorithm 1 (MultiplyZeros) takes as input a polynomial and a scalar in the base field and returns the polynomial whose roots are the roots of the input polynomial multiplied by the scalar. Algorithm 2 (SumZeros) takes two

Algorithm 1. (MultiplyZeros)**.** Returns a polynomial whose roots are the roots of the input polynomial scaled by the second input.

```
Input:   $f(x) = a\prod_{i=1}^{n}(x-\alpha_i) \in K[x]$, $r \in K$.
Output:  $g(x) = a\prod_{i=1}^{n}(x-r\alpha_i) \in K[x]$.
(1) return $r^n f\left(\frac{x}{r}\right)$.
```

Algorithm 2. (SumZeros)**.** Returns the polynomial whose roots are the sums of the roots of the input polynomials.

```
Input:   $f(x) = a\prod_{i=1}^{n}(x-\alpha_i) \in K[x]$, $g(x) = b\prod_{i=1}^{m}(x-\beta_i) \in K[x]$.
Output:  $s(x) = a^m b^n \prod_{i=1}^{n}\prod_{j=1}^{m}(x-(\alpha_i+\beta_j)) \in K[x]$.
(1) return $\mathrm{Res}_y(f(y), g(x-y))$.
```

polynomials as input and returns the polynomial whose roots are the sums of the roots of the input polynomials by using resultants (see Definition 3.4 and Example 3.5). Algorithm 3 (PolyRoot) is used to decrease the multiplicity of the roots of a polynomial so that the resolvent has the correct degree. In particular, Algorithm 3 (PolyRoot) is used when the same coefficient in the G-relative H-invariant polynomial, F, is repeated for multiple variables, which causes Algorithm 4 (LinResolv) to include multiple copies of roots in the computation of the resolvent that need to be removed.

Algorithm 3. (PolyRoot)**.** Returns the polynomial f such that $u = f^k$ given a polynomial u and $k \in \mathbb{N}$.

```
Input:   $u(x) \in K[x]$ and $k \in \mathbb{N}$, such that $u(x) = f(x)^k$ for some $f(x) \in K[x]$.
Output:  $f(x) \in K[x]$.

(1) if $k = 1$ then return $u(x)$.
(2) $t(x) \leftarrow u(x)/\gcd(u, u')$
    (Note: $u'(x)$ is the formal derivative of $u(x)$ and the zeros of $t(x)$ are precisely
    the distinct zeros of $u(x)$.)
(3) $r(x) \leftarrow t(x)$
    $s(x) \leftarrow u(x)$
(4) while $\deg(r) < (\deg(u))/k$
    (a) $s(x) \leftarrow s(x)/t(x)^k$
    (b) $t(x) \leftarrow \gcd(s, t)$
    (c) $r(x) \leftarrow t(x)r(x)$
(5) return $r(x)$
```

Algorithm 3 (PolyRoot) computes a polynomial f given a polynomial u and a $k \in \mathbb{N}$ such that $u = f^k$. First, the algorithm constructs the polynomial which contains each root of f exactly once by dividing $u = f^k$ by the greatest common divisor of u and its derivative. Then the algorithm builds up the polynomial until it reaches the required degree.

Example 3.7. We compute PolyRoot$((x-1)^4(x-2)^2, 2)$.
Note that $u(x) = (x-1)^4(x-2)^2$ is given here in factored form for demonstration only. The algorithm never requires the input in factored form.

Input: $u(x) = (x-1)^4(x-2)^2$, $k = 2$

(1) $k = 2$ so proceed.

(2) $t(x) \leftarrow u(x)/\gcd(u, u') = (x-1)(x-2)$

(3) $r(x) \leftarrow t(x) = (x-1)(x-2)$
$s(x) \leftarrow u(x) = (x-1)^4(x-2)^2$

(4) $\deg(r) = 2 < 3 = (\deg(u))/k$ so enter the loop.

(a) $s(x) \leftarrow s(x)/t(x)^k = ((x-1)^4(x-2)^2)/((x-1)(x-2))^2 = (x-1)^2$

(b) $t(x) \leftarrow \gcd(s, t) = (x-1)$

(c) $r(x) \leftarrow t(x)r(x) = (x-1)((x-1)(x-2)) = (x-1)^2(x-2)$

(4) $\deg(r) = 3 \not< 3 = (\deg(u))/k$ so do not enter the loop.

(5) return $r(x) = (x-1)^2(x-2)$

3.3 Multiset Operations

We describe the coefficients of the multivariate polynomial used for the resolvent as a multiset. Informally, a multiset is a generalization of a set where the elements of the set may be repeated. The number of occurrences of an element in a multiset is called the multiplicity of the element.

Definition 3.8 (Multiset)**.** A *multiset* is a pair (A, m) consisting of a set A and a function $m\colon A \to \mathbb{N}_{\geq 1} = \{1, 2, 3, \dots\}$. For an element $x \notin A$, we set $m(x) = 0$. The set A is called the underlying set of elements. For each a_i in A the *multiplicity* of a_i is the number $m(a_i) = m_i$. We denote (A, m) by $\{a_1^{\times m_1}, \dots, a_n^{\times m_n}\}$ where $A = \{a_1, \dots, a_n\}$.

Definition 3.9. Given a multiset $\mathcal{L} = (A, m)$ and $a \in A$, define $\operatorname{mult}(a, \mathcal{L}) = m(a)$. So that $\operatorname{mult}(a, \mathcal{L})$ is the multiplicity of a in the multiset $(A, m) = \mathcal{L}$. If $a \notin A$, then $\operatorname{mult}(a, \mathcal{L}) = 0$.

We define the operations $+$ and $-$ on multisets as follows.

Definition 3.10 (Multiset Sum)**.** For multisets (A, m) and (B, n) define $(A, m) + (B, n)$ as follows. Let $A \cup B = C = \{c_1, \ldots, c_k\}$. Then

$$(A, m) + (B, n) = \{c_1^{\times(m(c_1)+n(c_1))}, \ldots, c_k^{\times(m(c_k)+n(c_k))}\}.$$

Definition 3.11 (Multiset Difference)**.** For multisets (A, m) and (B, n) define $(A, m) - (B, n)$ as follows. Let $A = \{a_1, \ldots, a_k\}$.

$$(A, m) - (B, n) = \{a_1^{\times \max(m(a_1)-n(a_1),0)}, \ldots, a_k^{\times \max(m(a_k)-n(a_k),0)}\}$$

If $\max(m(a_i) - n(a_i), 0) = 0$ then a_i is not an element of $(A, m) - (B, n)$.

Definition 3.12 (Size of a Multiset)**.** Let $(A, m) = \{a_1^{\times m_1}, \ldots, a_k^{\times m_k}\}$ be a multiset and define $\#(A, m) = \sum_{i=1}^{k} m_i$ to be the size of (A, m).

3.4 LinResolv Algorithm

Algorithm 4 (LinResolv) computes the absolute linear resolvent, $R(G, H, F, f)$ as in Definition 3.3, where $G = S_n$ and $H = \operatorname{Stab}_G(F)$. We first describe how $\operatorname{Stab}_{S_n}(F)$ can be determined by the coefficients of F for any F of the form $F(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_nx_n$.

Proposition 3.13. *Let F be a multivariate polynomial such that $F(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_nx_n$ with coefficients $c_i \in K$ for $1 \leq i \leq n$. Let $\{a_1, \ldots, a_k\} \subset K$ be the* *set of distinct coefficients of F and let $\{a_1^{\times m_1}, \ldots, a_k^{\times m_k}\}$ be the multiset representation of the coefficients of F. Then* $\mathrm{Stab}_{S_n}(F) \cong S_{m_1} \times \cdots \times S_{m_k}$.

Proof. This follows directly from the definition of the action of S_n on $K[x_1, \ldots, x_n]$ described in Definition 3.1. □

Algorithm 4 (LinResolv) computes the resolvent polynomial recursively on the number of coefficients in the invariant multivariate polynomial. The algorithm computes the resolvent by using resultants to create polynomials and dividing out extra factors by computing other resolvents. For a proof of the algorithm see [Soi81].

3.5 Example using Algorithm 4 (LinResolv)

Although the algorithm does not use root approximations for the computations, showing what is happening in terms of the roots is instructive. Let $f \in K[x]$ such that $f(x) = \prod_{i=1}^{n}(x - \alpha_i)$ and $F \in K[x_1, \ldots, x_n]$ such that $F(x_1, \ldots, x_n) = x_1 + x_2 + 2x_3 + 0x_4 + \cdots + 0x_n$. Then $\mathcal{M} = \{1^{\times 2}, 2^{\times 1}\}$ is the multiset representing F. In Example 3.14 we compute $R(S_n, S_2 \times S_1 \times S_{n-3}, F, f)$ by computing LinResolv$(\mathcal{M}, f)$ using Algorithm 4 (LinResolv).

Example 3.14. We compute $R(S_n, S_2 \times S_1 \times S_{n-3}, F, f)$ using Algorithm 4(LinResolv). The numbered steps indicate the level of the recursion and match the numbers in Algorithm 4.

Input: $\mathcal{M} = \{1^{\times 2}, 2^{\times 1}\}, f(x) = \prod_{i=1}^{n}(x - \alpha_i)$

(1) $\#\mathcal{M} = 3$ so proceed.

(2) $\overline{\mathcal{M}} \leftarrow \{1^{\times 2}\}$

Algorithm 4. (LinResolv). Returns the absolute linear resolvent $R(G, H, F, f)$ where $G = S_n$ and $H = \mathrm{Stab}_{S_n}(F)$.

Input: A polynomial $f \in K[x]$ of degree n, and a multiset $\mathcal{M} = (A, m) = \{a_1^{\times m_1}, \ldots, a_k^{\times m_k}\}$, which is the multiset representation of the sequence $[c_1, \ldots, c_r]$ whose entries are the nonzero coefficients of F, a multivariate polynomial of the form

$$F(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_rx_r + 0x_{r+1} + \ldots + 0x_n$$

with $r \leq n$ and $c_i \in K$ and $c_i \neq 0$ for $1 \leq i \leq r$.

Output: The resolvent $R(S_n, \mathrm{Stab}_{S_n}(F), F, f)$ with F as defined in the input.

(1) If $\#\mathcal{M} = 1$, then return MultiplyZeros(a_1, f).
(Stop condition for the recursion: there is only one nonzero coefficient in F.)

(2) $\overline{\mathcal{M}} \leftarrow \mathcal{M} - \{a_k^{\times 1}\} = (B, h) = \{b_1^{\times h_1}, \ldots, b_j^{\times h_j}\}$
(Remove the last coefficient from F.)

(3) $u(x) \leftarrow$ LinResolv$(\overline{\mathcal{M}}, f)$
(Recursively call this algorithm with F having one fewer coefficient.)

(4) $s(x) \leftarrow \prod_{i=1}^{j}$ LinResolv$(\mathcal{M}_i, f)^{\mathrm{mult}(b_i + a_k, \mathcal{M}_i)}$,
where $\mathcal{M}_i = (\overline{\mathcal{M}} - \{b_i^{\times 1}\}) + \{(b_i + a_k)^{\times 1}\}$.
See Definition 3.10 and Definition 3.11.
(Compute the extra factors that occur from the use of the resultant.)

(5) $d \leftarrow \mathrm{mult}(a_k, \mathcal{M})$
(Compute the number of extra factors resulting from the multiplicity of the last coefficient of F.)

(6) $g(x) \leftarrow$ MultiplyZeros(a_k, f)
(Compute the new polynomial to be included using the last coefficient of F.)

(7) $t(x) \leftarrow$ SumZeros$(u(x), g(x))/s(x)$
(Compute the new polynomial by "summing" the roots of the new factor with the recursive result and dividing out the extra factors.)

(8) return PolyRoot$(t(x), d)$
(Remove the extra multiplicities of roots.)

(3) $u(x) \leftarrow \text{LinResolv}(\{1^{\times 2}\}, f)$

Computation of $\text{LinResolv}(\{1^{\times 2}\}, f)$:

(3.1) $\#\mathcal{M} = 2$ so proceed.

(3.2) $\overline{\mathcal{M}} \leftarrow \{1^{\times 1}\}$

(3.3) $u(x) \leftarrow \text{LinResolv}(\overline{\mathcal{M}} = \{1^{\times 1}\}, f)$

Computation of $\text{LinResolv}(\{1^{\times 1}\}, f)$:

(3.3.1) $\#\mathcal{M} = 1$ so return $\text{MultiplyZeros}(1, f) = \prod_{i=1}^{n}(x - \alpha_i)$

(3.3) Now we have $u(x) = \prod_{i=1}^{n}(x - \alpha_i)$

(3.4) $s(x) \leftarrow (\text{LinResolv}(\mathcal{M}_1 = \{(1+1)^{\times 1}\}, f))^1$

Computation of $\text{LinResolv}(\{2^{\times 1}\}, f)$:

(3.4.1) $\#\mathcal{M} = 1$ so return $\text{MultiplyZeros}(2, f) = \prod_{i=1}^{n}(x - 2\alpha_i)$

(3.4) Now we have $s(x) = \prod_{i=1}^{n}(x - 2\alpha_i)$

(3.5) $d \leftarrow \text{mult}(1, \mathcal{M} = \{1^{\times 2}\}) = 2$

(3.6) $g(x) \leftarrow \text{MultiplyZeros}(1, f) = \prod_{i=1}^{n}(x - \alpha_i)$

(3.7) Computation of $t(x)$:

$$t(x) \leftarrow \text{SumZeros}(u(x), g(x))/s(x)$$
$$= \text{Res}_y(u(y), g(x-y))/s(x)$$
$$= \left(\prod_{i=1}^{n} g(x-\alpha_i)\right) \Big/ \left(\prod_{i=1}^{n}(x-2\alpha_i)\right)$$
$$= \left(\prod_{\substack{1\le i\le n\\1\le j\le n}} (x-(\alpha_i+\alpha_j))\right) \Big/ \left(\prod_{i=1}^{n}(x-2\alpha_i)\right)$$
$$= \prod_{\substack{1\le i,j\le n\\ i\ne j}} (x-(\alpha_i+\alpha_j))$$

(3.8) return PolyRoot$(t(x), d)$

$$= \text{PolyRoot}\left(\prod_{\substack{1\le i,j\le n\\ i\ne j}} (x-(\alpha_i+\alpha_j)), 2\right)$$
$$= \sqrt{\prod_{\substack{1\le i,j\le n\\ i\ne j}} (x-(\alpha_i+\alpha_j))}$$
$$= \prod_{1\le i<j\le n} (x-(\alpha_i+\alpha_j))$$

(3) Now we have $u(x) = \prod_{1\le i<j\le n} (x-(\alpha_i+\alpha_j))$

(4) $s(x) \leftarrow (\text{LinResolv}(\{1^{\times 1}, (1+2)^{\times 1}\}, f))^1$

Computation of LinResolv$((\{1^{\times 1}, 3^{\times 1}\}, f)$:

(4.1) $\#\mathcal{M} = 2$ so proceed

(4.2) $\overline{\mathcal{M}} \leftarrow \{1^{\times 1}\}$

(4.3) $u(x) \leftarrow \text{LinResolv}(\{1^{\times 1}\}, f) = \prod_{i=1}^{n}(x-\alpha_i)$

(4.4) $s(x) \leftarrow (\text{LinResolv}(\{(1+3)^{\times 1}\}, f)^1 = \prod_{i=1}^{n}(x-4\alpha_i)$

(4.5) $d \leftarrow \text{mult}(3, \{1^{\times 1}, 3^{\times 1}\}) = 1$

(4.6) $g(x) \leftarrow \text{MultiplyZeros}(3, f) = \prod_{i=1}^{n}(x - 3\alpha_i)$

(4.7) Computation of $t(x)$:

$$\begin{aligned} t(x) &\leftarrow \text{SumZeros}(u(x), g(x))/s(x) \\ &= \text{Res}_y(u(y), g(x-y))/s(x) \\ &= \left(\prod_{i=1}^{n} g(x-\alpha_i)\right) \Big/ \left(\prod_{i=1}^{n}(x - 4\alpha_i)\right) \\ &= \left(\prod_{\substack{1\le i\le n \\ 1\le j\le n}} (x - (\alpha_i + 3\alpha_j))\right) \Big/ \left(\prod_{i=1}^{n}(x - 4\alpha_i)\right) \\ &= \prod_{\substack{1\le i,j\le n \\ i\ne j}} (x - (\alpha_i + 3\alpha_j)) \end{aligned}$$

(4.8) return $\text{PolyRoot}(t(x), 1) = \prod_{\substack{1\le i,j\le n \\ i\ne j}} (x - (\alpha_i + 3\alpha_j))$

(4) Now we have $s(x) = \prod_{\substack{1\le i,j\le n \\ i\ne j}} (x - (\alpha_i + 3\alpha_j))$

(5) $d \leftarrow \text{mult}(2, \{1^{\times 2}, 2^{\times 1}\}) = 1$

(6) $g(x) \leftarrow \text{MultiplyZeros}(2, f) = \prod_{i=1}^{n}(x - 2\alpha_i)$

(7) Computation of $t(x)$:

$$\begin{aligned}
t(x) &\leftarrow \mathrm{SumZeros}(u(x), g(x))/s(x) \\
&= \mathrm{Res}_y(u(y), g(x-y))/s(x) \\
&= \left(\prod_{1\le i<j\le n} g(x-(\alpha_i+\alpha_j)) \right) \Bigg/ \left(\prod_{\substack{1\le i,j\le n \\ i\ne j}} (x-(\alpha_i+3\alpha_j)) \right) \\
&= \left(\prod_{\substack{1\le i<j\le n \\ 1\le k\le n}} (x-(2\alpha_k+\alpha_i+\alpha_j)) \right) \Bigg/ \left(\prod_{\substack{1\le i,j\le n \\ i\ne j}} (x-(\alpha_i+3\alpha_j)) \right) \\
&= \prod_{\substack{1\le i<j\le n \\ 1\le k\le n \\ k\ne i,j}} (x-(2\alpha_k+\alpha_i+\alpha_j))
\end{aligned}$$

(8) return $\mathrm{PolyRoot}(t(x), 1) = \prod_{\substack{1\le i<j\le n \\ 1\le k\le n \\ k\ne i,j}} (x-(2\alpha_k+\alpha_i+\alpha_j))$

3.6 Implementation Extensions and Computational Example

We extend Algorithm 2 (SumZeros) in a natural way to successively compute the desired resultant for a sequence of inputs in Algorithm 5 (SumZerosExt).

Using this extended algorithm, we can change the recursion to alter the order and number of operations. Considering steps (3), (4), (6) and (7) of Algorithm 4 (LinResolv), we can compute the entire numerator and denominator of step (7) before the division instead of computing the division at each recursive step. This approach has not been implemented at this time, since it is unclear if it would be a computational improvement as the degrees of the polynomials being divided could become extremely large.

Algorithm 5. (SumZerosExt)**.** Successively calls Algorithm 2 (SumZeros) on a sequence of polynomials.

```
Input:   A sequence $[f_1, \ldots, f_n]$ of polynomials $f_i \in K[x]$
Output:  A polynomial $g \in K[x]$ such that the roots of $g$ are the sums of the
         roots of the polynomials in the sequence
(1) $g \leftarrow f_1$
(2) for $i = 2$ to $n$ do
    (a) $g \leftarrow \text{SumZeros}(g, f_i)$
(3) return $g$
```

Example 3.15. Let $\mathcal{M} = \{1^{\times 3}, 2\}$ be the multiset representation of the multivariate invariant polynomial, and let $f = x^7 + 3x^3 + 4x^2 + 7x + 11$. Then $\text{LinResolv}(\mathcal{M}, f) =$

$$\begin{aligned}
x^{35} - 1608x^{31} - 3720x^{30} - 11480x^{29} - 124542x^{28} + 1190934x^{27}+ \\
4788600x^{26} + 23730880x^{25} + 387395658x^{24} + 1281458350x^{23}+ \\
9929194794x^{22} + 46950037084x^{21} - 521628779362x^{20}- \\
3983735495983x^{19} - 33159243626780x^{18} - 223227162404084x^{17}- \\
389788917736212x^{16} + 329004097616786x^{15} + 2395843935796812 8x^{14}+ \\
320609240851560516x^{13} + 1808067075090694802x^{12} + 12514704387593949725x^{11}+ \\
71368259108581999048x^{10} + 149168426005968199140x^{9} - 579412026047441768534x^{8}- \\
1872379046553154614109 3x^{7} - 1984467792852011146520 96x^{6}- \\
1206615884365471337508848x^{5} - 6287804695563776500883816x^{4}- \\
17366032191353538639518784x^{3} + 79217323577733757125489632x^{2}+ \\
129720644195688789442824819 2x + 126417879134427486692720643 84
\end{aligned}$$

CHAPTER IV
ALGORITHM FOR THE SYMBOLIC COMPUTATION OF RELATIVE LINEAR RESOLVENTS

In this chapter we explain an algorithm to compute relative linear resolvents, $R(G,H,F,f)$, where $G = S_m \wr S_l$, the wreath product of S_m and S_l, and $H = \mathrm{Stab}_G(F)$. We first define and discuss wreath products of groups, a type of semidirect product. We continue to use right group actions in this chapter.

4.1 Wreath Products

Definition 4.1 (Semidirect Product)**.** Let G and H be groups and let φ be a homomorphism from H into $\mathrm{Aut}(G)$. Then the *semidirect product of* G *and* H denoted $G \rtimes_\varphi H$ is a group with respect to the following operation

$$(g_1,h_1)(g_2,h_2) = (g_1(\varphi(h_1)(g_2)), h_1h_2)$$

on ordered pairs of elements of G and H.

A wreath product $G \wr H$ of two permutation groups (G,X) and (H,Y) is itself a permutation group on the set $X \times Y$. The action of the wreath product on $X \times Y$ can best be described if we consider $X \times Y$ as $\bigcup_{y\in Y} X \times \{y\}$. So that, loosely, an element of the wreath product can be thought of as first permuting the element of Y as the group H does and then permuting the copy of X in $X \times \{y\}$ in the way that the group G does. In fact, $\{X \times \{y\} \mid y \in Y\}$ is a block system for $G \wr H$ (see Definition 2.8).

In order to properly define a wreath product as a semidirect product, some concepts are needed in advance of the wreath product definition. We are only defining wreath products for permutation groups on finite sets. For a more thorough discussion, see [Mel95].

Let (G, X) and (H, Y) be permutation groups on finite sets X and Y, and let G^Y be the set of all maps from Y to G. Then G^Y is a group with respect to the following operation. Let $f, g \in G^Y$, $fg \in G^Y$ is defined by $(fg)(y) = f(y)g(y)$. We define an action of H on G^Y as follows. Let $f \in G^Y$ and note that $f(y) \in G$ for all $y \in Y$. Then $h \in H$ acts on G^Y by

$$(f \cdot h)(y) = f^h(y) = f(yh^{-1})$$

Since $h \in H$, $yh^{-1} \in Y$. For each $h \in H$, we define $\psi^h \in \mathrm{Aut}(G^Y)$ by $f \mapsto f^h$.

Definition 4.2 (Wreath Product)**.** Let (G, X) and (H, Y) be permutation groups with X and Y finite sets. Let $\varphi : H \to \mathrm{Aut}(G^Y)$ be defined by $\varphi(h) = \psi^h$ as defined above. Then the *(permutational) wreath product* of G and H denoted by $G \wr H$ is the semidirect product, $G^Y \rtimes_\varphi H$.

Definition 4.3 (Imprimitive Action of $G \wr H$)**.** Let (G, X) and (H, Y) be permutation groups with X and Y finite sets. $G \wr H$ acts imprimitively on $X \times Y$ as follows

$$(x, y) \cdot (f, h) = (xf^h(y), yh) = (xf(yh^{-1}), yh) \quad \text{for } f \in G^Y, h \in H.$$

$\{X \times \{y\} \mid y \in Y\}$ is a block system of the permutation group $(G \wr H, X \times Y)$.

4.2 Computation of $\mathrm{Stab}_G(F)$

Just as with the computation of absolute linear resolvents, the subgroup $H = \mathrm{Stab}_G(F)$ can be completely determined just by examining F for the computation of relative linear resolvents. We are only computing relative resolvents with respect to wreath products; however, the computation of the stabilizer discussed in this section is relevant for computing relative resolvents with respect to any group G. First, note the following remark.

Remark. Let $G \leq S_n$. Then $\mathrm{Stab}_G(F) \cong G \cap \mathrm{Stab}_{S_n}(F)$.

In the computation of the absolute linear resolvent, the ordering of the coefficients does not affect the stabilizer of F. However, when computing the relative resolvent, the ordering of the coefficients does affect the computation of the resolvent and the subgroup H. In order to correctly compute $\mathrm{Stab}_G(F)$, the direct product in Proposition 3.13 must be correctly embedded in S_n, where n is the degree of f. Of course the ordering of the coefficients of the G-relative H-invariant polynomial F and the ordering of the roots of the polynomial f should be considered in tandem. So any embedding of the stabilizer used for the computation of a relative resolvent should take into account the ordering of the roots of the polynomial f as well.

Let $\mathrm{Stab}_G(F) \cong S_{k_1} \times \cdots \times S_{k_m} =: H$. In order to correctly compute this embedding, we must construct a homomorphism $\phi : H \to S_n$ by mapping the standard generators of the direct product to permutations determined by the indices of the variables associated with the respective coefficients.

Example 4.4. We construct the stabilizer $\mathrm{Stab}_{S_{16}}(F)$ of

$F(x_1, \ldots, x_{16}) = x_1 + x_5 + 2x_6 + 2x_7 + x_8 + x_{10} + 2x_{11} + 3x_{14} + 3x_{16}$

By Proposition 3.13, $\mathrm{Stab}_{S_{16}}(F) \cong S_4 \times S_3 \times S_2 \times S_7 =: H$.

Define $\phi : H \to S_{16}$ by

S_4	$(1\ 2)$	$\mapsto$	$(1\ 5)$	$(1\ 2\ 3\ 4)$	$\mapsto$	$(1\ 5\ 8\ 10)$
S_3	$(5\ 6)$	$\mapsto$	$(6\ 7)$	$(5\ 6\ 7)$	$\mapsto$	$(6\ 7\ 11)$
S_2	$(8\ 9)$	$\mapsto$	$(14\ 16)$			
S_7	$(10\ 11)$	$\mapsto$	$(2\ 3)$	$(10\ 11\ 12\ 13\ 14\ 15\ 16)$	$\mapsto$	$(2\ 3\ 4\ 9\ 12\ 13\ 15)$

The correct embedding of $\mathrm{Stab}_{S_{16}}(F)$ is $\phi(H)$.

For the purpose of quickly and easily computing this embedding, a function was written to determine this group directly from the sequence of coefficients of F. The constructed group could then be intersected with the desired group G.

4.3 RLRSetup and RelLinResolv Algorithms

Algorithm 7 (RelLinResolv) computes the relative linear resolvent for a separable, irreducible polynomial $f \in K[x]$ by using a factorization of f in a normal extension T of K. We assume that f factors into l distinct factors each of degree m in T (Proposition 5.1 in Chapter V shows that this indeed is the case). Given these conditions, Algorithm 7 (RelLinResolv) computes the relative resolvent with respect to $G = S_m \wr S_l$. The algorithm computes the absolute resolvents of each factor of f using Algorithm 4 (LinResolv) and produces the polynomial whose roots are the sums of the roots of these resolvents using Algorithm 2 (SumZeros). Unlike the computation of the absolute resolvent, no extra factors are produced from the use of the resultant in this step because the factors of f do not share any roots.

Algorithm 7 (RelLinResolv) requires some setup to modify the sequence of coefficients of the multivariate polynomial F; so, we first describe Algorithm 6 (RLRSetup), which prepares the multiset required as the input for Algorithm 7 (RelLinResolv).

Algorithm 6 (RLRSetup) partitions the sequence of coefficients by the degree of the factors of f in T and then removes any zeros which served as place holders in the invariant F. Lastly, Algorithm 6 (RLRSetup) returns the multiset representation of this partition which can be sent to Algorithm 7 (RelLinResolv).

4.4 Proof of Algorithm 7 (RelLinResolv)

Theorem 4.5. *Let $K \subset L$ be a tower of field extensions and let $L \cong K[x]/(f)$ for an irreducible, separable polynomial $f \in K[x]$. Let T be a normal extension of K such that f factors as $\prod_{i=1}^{l} f_i$ over $T[x]$, with $\deg(f_i) = m$ for $1 \leq i \leq l$, and denote by $\alpha_i^{(1)}, \ldots, \alpha_i^{(m)}$ the roots of f_i in an algebraic closure $\overline{K}$ of K. Let $\Phi = [f_1, \ldots, f_l]$, and let $F(x_1, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_sx_s + 0x_{s+1} + \cdots + 0x_n$ be a multivariate polynomial and C the sequence of coefficients, $[c_1, \ldots, c_s]$. Let $\mathcal{M}$ be the multiset constructed by* RLRSetup(C, m). *Let $G = S_m \wr S_l$ and $H = \mathrm{Stab}_G(F)$. Let the roots of the factors of f be ordered such that $\{\{\alpha_i^{(1)}, \ldots, \alpha_i^{(m)}\} \mid 1 \leq i \leq l\}$ forms a block system for G under the action of G on the indices analogous to Definition* 3.1. *Then* RelLinResolv$(\mathcal{M}, \Phi)$ *computed by Algorithm* 7 (RelLinResolv) *constructs the relative linear resolvent $R(G, H, F, f)$.*

Proof. Let $\alpha_1, \ldots, \alpha_n$ denote the roots of f in $\overline{K}$. We reindex the set $\{\alpha_1, \ldots, \alpha_n\}$ as $\{\alpha_{(1,1)}, \ldots, \alpha_{(m,l)}\}$ where $\alpha_{(i,j)} = \alpha_j^{(i)}$ is a root of f_j. Note that since we are computing an absolute resolvent for each f_j, the root ordering in each polynomial f_j does not affect the computation; however, we can take $i = \lceil \frac{n}{m} \rceil$. For clarity, we denote $\alpha_{(i,j)}$ as $\alpha_j^{(i)}$ where appropriate so that the action of the group is always on the lower index, as defined in Definition 3.1.

We proceed by induction on the size of $\mathcal{M}$.

Base case: $\#\mathcal{M} = 1$.

If $\#\mathcal{M} = 1$, then $\mathrm{RelLinResolv}(\mathcal{M}, \Phi)$ returns $\prod_{i=1}^{l} \mathrm{LinResolv}(M_1, f_i)$. We want to show that this is $R(G, H, F, f)$. Let $M_1 = \{a_1^{\times m_1}, \ldots, a_k^{\times m_k}\}$. Since $\#\mathcal{M} = 1$, without loss of generality, we assume that the coefficients of x_{m+1} through x_n in F are all zero. Since c_i is the coefficient of x_i in F, let

$$F_1(x_1, \ldots, x_m) = \sum_{i=1}^{m} c_i x_i$$

In other words, $F_1(x_1, \ldots, x_m) \equiv F(x_1, \ldots, x_n)$ under equivalence of functions.

Recalling that $G = S_m \wr S_l$ has block system $\{\{1, \ldots, m\} \times \{i\} \mid i \in \{1, \ldots, l\}\}$ (see Definition 4.3), we have

$$\begin{aligned}
\mathrm{RelLinResolv}(\Phi, \mathcal{M}) &= \prod_{i=1}^{l} \mathrm{LinResolv}(M_1, f_i) \\
&= \prod_{i=1}^{l} \prod_{\tau \in S_m // H \cap S_m} (x - F_1^{\tau}(\alpha_1^{(i)}, \ldots, \alpha_m^{(i)})) \\
&= \prod_{i=1}^{l} \prod_{\substack{\sigma \in S_m \wr S_L // H \\ \sigma \in \mathrm{Stab}_G(i)}} (x - F^{\sigma}(\alpha_{(1,i)}, \ldots, \alpha_{(m,i)}, 0, \ldots, 0)) \\
&= \prod_{\sigma \in S_m \wr S_L // H} (x - F^{\sigma}(\alpha_{(1,1)}, \ldots, \alpha_{(m,l)})) \\
&= R(G, H, F, f)
\end{aligned}$$

This proves the base case.

Inductive step: Assume that $\mathrm{RelLinResolv}(\mathcal{M}, \Phi) = R(G, H, F, f)$ when $\#\mathcal{M} = t$. We need to show that $\mathrm{RelLinResolv}(\mathcal{M}, \Phi) = R(G, H, F, f)$ when $\#\mathcal{M} = t + 1$.

Let $\mathcal{M} = \{M_1^{\times m_1}, \ldots, M_k^{\times m_k}\}$. Note that the polynomial F determines $\mathcal{M}$ in Algorithm 6 (RLRSetup) and we can assume that the coefficients of $x_{n-tm+1}, \ldots, x_{n-tm+m}$ in F correspond to the multiset M_k in $\mathcal{M}$.

Define $\overline{F} = F(x_1, \ldots, x_{n-tm}, 0, \ldots, 0)$ and $F_{t+1} = F(0, \ldots, 0, x_{n-tm+1}, \ldots, x_{n-tm+m}, 0 \ldots, 0)$. As before, since $\#\mathcal{M} = t+1$, we know that

$$\begin{aligned} F(x_1, \ldots, x_n) &= F(x_1, \ldots, x_{n-tm+m}, 0, \ldots, 0) \\ &= F(x_1, \ldots, x_{n-tm}, 0, \ldots, 0) + F(0, \ldots, 0, x_{n-tm+1}, \ldots, x_{n-tm+m}, 0, \ldots, 0) \\ &= \overline{F} + F_{t+1} \end{aligned}$$

We also know that $R(G, H, F_{t+1}, f) = \prod_{i=1}^{l} \text{LinResolv}(M_k, f_i)$ by our definition of f and F_{t+1}. By our inductive assumption, we have that $\text{RelLinResolv}(\overline{\mathcal{M}}, \Phi_i)$ computes the desired resolvent for $1 \leq i \leq l$, where $\Phi_i = [f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_l]$ as in

Algorithm 7 (RelLinResolv). Then we have the following.

$$\begin{aligned}
&\mathrm{RelLinResolv}(\Phi, \mathcal{M}) \\
=&\prod_{i=1}^{l} \mathrm{SumZeros}(\mathrm{RelLinResolv}(\Phi_i, \overline{M}), \mathrm{LinResolv}(M_k, f_i)) \\
=&\prod_{i=1}^{l} \mathrm{SumZeros}\Bigg(\prod_{\substack{\sigma \in S_m \wr S_L // H \\ \sigma \in \mathrm{Stab}_G(i)}} (x - \overline{F}^{\sigma}(\alpha_{(1,1)}, \ldots, \alpha_{(m,i-1)}, 0, \ldots, 0, \alpha_{(1,i+1)}, \ldots, \alpha_{(m,l)})), \\
&\prod_{\substack{\sigma \in S_m \wr S_L // H \\ \sigma \in \mathrm{Stab}_G(i)}} (x - F_{t+1}^{\sigma}(0, \ldots, 0, \alpha_{(i,1)}, \ldots, \alpha_{(i,m)}, 0, \ldots, 0)\Bigg) \\
=&\prod_{i=1}^{l} \Bigg(\prod_{\substack{\sigma \in S_m \wr S_L // H \\ \sigma \in \mathrm{Stab}_G(i)}} (x - (\overline{F}^{\sigma}(\alpha_{(1,1)}, \ldots, \alpha_{(m,i-1)}, 0, \ldots, 0, \alpha_{(1,i+1)}, \ldots, \alpha_{(m,l)}) + \\
&F_{t+1}^{\sigma}(0, \ldots, 0, \alpha_{(i,1)}, \ldots, \alpha_{(i,m)}, 0, \ldots, 0))\Bigg) \\
=&\prod_{i=1}^{l} \Bigg(\prod_{\substack{\sigma \in S_m \wr S_L // H \\ \sigma \in \mathrm{Stab}_G(i)}} (x - F^{\sigma}(\alpha_{(1,1)}, \ldots, \alpha_{(m,l)})\Bigg) \\
=& \prod_{\sigma \in S_m \wr S_L // H} (x - F^{\sigma}(\alpha_{(1,1)}, \ldots, \alpha_{(m,l)})) \\
=&R(G, H, F, f)^c = R
\end{aligned}$$

where $c = \mathrm{mult}(M_k, \mathcal{M})$. So that $\mathrm{PolyRoot}(R, c)$ yields $R(G, H, F, f)$ as desired and Algorithm 7 (RelLinResolv) returns $R(G, H, F, f)$. □

4.5 Implementation and Extensions

In order to construct the resolvent efficiently, the multiset constructed in Algorithm 6 (RLRSetup) should be sorted in order of decreasing multiplicities to ensure that the degree is kept as small as possible for as long as possible in the resultants.

However, in practice, it is much more efficient to compute all resultants before any multiplication takes place instead of the recursive algorithm given here. It is also possible to remove the need for the PolyRoot algorithm in Algorithm 7 (RelLinResolv) if the loop is constructed appropriately.

It should be noted that given a carefully constructed loop, we can actually compute the resolvent with respect to $S_m \wr G$ for any transitive group G. Using Algorithm 5 (SumZerosExt) we can easily write the loop that computes the relative resolvent $R(S_m \wr G, H, F, f)$ where $H = \mathrm{Stab}_{S_m \wr G}(F)$.

Proposition 4.6. *Let $G \leq S_l$ be a transitive permutation group. Let C be the sequence of coefficients of a multivariate polynomial F, let Φ and f be as described in Algorithm 7* (RelLinResolv) *and let $\mathcal{M} = \{M_1^{\times 1}, \ldots, M_k^{\times 1}\}$ be the multiset constructed in Algorithm 6* (RLRSetup)*. Let* SumZerosExt *as in Algorithm 5. Then the relative linear resolvent with respect to $S_m \wr G$ is*

$$\prod_{\sigma \in G} \mathrm{SumZerosExt}([\mathrm{LinResolv}(M_1, f_{\sigma(1)}), \ldots, \mathrm{LinResolv}(M_k, f_{\sigma(k)})])$$

For multisets of arbitrary multiplicity, let $\mathcal{M} = \{M_1^{\times m_1}, \ldots, M_k^{\times m_k}\}$. Let the sequence $[N_1, N_2, \ldots, N_j]$ be a sequence representation of $\mathcal{M}$ with $j = \#\mathcal{M}$ and $N_1 = \ldots = N_{m_1} = M_1$, $N_{m_1+1} = \ldots = N_{m_1+m_2} = M_2$ and so on. For simplicity of numbering, let L_i be the index of the first N corresponding to M_i and let $J_i = L_i + m_i - 1$, the index of the last N corresponding to M_i for $1 \leq i \leq k$. Then the relative linear resolvent with respect to $S_m \wr G$ is

$$\prod_{\sigma \in G} \prod_{i=1}^{k} \mathrm{PolyRoot}(\mathrm{SumZerosExt}([\mathrm{LinResolv}(N_{L_i}, f_{\sigma(L_i)}), \ldots, \mathrm{LinResolv}(N_{J_i}, f_{\sigma(J_i)})]), m_i)$$

Proof. The proof of this is the same as the proof of Theorem 4.5. The use of Algorithm 5 (SumZerosExt) changes the recursion to a loop and we may easily substitute the action of G for the loop that goes through every permutation because the action of $S_m \wr G$ on each block of the block system $\{\{1, \ldots, m\} \times \{a\} \mid a \in \{1, \ldots, l\}\}$ is determined by the group G. $\square$

In a similar way, we can also compute the relative resolvent with respect to $S_m \wr H \wr J$. If we let G in Proposition 4.6 be $H \wr J$, then reindexing the set Φ as ordered pairs as determined by the block system of $H \wr J$ yields the desired result.

While the algorithm is relevant for any field, some loss of precision that occurs in the computation of the resultants and the greatest common divisor occasionally causes incorrect results in p-adic fields. Ways to compensate for these issues are currently being pursued, including the modified computation discussed in Section 3.6.

Algorithm 6. (RLRSetup)**.** Sets up the input for Algorithm 7 (RelLinResolv) and returns a multiset $\mathcal{M}$ to be sent to Algorithm 7 (RelLinResolv).

Input: A sequence $C = [c_1, \ldots, c_s]$ whose entries are the coefficients of F, a multivariate polynomial of the form $F(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_sx_s + 0x_{s+1} + \cdots + 0x_n$ with $s \leq n$ and $c_i \in K$ for $1 \leq i \leq s$ and a positive integer m representing the size of the partitions of C that will form the multisets.

Output: A multiset $\mathcal{M}$ whose elements are multisets each representing a partition of C.

```
(1) $P \leftarrow [c_1, \ldots, c_m \mid c_{m+1}, \ldots, c_{2m} \mid \ldots \mid c_{qm+1}, \ldots c_s] = [C_1 \mid C_2 \mid \ldots \mid C_p]$
    where $C_i$ is the $i$th part of the partition $P$.
    (Partition $C$ into $q$ parts of size $m$ and one part of size $r$ where $s = q \cdot m + r$.)
(2) $K \leftarrow [M_1, M_2, \ldots, M_p]$
    where $M_i$ is the multiset representation of $C_i$.
    (Construct a sequence of multisets from $P$.)
(3) $K \leftarrow [M_1 - \{0^{\times \operatorname{mult}(0,M_1)}\}, \ldots, M_p - \{0^{\times \operatorname{mult}(0,M_p)}\}]$
    (Remove the zeros from each $M_i$, $1 \leq i \leq p$, any empty multisets are removed from $K$ in this step.)
(4) $\mathcal{M} \leftarrow$ multiset representation of $K$.
    (Construct the multiset representation of $K$ so that $M$ is a multiset of multisets.)
(5) return $\mathcal{M}$.
```

Algorithm 7. (RelLinResolv). Returns the relative linear resolvent $R(G, H, F, f)$ where $G = S_m \wr S_l$ and $H = \mathrm{Stab}_G(F)$.

Input: A sequence of polynomials, $\Phi = [f_1, \ldots, f_l]$ where $\deg(f_i) = m$ for $1 \le i \le l$, such that $\{\alpha_i^{(j)} \mid 1 \le j \le m\}$ are the roots of f_i in $\overline{K}$ and $\{\{\alpha_i^{(1)}, \ldots, \alpha_i^{(m)}\} \mid 1 \le i \le l\}$ is a block system of $S_m \wr S_l$ and a multiset $\mathcal{M} = \{M_1^{\times m_1}, \ldots, M_k^{\times m_k}\}$ of multisets representing the partitions of the sequence $[c_1, \ldots, c_s]$ created by RLRSetup($[c_1, \ldots, c_s], m$). The entries of the sequence are the coefficients of F, a multivariate polynomial of the form

$$F(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_sx_s + 0x_{s+1} + \cdots + 0x_n$$

with $s \le n$ and $c_i \in K$ for $1 \le i \le s$.

Output: The resolvent $R(G, H, F, f)$ where $G = S_m \wr S_l$, $H = \mathrm{Stab}_G(F)$, F is defined as in the input, and $\prod_{i=1}^{l} f_i = f \in K[x]$ of degree $lm = n$

(1) If $\#\mathcal{M} = 1$, then return $\prod_{i=1}^{l} \mathrm{LinResolv}(M_1, f_i)$
(Stop condition for the recursion: there is only one partition of coefficients of F; i.e., F has fewer than m coefficients.)

(2) $\overline{\mathcal{M}} = \mathcal{M} - \{M_k^{\times 1}\}$
(Remove one partition of the coefficients of F.)

(3) $R \leftarrow \prod_{i=1}^{l} \mathrm{SumZeros}(\mathrm{RelLinResolv}(\overline{\mathcal{M}}, \Phi_i), \mathrm{LinResolv}(M_k, f_i))$
where $\Phi_i = [f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_l]$.
(Recursively call algorithm with one fewer partition of coefficients and compute the new polynomial by "summing" the roots of the new factor with the recursive result.)

(4) $c \leftarrow \mathrm{mult}(M_k, \mathcal{M}) = m_k$
(Compute the number of extra factors resulting from the multiplicity of the last partition of coefficients of F.)

(5) return PolyRoot(R, c)
(Remove the extra multiplicities of roots and return the result.)

CHAPTER V

USING RESOLVENTS TO COMPUTE GALOIS GROUPS

In this chapter we will discuss the theorems outlining how the resolvent is used in the computation of Galois groups and then show some examples with partial results. In particular, we will show the creditability of the requirements of Algorithm 7 (RelLinResolv) and the relevance of computing the relative resolvent with respect to the wreath product of groups. We then detail the properties of the resolvent that make it useful for differentiating candidate Galois groups of a polynomial.

5.1 Theorems

5.1.1 *Requirements of Algorithm* 7 (RelLinResolv)

The computation in Algorithm 7 (RelLinResolv) requires factors of the polynomial f in a normal extension of K that meet the criteria in Theorem 4.5. Proposition 5.1 guarantees that the conditions required by Theorem 4.5 can be attained. This proposition and proof appear in [Mil17].

Proposition 5.1. *Let $f \in K[x]$ be irreducible and separable of degree n. Let N denote the splitting field of f. If T/K is a normal subextension of N/K, then f factors over T as a product of distinct irreducible polynomials of the same degree.*

Proof. Let $\alpha^{(1)}, \ldots, \alpha^{(n)}$ denote the roots of f in N. As f is squarefree, all factors of f are distinct. For each root $\alpha^{(i)}$ of f we denote by $f_{j(i)}$ the irreducible factor of f, over T, for which $\alpha^{(i)}$ is a root.

Let $L = K[x]/(f)$, and let $L_1 = T \cap L$. For $1 \leq i \leq n$, the conjugates of L are $L^{(i)} = K(\alpha^{(i)}) = L_1(\alpha^{(i)})$. Similarly, the conjugates of L_1 are $L_1^{(i)} = L^{(i)} \cap T$. Since $K(\alpha^{(i)})$ is always the same up to isomorphism, we have the following diagram for $1 \leq i \leq n$:

$$\begin{tikzcd}
 & N \arrow[d, dash] \\
L^{(i)} \arrow[r, phantom, "\subseteq"] \arrow[d, dash] & TL^{(i)} \arrow[d, dash] \\
L^{(i)} \cap T = L_1^{(i)} \arrow[r, phantom, "\subseteq"] \arrow[d, dash] & T \\
K &
\end{tikzcd}$$

where both $L^{(i)}/L_1^{(i)}$ and $TL^{(i)}/T$ have degree $\deg(f_{j(i)})$. Thus, each $\alpha^{(i)}$ is a root of an irreducible factor of f over T of degree

$$\frac{[L^{(i)} : K]}{[L^{(i)} \cap T : K]}.$$

$\square$

5.1.2 Embedding in the Wreath Product

In this section we will show that once we have the conditions necessary for Algorithm 7 (RelLinResolv), that the Galois group embeds in the wreath product for which the algorithm is computing the resolvent. First, we will show that the Galois group has the necessary block system. This theorem appears in [GK00] with the proof from [Mil17].

Theorem 5.2. *Let $E_1 = K(\beta)$, $E_2 = K(\alpha)$ be algebraic extensions of K with $K \subseteq E_1 \subseteq E_2$ and g, $f \in K[x]$ be the minimal polynomials of β and α, respectively. Let $h \in K[x]$ be the embedding polynomial with $h(\alpha) = \beta$. Denote the conjugates of α and β in some algebraic closure with $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_m$, respectively. Defining $B_i = \{\alpha_j \mid h(\alpha_j) = \beta_i\}$ it follows that $B_1, \ldots, B_m$ form a block system of* $\mathrm{Gal}(f)$. *Furthermore, $n = |B_i|\, m$.*

Proof. Let $\sigma \in \mathrm{Gal}(f)$, and let i satisfy $1 \leq i \leq m$. Since $\beta \in K(\alpha)$ is algebraic over K, $\sigma(\beta_i)$ is a conjugate of β. We claim that $\sigma(\beta_i) = \beta_k$ if and only if $\sigma(B_i) = B_k$.

Suppose $\sigma(\beta_i) = \beta_k$ and let $\delta \in B_i$. Since σ is an automorphism and h is a polynomial, we have that $\sigma(h(a)) = h(\sigma(a))$ for all a in the domain of h. This directly leads to

$$\begin{aligned} h(\sigma(\delta)) &= \sigma(h(\delta)) \\ &= \sigma(\beta_i) \\ &= \beta_k \end{aligned}$$

which implies that $\sigma(\delta) \in B_k$. Because δ was selected arbitrarily, we conclude that $\sigma(B_i) \subseteq B_k$. Furthermore, by a similar argument, $\beta_i = \sigma^{-1}(\beta_k)$ leads us to $\sigma^{-1}(B_k) \subseteq B_i$. This is equivalent to $B_k \subseteq \sigma(B_i)$ since σ is bijective. Therefore, the forward direction has been proven.

Conversely, suppose that $\sigma(B_i) = B_k$. Let $\tau \in \sigma(B_i)$. Then there exists $\delta \in B_i$ such that $\tau = \sigma(\delta)$. Furthermore, we have that $\delta = \sigma^{-1}(\tau)$ and $h(\delta) = \beta_i$. Putting all of this together we can determine $\sigma(\beta_i)$:

$$\begin{aligned}\sigma(\beta_i) &= \sigma(h(\delta)) \\ &= h(\sigma(\delta)) \\ &= h(\tau) \\ &= \beta_k.\end{aligned}$$

Hence the assertion has been proven. This implies that $\sigma(B_i)$ is either B_i or another set B_j. Since the sets $B_1, \ldots, B_m$ must be disjoint we have $\sigma(B_i) \cap B_i = \{B_i, \emptyset\}$ for $\sigma \in \mathrm{Gal}(f)$. The cardinality condition on B_i follows from the fact that $\mathrm{Gal}(f)$ is transitive. □

The following theorem is a version of the Krasner-Kaloujnine Theorem and the proof and the proof of the corollary are based on work in [Gei97] and in part on [DS07].

Theorem 5.3. *Let (G, Z) be a transitive, imprimitive permutation group with block system $\mathfrak{B} = \{B_1, \ldots, B_m\}$ where each block is size l. Let X and Y be finite sets such that $|X| = l$ and $|Y| = m$. Then G acts transitively on Y and there is $H \leq G$ that acts on X such that (G, Z) can be embedded in $(H \wr (G, Y), X \times Y)$.*

Proof. Let $Y = \{y_1, \ldots, y_m\}$ and $X = \{x_1, \ldots, x_l\}$. Let $\theta : Z \to X \times Y$ be a bijection such that $\theta(z) = (x_i, y_j) \implies z \in B_j$ for all $z \in Z$.

Using θ, we view G as a transitive, imprimitive permutation group on $X \times Y$ with blocks $B_j = X \times \{y_j\}$ for $1 \leq j \leq m$. We write $(x, y)g$ instead of $\theta^{-1}((x, y))g$.

Let $\psi : G \to S_m$ be the permutation representation of G with respect to the action of G on $\mathfrak{B}$. Let $g \in G$ with $\psi(g) = \sigma \in S_m$. Then G acts on Y by

$$(y_i)g = (y_i)\psi(g) = (y_i)\sigma = y_{\sigma(i)}.$$

Since the action of G on Z is transitive, the action of G on Y is also transitive.

Fix $y_1 \in Y$ and let $H = \mathrm{Stab}_G(y_1)$. Since $B_1 = X \times \{y_1\}$. This implies that H permutes the elements of X. Since $|X| = l$, we have that $\phi : H \to S_l$ is the permutation representation of H. Let $h \in H$ with $\phi(h) = \tau \in S_l$. So again H acts transitively on X by

$$x_i h = x_i \phi(h) = (x_i)\tau = x_{\tau(i)}.$$

Fix $(x_1, y_1) \in X \times Y$ and let $g \in G$ such that $(x_1, y_1)g = (x_2, y_2)$ for some $(x_2, y_2) \in X \times Y$.

With respect to g as above, define $f \in H^Y$ and $h \in (G, Y)$ by $(y_1)h = y_2$ and $(x_1)f^h(y_1) = (x_1)f(y_1 h^{-1}) = x_2$. Since G acts transitively on $X \times Y$ and H acts transitively on X, it is clear that we can define such a pair, (f, h) for each $g \in G$ given by the action of g on (x_1, y_1).

Define the map $\chi : G \to H \wr (G, Y)$ by $g \mapsto (f, h)$ defined as above by the action of g on the fixed point (x_1, y_1). Let $g \in G$ with $(x_1, y_1)g = (x_2, y_2)$ for some

$(x_2, y_2) \in X \times Y$.

$$\begin{aligned}(x_1, y_1)\chi(g) &= (x_1, y_1)(f, h)\\ &= (x_1 f^h(y_1), (y_1)h)\\ &= (x_2, y_2)\\ &= (x_1, y_1)g\end{aligned}$$

This shows that $\chi(g)$ acts on $X \times Y$ as G does. We use this to show that χ is a homomorphism. Let $g_1, g_2 \in G$ be arbitrary.

$$\begin{aligned}(x, y)\chi(g_1 g_2) &= (x, y)g_1 g_2\\ &= ((x, y)g_1)\chi(g_2)\\ &= ((x, y)\chi(g_1))(\chi(g_2)\\ &= (x, y)(\chi(g_1)\chi(g_2))\end{aligned}$$

To prove that χ is injective, we show $\ker(\chi)$ is trivial. Let $g \in G$ with $g \in \ker(\chi)$. Then we have that

$$(x, y) = (x, y)\chi(g) = (x, y)g$$

for all $(x, y) \in X \times Y$. So we must have that $g = id_G$ and the kernel is trivial as desired. □

Corollary 5.4. *Let $K \subset L \subset M$ be finite separable field extensions. Then the Galois group* $\mathrm{Gal}(M/K)$ *of M over K can be embedded as a permutation group into the wreath product* $\mathrm{Gal}(M/L) \wr \mathrm{Gal}(L/K)$.

Proof. Let $L = K(\alpha)$ and $M = L(\beta)$ with $h(\beta) = \alpha$ for $h \in K[t]$. Fix a normal closure N of M over K that contains L. Let $G = \mathrm{Gal}(N/K)$. Define Z to be the K-embeddings of M into N, Y to be the K-embeddings of L into N, and X to be the L-embeddings of M into N. Then $\mathrm{Gal}(M/K) = (G, Z)$ is a transitive imprimitive permutation group with block system $\mathfrak{B} = \{B_y \mid y \in Y\}$ with each block $B_y = \{z \in Z \mid h(z) = y\}$ by Theorem 5.2. Fix $y \in Y$, and set $H = \mathrm{Stab}_G(y)$. The statement follows since $\mathrm{Gal}(M/L) \cong (H, X)$ and $\mathrm{Gal}(L/K) = (G, Y)$. $\square$

5.1.3 Properties of the Resolvent Polynomial

The following theorems and proofs, taken from [Gei03], demonstrate the properties of the resolvent polynomial that are useful for determining $\mathrm{Gal}(f)$. In all cases, it is the factorization of the resolvent polynomial (if it is squarefree) that provides the required information.

Theorem 5.5. *Let K be a field and $\overline{K}$ an algebraic closure of K. Let $f(x) \in K[x]$ be a monic, separable, irreducible polynomial of degree n and fix an ordering $\alpha_1, \ldots, \alpha_n \in \overline{K}$ of the roots of f. Let G be a transitive, subgroup of S_n such that* $\mathrm{Gal}(f) \leq G$. *Let H be a subgroup of G and $F \in K[x_1, \ldots, x_n]$ a G-relative H-invariant polynomial with $R(G, H, F, f)$ the corresponding resolvent polynomial. Then*

(1) $R(G, H, F, f) = \prod_{\sigma \in G//H} (x - F^\sigma(\alpha_1, \ldots, \alpha_n)) \in K[x]$.

(2) *Let* $Q(x) = \prod_{i=1}^{l} (x - F^{\sigma_i}(\alpha_1, \ldots, \alpha_n))$ *be a factor of* $R(G, H, F, f)$ *such that Q and $R(G, H, F, f)$ are relatively prime. Let* $S = \mathrm{Stab}_G(\{F^{\sigma_1}, \ldots, F^{\sigma_l}\})$. *Then* $\mathrm{Gal}(f) \leq S$ *if and only if* $Q(x) \in K[x]$. $\mathrm{Gal}(f) \leq S \implies Q \in K[x]$ *without the relatively prime condition.*

(3) *In particular, let $F^{\sigma}(\alpha_1, \ldots, \alpha_n)$ be a simple root of $R(G, H, F, f)$ then $\mathrm{Gal}(f) \leq \sigma H \sigma^{-1}$ if and only if $F^{\sigma}(\alpha_1, \ldots, \alpha_n) \in K$.*

Proof. (1) The coefficients of $R(G, H, F, f)$ are fixed by G and therefore also by $\mathrm{Gal}(f)$, so that $R(G, H, F, f) \in K[x]$.

(2) First assume that $\mathrm{Gal}(f) \leq S$, then it follows that $\{F^{\sigma_i}(\alpha_1, \ldots, \alpha_n) \mid 1 \leq i \leq l\}$ is also fixed by $\mathrm{Gal}(f)$, so that $Q \in K[x]$.

Now, let $Q \in K[x]$ so that Q and $R(G, H, F, f)/Q$ are relatively prime, and let $r = [G : H]$. Select a set $\{\sigma_1, \ldots, \sigma_l\}$ from a complete set of coset representatives, $\{\sigma_1, \ldots, \sigma_r\}$. Let $\tau \in \mathrm{Gal}(f)$, then for $1 \leq i \leq l$, $\tau(F^{\sigma_i}) = F^{\sigma_j}$ with $j \in \{1, \ldots, r\}$. Because $Q \in K[x]$, it follows that $\tau(F^{\sigma_i}(\alpha_1, \ldots, \alpha_n)) = F^{\sigma_k}(\alpha_1, \ldots, \alpha_n)$ with $k \in \{1, \ldots, l\}$. To see that $F^{\sigma_k}(\alpha_1, \ldots, \alpha_n) = F^{\sigma_j}(\alpha_1, \ldots, \alpha_n)$, recall that Q and $R(G, H, F, f)$ are relatively prime, so $j \in \{1, \ldots, l\}$ as desired.

(3) Since $\mathrm{Stab}_G(F^{\sigma}) = \sigma H \sigma^{-1}$, the result follows from part (2) for $l = 1$.

□

Theorem 5.6. *Given the conditions of Theorem 5.5, let $r = [G : H]$ and $\tau : \mathrm{Gal}(f) \to S_r$ the permutation representation of $\mathrm{Gal}(f)$ with respect to the action of $\mathrm{Gal}(f)$ on the set of right coset representatives $G//H$. If $R(G, H, F, f)$ is separable, then the Galois group of $R(G, H, F, f)$, as a subgroup of S_r, is isomorphic to the group $\tau(\mathrm{Gal}(f))$.*

Proof. Let $\Delta = \{H\sigma_1, \ldots, H\sigma_r\}$ be a set of right cosets of H in G with $\{\sigma_1, \ldots, \sigma_r\} = G//H$ and set $\Omega = \{F^{\sigma_1}(\alpha_1, \ldots, \alpha_n), \ldots, F^{\sigma_r}(\alpha_1, \ldots, \alpha_n)\}$. Define $\overline{\psi} : \Delta \to \Omega$ by $H\sigma_i \mapsto F^{\sigma_i}(\alpha_1, \ldots, \alpha_n)$. We want to show that $\overline{\psi}$ is a bijection of sets. To see that $\overline{\psi}$

is well-defined and injective, consider the following equivalences.

$$\begin{aligned} H\sigma_i = H\tilde{\sigma_i} &\iff \sigma_i\tilde{\sigma_i}^{-1} \in H \\ &\iff F^{\sigma_i\tilde{\sigma_i}^{-1}} = F \\ &\iff F^{\sigma}(\alpha_1, \ldots, \alpha_n) = F^{\tilde{\sigma_i}^{-1}}(\alpha_1, \ldots, \alpha_n) \end{aligned}$$

The last line follows from $R(G, H, F, f)$ being separable.

Since $|\Delta| = |\Omega|$, we have that $\overline{\psi}$ is also surjective. Under this bijection, we have an isomorphism of permutation groups S_Δ and S_Ω, $\psi : S_\Delta \to S_\Omega$ such that $\psi(\omega)((F^{\sigma_i}(\alpha_1, \ldots, \alpha_n)) = \overline{\psi}(\omega(H\sigma_i))$. Let $\sigma \in \mathrm{Gal}(f)$. Define the permutation representation τ' of $\mathrm{Gal}(f)$ to S_Δ defined by $\tau'(\sigma)(H\sigma_i)) = H\sigma_i\sigma$ and let the homomorphism φ be the restriction of σ to $\mathrm{Gal}(R(G, H, F, f))$. We want to show that the following diagram commutes:

$$\begin{tikzcd} & \mathrm{Gal}(f) \arrow[dl, "\tau'"'] \arrow[dr, "\varphi"] & \\ S_\Delta \geq \tau'(\mathrm{Gal}(f)) \arrow[rr, "\psi"] & & \mathrm{Gal}(R(G, H, F, f)) \leq S_\Omega \end{tikzcd}$$

So

$$\varphi(\sigma)(F^{\sigma_i}(\alpha_1, \ldots, \alpha_n)) = F^{\sigma_i\sigma}(\alpha_1, \ldots, \alpha_n) \text{ for } 1 \leq i \leq r$$

and we get

$$\begin{aligned} \varphi(\sigma)(F^{\sigma_i}(\alpha_1, \ldots, \alpha_n)) &= F^{\sigma_i\sigma}(\alpha_1, \ldots, \alpha_n) \\ &= \overline{\psi}(H\sigma_i\sigma) = \overline{\psi}(\tau'(\sigma)(H\sigma_i)) \\ &= \psi(\tau'(\sigma))(F^{\sigma_i}(\alpha_1, \ldots, \alpha_n)). \end{aligned}$$

Since φ is surjective, it follows that $\mathrm{Gal}(R(G,H,F,f)) = \varphi(\mathrm{Gal}(f)) = \psi(\tau'(\mathrm{Gal}(f))$. Identifying Δ and Ω with $\{1,\ldots,r\}$ proves the theorem. $\square$

Remark. It follows from Theorem 5.6 that the set of the degrees of the irreducible factors of $R(G,H,F,f) \in K[x]$ equals the set of orbit lengths induced by the action of $\tau(\mathrm{Gal}(f))$ on the set $\{1,\ldots,r\}$.

Theorem 5.6 gives us a method for using the factorization of the resolvent for differentiating among possible Galois groups of f. By computing and factoring the resolvent polynomial (over the base field) and comparing the degrees of the irreducible factors to the orbit lengths generated by the possible Galois groups, we can rule out the groups that generate orbit lengths that do not match the degrees of the factors of the resolvent.

5.2 Examples

All examples were computed using the Magma computer algebra system.

Example 5.7. Let $K = \mathbb{Q}$ and $f(x) = x^{16} - 2x^{15} + 3x^{14} + 8x^{13} + 15x^{12} - 31x^{11} + 92x^{10} + 166x^9 - 6x^8 - 83x^7 + 921x^6 + 1597x^5 + 431x^4 - 371x^3 + 2303x^2 + 4116x + 2401$. If we factor f over the normal extension $T = \mathbb{Q}(\zeta_5)$, we obtain four degree-four factors of f. Let $F(x_1,\ldots,x_{16}) = x_1 + x_2 + 2x_5 + 2x_6 + x_9 + x_{10}$. The factorization of $R(S_4 \wr S_4, \mathrm{Stab}_{S_4 \wr S_4}(F), F, f)$ yields three factors each of degree 864. Examining the orbit length partitions of the transitive subgroups of $S_4 \wr C_4$, yields 11 groups that could be isomorphic to $\mathrm{Gal}(f)$.

Example 5.8. Let $K = \mathbb{Q}$ and $f(x) = x^{30} - x^{28} - 3x^{27} + 49x^{26} + 5x^{25} - 32x^{24} - 127x^{23} + 972x^{22} + 134x^{21} - 340x^{20} - 2191x^{19} + 10335x^{18} + 1146x^{17} - 1364x^{16} - 18441x^{15} + 62708x^{14} + 2869x^{13} - 1777x^{12} - 72551x^{11} + 205320x^{10} - 5465x^9 - 6770x^8 - 102531x^7 +$

$288164x^6 - 36848x^5 + 4833x^4 - 626x^3 + 61x^2 - 8x + 1$. If we factor f over the normal extension $T = \mathbb{Q}(\zeta_7)$, we obtain six degree-five factors of f. Let $F(x_1, \ldots, X_{30}) = x_1 + x_2 + 2x_3 + 2x_4 + 2x_6$. The factorization of $R(S_5 \wr S_6, \mathrm{Stab}_{S_5 \wr S_6}(F), f)$ yields five factors each of degree 900. Examining the orbit length partitions of the transitive subgroups of $S_5 \wr S_6$ yields 12 groups that could be isomorphic to $\mathrm{Gal}(f)$.

Example 5.9. Let $\mathbb{Q}_3$ be the field of 3-adic numbers. Let $f(x) = x^9 + 6x^8 + 3x^3 + 18x + 6 \in \mathbb{Q}_3[x]$. Then $\tau(x) = x^2 + 3072$ generates the maximal tamely ramified subfield T of the splitting field of f [GP12], and $\psi(x) = x^3 + 36x^2 + 3x + 21$ generates a subfield L_1 of $\mathbb{Q}_3[x]/(f)$ such that, by the earlier proposition, f splits into three degree-3 polynomials over TL_1. Let $G = S_3 \wr S_3$. There are 24 transitive subgroups of G which could be isomorphic to $\mathrm{Gal}(f)$. We use a series of multivariate polynomials to reduce the possibilities.

Let $F(x_1, \ldots, x_9) = x_1 + x_4 + x_7$. The sequence $[1, 0, 0, 1, 0, 0, 1]$ represents F and $\mathrm{RLRSetup}([1, 0, 0, 1, 0, 0, 1], 3) = \{\{1^{\times 1}\}^{\times 3}\}$. If we factor the resolvent polynomial produced over $\mathbb{Q}_3$ then we obtain one degree-27 factor. Examining the orbit length partitions yields 12 possibilities for $\mathrm{Gal}(f)$,
{9T6, 9T10, 9T17, 9T20, 9T21, 9T22, 9T24, 9T25, 9T28, 9T29, 9T30}

Let $F(x_1, \ldots, x_9) = x_1 + x_4 + x_5$. The sequence $[1, 0, 0, 1, 1]$ represents F and $\mathrm{RLRSetup}([1, 0, 0, 1, 1], 3) = \{\{1^{\times 1}\}^{\times 1}, \{1^{\times 2}\}^{\times 1}\}$. The factorization of the resolvent yields one degree-54 factor, which has possible groups {9T10, 9T11, 9T12, 9T18, 9T20, 9T21, 9T24, 9T29, 9T30, 9T31}.

Intersecting this set with the previous possibilities reduces the possible groups to {9T10, 9T20, 9T21, 9T24, 9T29, 9T30, 9T31}. Let $F(x_1, \ldots, x_9) = x_1 + x_4 + x_5 + x_7$. The sequence $[1, 0, 0, 1, 1, 0, 1]$ represents F and $\mathrm{RLRSetup}([1, 0, 0, 1, 1, 0, 1], 3) =$

$\{\{1^{\times 1}\}^{\times 2}, \{1^{\times 2}\}^{\times 1}\}$. Factoring the resolvent yields one factor of degree 54 and one factor of degree 27.

The groups that have corresponding orbit lengths are {9T10, 9T11, 9T13, 9T18}. The intersection with the previous possibilities yields one group, 9T10. So $\mathrm{Gal}(f) \cong$ 9T10.

REFERENCES

[Coh93] Henri Cohen, *A course in computational algebraic number theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[DF04] David S. Dummit and Richard M. Foote, *Abstract algebra*, third edition ed., John Wiley and Sons, Inc., Hoboken, NJ, 2004.

[DS07] Bart De Smit, *Galois groups and wreath products*, `http://www.math.leidenuniv.nl/~desmit/notes/krans.pdf`, October 2007.

[Gei97] Katharina Geißler, *Zur Berechnung von Galoisgruppen*, Master's thesis, Technische Universität Berlin, Berlin, 1997.

[Gei03] ______, *Berechnung von Galoisgruppen über Zahl- und Funktionenkörpern*, Ph.D. thesis, Technische Universität Berlin, Berlin, 2003.

[GK00] Katharina Geißler and Jürgen Klüners, *Galois group computation for rational polynomials*, Journal of Symbolic Computation **30** (2000), no. 6, 653 – 674.

[GP12] Christian Greve and Sebastian Pauli, *Ramification polygons, splitting fields, and Galois groups of Eisenstein polynomials*, Int. J. Number Theory **8** (2012), no. 6, 1401–1424. MR 2965757

[Mel95] J.D.P. Meldrum, *Wreath products of groups and semigroups*, John Wiley and Sons, Inc., New York, NY, 1995.

[Mil17] Jonathan Milstead, *Computing Galois groups of Eisenstein polynomials over p-adic fields*, Ph.D. thesis, University of North Carolina at Greensboro, Greensboro, NC, 2017.

[Soi81] Leonard Soicher, *The computation of Galois groups*, Master's thesis, Concordia University, Montreal, 1981.