4-25-2022 1:15 PM

Towards a Generalization of Fulton's Intersection Multiplicity Algorithm

Ryan Sandford, *The University of Western Ontario*

Supervisor: Moreno Maza, Marc, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

In this manuscript we generalize Fulton's bivariate intersection multiplicity algorithm to a partial intersection multiplicity algorithm in the n-variate setting. We extend this generalization of Fulton's algorithm to work at any point, rational or not, using the theory of regular chains. We implement these algorithms in Maple and provide experimental testing. The results indicate the proposed algorithm often outperforms the existing standard basis-free intersection multiplicity algorithm in Maple, typically by one to two orders of magnitude. Moreover, we also provide some examples where the proposed algorithm outperforms intersection multiplicity algorithms which rely on standard bases, indicating the proposed algorithm is a viable alternative as a standard basis-free intersection multiplicity algorithm.

Keywords: Algebraic geometry, Computer algebra, Intersection multiplicity, Intersection number, Fulton's algorithm, Regular chains, Regular sequence.

Summary for Lay Audience

In this manuscript we describe a new, partial algorithm for computing intersection multiplicities. Given a system of polynomial equations, the intersection multiplicity of a solution point is a mathematically meaningful way of assigning a weight to that solution. To compute intersection multiplicities, complete algorithms exist which rely on a tool known as a standard basis. Standard bases are a powerful tool with many applications but suffer from some practical drawbacks. Although theoretically, it is always possible to compute a standard basis, practically this is not true; for some systems, standard bases can take hours, days, or weeks to compute. Moreover, intersection multiplicity algorithms which use standard bases assume the solution is the origin. If one wishes to compute the intersection multiplicity at a solution which is not the origin, they can in some cases apply mathematical techniques to shift their system in a way that makes the desired solution the origin, but practically this only works for a subset of "nice" solutions (points with rational coordinates). When the desired solution is not the origin and does not behave nicely (points with non-rational coordinates), intersection multiplicity algorithms which use standard bases cannot be used.

In this manuscript, we propose and implement an alternative, partial algorithm to address both of the above issues. By extending a solution to this problem which works for systems with two polynomials in two variables to a partial solution for systems with n polynomials in n variables, we obtain a powerful means of computing intersection multiplicities, which can work when techniques using standard bases may not. Moreover, we extend this partial algorithm to work at any point, rational or not, greatly extending the breadth of systems and solutions for which we can compute intersection multiplicities. Lastly, we implement these algorithmic improvements in MAPLE and show experiments where the proposed algorithm outperforms other intersection multiplicity algorithms.

Co-Authorship Statement

The main results of this manuscript are based on papers coauthored with Marc Moreno Maza and Jürgen Gerhard, see [23, 13]. In [23], Marc Moreno Maza proposed the problem of extending Fulton's algorithm and discovered an optimization in the special case of triangular regular sequences. In [13], Marc Moreno Maza played an integral role in extending the generalization of Fulton's algorithm to handle a regular chain as input, particularly in the design of the valuation algorithm (Algorithm 5) and in the description accompanying the generalization of Fulton's algorithm extended to regular chains. In [13], Jürgen Gerhard oversaw the implementation, acting as the primary consultant for efficient MAPLE programming practices and design decisions regarding the user interface. The author of this manuscript was responsible for the discovery and proofs pertaining to the core result of [23], the generalization of Fulton's algorithm. For [13], the author of this manuscript worked on the theory of extending the generalization of Fulton's algorithm to use regular chains, alongside Marc Moreno Maza, and implemented the resulting algorithm and its related commands. Moreover, the author of this manuscript proposed many of the implementation features, optional arguments, and performed all experiments.

Contents

List of Figures

List of Tables

List of Appendices

List of Algorithms

Chapter 1

Introduction

The study of singularities in algebraic sets is one of the driving application areas of computer algebra and has motivated the development of numerous algorithms and software, see the books [8, 16] for an overview. One important question in that area is the computation of intersection multiplicities. Given a point in the intersection of hypersurfaces, intersection multiplicity describes the behaviour of the hypersurfaces at the point of intersection. In a sense, intersection multiplicity is used to describe how hypersurfaces intersect rather than where they intersect. Support for the computation of intersection multiplicities has been integrated into the computer algebra systems Maple, Singular, and Magma and continues to be an active area of research.

Computing intersection multiplicities in the n-variate setting presents its own set of challenges. The first algorithmic solution to computing intersection multiplicities in the general setting was proposed by Mora, for which a modern presentation is given in [16]. Mora's approach relies on the computation of standard bases (Gröbner bases). Standard bases are a powerful tool in computational algebraic geometry but often run into practical issues. Namely, it is not always possible to compute a standard basis with the computing resources available.

In this manuscript we provide a viable alternative to algorithms which rely on standard bases to compute intersection multiplicities. To do so, we generalize Fulton's bivariate intersection multiplicity algorithm to a partial intersection multiplicity algorithm in the n-variate setting. We also extend this generalization of Fulton's algorithm to handle non-rational coordinates using the theory of regular chains. Both of these partial algorithms are implemented and benchmarked against other intersection multiplicity algorithms.

1.1 Examples

Intersection multiplicity generalizes the familiar notion of the multiplicity of a root of a univariate polynomial to the case of n polynomials in n variables. If p is a root of f a univariate polynomial in $\mathbb{K}[x]$, the multiplicity of f is the largest $m \in \mathbb{N}$ such that we can write $f = (x - p)^m g$ for some $g \in \mathbb{K}[x]$. When p is a solution to $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$, a similar notion can be defined in this more general setting. When studying systems of polynomial equations, it is intuitive to count the number of times they intersect at a given solution, rather than just studying the solutions themselves. Once tangency is accounted for, we then arrive at the notion of

intersection multiplicity.

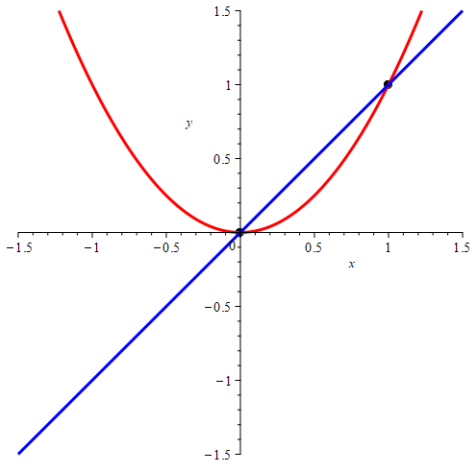
Figure 1.1 gives five systems of planar curves and their corresponding intersection multiplicity. Since we have yet to define intersection multiplicity precisely, we will describe each system in Figure 1.1 intuitively, to give the reader a feeling for how one would arrive at the correct intersection multiplicity. In Figure 1.1a, a line intersects a parabola transversally at two points. Since the line crosses the parabola only once at each point of intersection, and since the curves intersect transversally, the intersection multiplicity is 1 at both points. In Figure 1.1b, a line intersects a parabola tangentially at the origin. We can think of this system as analogous to the univariate notion of multiplicity. Here, the parabola would have a multiplicity of 2 at the origin. Since the line which intersects the parabola is the x-axis, it is natural that the intersection multiplicity of this system is also 2. In Figure 1.1c, there is no intersection at the origin, therefore, the intersection multiplicity at the origin is zero. The system in Figure 1.1d consists of a line and an elliptical curve, which intersect twice at the origin, once transversally and once tangentially. In this case, we can think of the intersection multiplicity as a weighted sum, with the transversal intersection contributing 1 to the sum, and the tangential intersection contributing 2, for a total intersection multiplicity of 3. Lastly, Figure 1.1e consists of fourteen branches transversally intersecting at the origin. Since the intersections are transversal, computing the intersection multiplicity amounts to counting the number of branches which pass through the origin. In this case, the intersection multiplicity is 14.

1.2 Literature Review

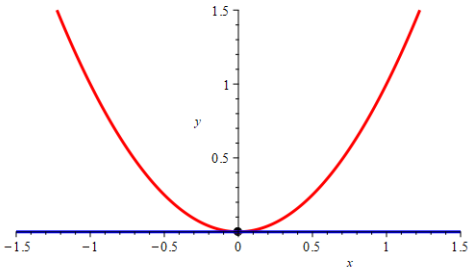
1.2.1 The Algorithm of Fulton

The case of two planar curves was solved by William Fulton in [12]. Fulton's algorithm is based on 7 properties (see Section 2.3 of the present manuscript) which uniquely define the intersection multiplicity of two plane curves at the origin, and yield a procedure for computing it, see Algorithm 1. If the input is a pair (f_0, g_0) of bivariate polynomials over some algebraically closed field $\mathbb{K}$, then Fulton's 7 properties act as a set of *rewrite rules* replacing (f_0, g_0), by a sequence of pairs $(f_1, g_1), (f_2, g_2), \ldots$ of bivariate polynomials over $\mathbb{K}$, which preserves the intersection multiplicity at the origin. This process may split the computation and terminates in each branch once reaching a pair for which the intersection multiplicity at the origin can be determined. This is an elegant process, using only standard operations on polynomials to rewrite the input system as a series of simpler input systems, until the intersection multiplicity can be determined.

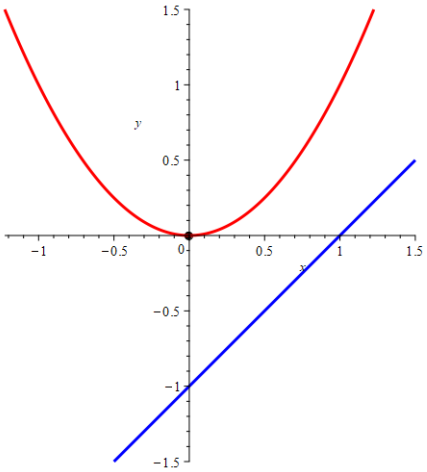
In theory, Fulton's algorithm could be applied to any point p rather than just the origin, by performing a change of coordinates, which preserves intersection multiplicity as discussed in 2.3. Points containing non-rational coordinates, and the explicit implementation of these non-rational numbers, present many challenges to computer algebra systems. This limits the applicability of changing coordinates to only points whose coordinates which are rational, and hence, Fulton's algorithm, as presented in [12], is essentially limited to points with rational coordinates.



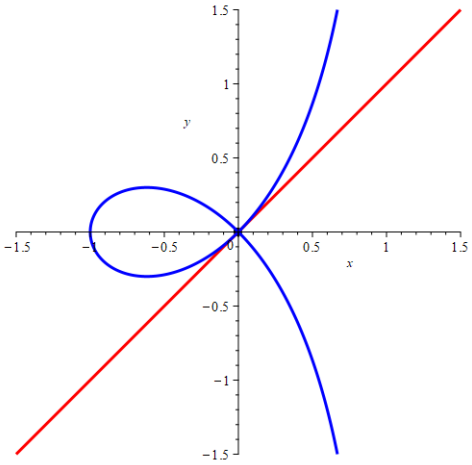
(a) $\mathrm{Im}\big((0,0);\ y-x, y-x^2\big) = 1$
$\mathrm{Im}\big((1,1);\ y-x, y-x^2\big) = 1$



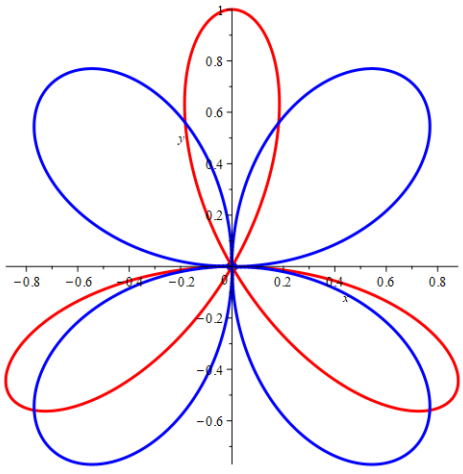
(b) $\mathrm{Im}\big((0,0);\ y, y-x^2\big) = 2$



(c) $\mathrm{Im}\big((0,0);\ y-x+1, y-x^2\big) = 0$



(d) $\mathrm{Im}\big((0,0);\ x^3+xy^2+x^2-y^2, y-x\big) = 3$



(e) $\mathrm{Im}\big((0,0);\ (x^2+y^2)^2+3x^2y-y^3, (x^2+y^2)^3-4x^2y^2\big) = 14$

Figure 1.1: Intersection Multiplicities of Planar Curves

1.2.2 Maple's `IntersectionMultiplicity` Algorithm

An alternative approach to that of Mora was investigated in [22, 1, 27], building off of the algorithm of Fulton. Given n polynomials $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ generating a zero-dimensional ideal, and a point $p \in \mathbf{V}(f_1, \ldots, f_n)$, the authors of [22, 1, 27] propose an algorithmic criterion for reducing the intersection multiplicity of p in $\mathbf{V}(f_1, \ldots, f_n)$, to computing another intersection multiplicity with $n-1$ polynomials in $n-1$ variables. For this criterion to be applicable, a transversality condition must hold. Unfortunately, this assumption does not generally hold.

In 2014, the `RegularChains` library introduced the `AlgebraicGeometryTools` subpackage, which contained commands for such computations. The main command, `IntersectionMultiplicity`, based on [22, 1, 27], appeared promising as a viable alternative to standard basis methods for computing intersection multiplicities but suffered from two main issues. The first issue was caused by a buggy implementation which resulted in many standard examples returning errors. This greatly hindered the practicality of this command as an alternative to other intersection multiplicity algorithms, see Chapter 7. The second issue observed was a slow performance on many standard examples, see Chapter 7. This is due to the underlying algorithm invoking the heavy duty `LimitPoints`[1] command several times. Although this approach avoids the use of standard bases, the several invocations to the `LimitPoints` command has severely hindered `IntersectionMultiplicity`'s performance in our experiments. These issues essentially limit MAPLE's `IntersectionMultiplicity` command to the case of two planar curves and to systems with intersection multiplicity one (for which the implementation is optimized).

Going forward, we will refer to this algorithm as the algorithm of Vrbik et al. As we will discuss in Chapter 6, we have reimplemented MAPLE's `IntersectionMultiplicity` command as a hybrid procedure which applies several partial algorithms for computing the intersection multiplicity, thus it is necessary to distinguish between the algorithm described above and other algorithms we will implement under the same command.

At the time of this manuscript, we are actively reimplementing the algorithm of Vrbik et al. to strengthen the overall hybrid algorithm, discussed in Chapter 6, and to provide a more reliable comparison between the algorithms developed in this manuscript and other standard basis-free intersection multiplicity algorithms.

1.2.3 Singular's `iMult` Algorithm

SINGULAR's `iMult` [10], found in the normal library (`normal.lib` [15]), is based on the techniques of Mora and is a complete algorithm for computing intersection multiplicities at the origin. Since `iMult` can only compute intersection multiplicities at the origin, exploring algorithms which can compute intersection multiplicities at any point, rational or not, is still an important question. Moreover, the `iMult` command uses standard bases to compute intersection multiplicities, and is therefore limited to only those examples for a which a standard basis

[1] Consider a constructible set C given as the set theoretic difference of a one-dimensional variety V (say, a space curve) and a hypersurface H. The command LimitPoints applied to V and H computes the limit points, for Zariski topology, of C that do not belong to C. In lose terms, those are the points obtained when moving on H towards V. In practice, computing the limit points of such constructible set C is achieved by factoring univariate polynomials over a field of univariate Puiseux series.

can be obtained.

The output specifications of `iMult` should also be noted. Rather than computing the intersection multiplicity of a list of polynomials at the origin `iMult` computes the intersection multiplicity of a list of ideals, which is defined in [14]. In order to recover our definition of intersection multiplicity from the one used in `iMult`, we will pass to `iMult` a list I, J where I and J are ideals defined by $f_1, \ldots, f_{n-1}$ and f_n respectively. If $f_1, \ldots, f_n$ is the system of polynomial equations and the ring given to SINGULAR has a local ordering, this calling sequence will compute the intersection multiplicity of $f_1, \ldots, f_n$ consistent with the definition given in this manuscript.

For the purpose of this manuscript, we will use `iMult` as a benchmark for our algorithm's performance against intersection multiplicity algorithms that utilize standard bases in their approach.

1.2.4 Magma's `IntersectionNumber` Algorithm

The computer algebra system MAGMA [4] also provides support for computing intersection multiplicities. MAGMA's `IntersectionNumber` command computes the intersection multiplicity of two projective curves at a given point, or in the case of the `IntersectionNumbers` command, computes the intersection multiplicity of two projective curves at all points in their intersection. Points can lie in a finite field, the rationals, or an algebraic field. The `IntersectionNumber` and `IntersectionNumbers` command are based on the algorithm of [17].

Since the `IntersectionNumber` command is limited to two projective curves, and since the MAGMA is not publicly available, we do not compare the algorithms developed in this manuscript to `IntersectionNumber` in any of our experiments.

1.2.5 Methods in Numerical Polynomial Algebra

Multiple Roots of polynomial systems are of great interest in the community of numerical polynomial algebra, see the landmark paper [7] by Corless, Gianni and Trager and the landmark textbook [25] by Stetter. A Numerical method for computing the multiplicity structure of an isolated root of a polynomial system is proposed by Dayton and Zeng in [9]. Their method can work with exact or approximate data; it relies on the notion of a *dual space* of a polynomial ideal at one of its isolated roots and the authors prove that their notion of multiplicity matches that of intersection multiplicity considered in this thesis. Another numerical method for computing the intersection multiplicity of a polynomial system at one of its isolated roots is presented in [19] by Kobayashi, Suzuki and Sakai. Their method can work with exact or approximate data: it relies on he notion of algebraic correspondence in algebraic geometry and an application of Zeuthen's rule.

1.3 Contributions

Our contributions are as follows:

We extend Fulton's bivariate intersection multiplicity algorithm to a partial intersection multiplicity algorithm in the n-variate setting. We believe this generalization of Fulton's bivariate intersection multiplicity algorithm to a partial, n-variate intersection multiplicity algorithm to be the best generalization possible since one of the steps in Fulton's algorithm does not apply generically. To our knowledge, this is the first practical alternative, in the general, n-variate setting, to intersection multiplicity algorithms which rely on standard bases.

We provide an optimization for the generalization of Fulton's algorithm for cases when the input system is both a triangular set and a regular sequence. Although the generalization of Fulton's algorithm can successfully compute the intersection multiplicity for any system which is a triangular set and a regular sequence, the intersection multiplicity can instead be computed immediately by means of polynomial evaluation. This observation has two important consequences. First, when the input is known to be a triangular regular sequence, this optimization allows us to avoid long stacks of recursive calls, providing much faster results. Second, it suggests a new way to compute intersection multiplicities without using standard bases, namely, by exploring intersection multiplicity preserving triangular decomposition techniques.

To extend our generalization of Fulton's algorithm to work beyond points with rational coordinates, we provide a modified algorithm which applies techniques from the theory of regular chains. This modification allows the generalization of Fulton's algorithm to run on a collection of points, encoded by a zero-dimensional regular chain. This modification is far from trivial and requires significant changes to the underlying algorithm and adjustments to the theory to make the notion of intersection multiplicity coherent for regular chains rather than a point. The resulting partial algorithm can compute intersection multiplicities in the n-variate setting at any point, rational or not. This feature is desirable as it greatly extends the breadth of systems one can compute intersection multiplicities at. Moreover, this feature is important as all complete, n-variate intersection multiplicity algorithms that we are currently aware of are limited to only rational points.

Lastly, we implement both of these versions of the generalization of Fulton's algorithm in MAPLE, as well as the optimization for triangular regular sequences. We combine the generalization of Fulton's algorithm with MAPLE's existing intersection multiplicity algorithm, to form a hybrid procedure. These changes are implemented under the `IntersectionMultiplicity` command. Commands related to the `IntersectionMultiplicity` command, such as `TriangularizeWithMultiplicity` were also modified to be compatible with the new implementation. Finally, we provide experimental testing for the generalization of Fulton's algorithm and compare the results to both the algorithm of Vrbik et al. and SINGULAR's `iMult` command.

1.4 Summary

The manuscript has been organized using the following structure:

- Chapter 2 provides an overview of the mathematical background needed to understand the generalization of Fulton's algorithm.
- Chapter 3 introduces more advanced concepts such as Cohen-Macaulay Rings, Krull dimension, and depth, and uses these concepts to prove a key result which justifies the

input constraints placed on our generalization of Fulton's algorithm.

- Chapter 4 contains the core results of this manuscript. It begins by extending Fulton's properties, and then generalizes Fulton's algorithm to a partial algorithm in the n-variate setting. The chapter concludes by discussing an important optimization for triangular regular sequences.

- Chapter 5 extends the generalization of Fulton's algorithm to handle points with non-rational coordinates encoded by a zero-dimensional regular chain. The chapter begins with an overview of the background and notations used in the theory of regular chains necessary to develop such an algorithm.

- Chapter 6 describes the details of the implementation of these algorithms in MAPLE.

- Chapter 7 reports the results of our experiments, contrasting the generalization of Fulton's algorithm to the algorithm of Vrbik et al. and `iMult`.

Chapter 2

Intersection Multiplicity Preliminaries

Let $\mathbb{K}$ denote a field. With the exception of Section 5.1, $\mathbb{K}$ will be used to denote an algebraically closed field. Let $\mathbb{A}^n$ denote $\mathbb{A}^n(\mathbb{K})$, the affine space of dimension n over $\mathbb{K}$. We define the degree of the zero polynomial to be $-\infty$ with respect to any variable. All rings are assumed to be commutative and $1 \neq 0$.

2.1 Preliminaries

2.1.1 Notations for Polynomials

Let f and g be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ and fix some i such that $1 \leq i \leq n$. We will use the notation $x_1, \ldots, \widehat{x_i}, \ldots, x_n$ to denote the omission of x_i from the sequence $x_1, \ldots, x_n$. We will use $\mathrm{quo}(f, g; x_i)$ to denote the quotient of f and g as univariate polynomials in x_i with coefficients in $\mathbb{K}[x_1, \ldots, \widehat{x_i}, \ldots, x_n]$. Similarly, we will use $\mathrm{lc}(f)x_i$ to denote the leading coefficient of f as a univariate polynomial in $\mathbb{K}[x_1, \ldots, \widehat{x_i}, \ldots, x_n][x_i]$. Lastly we will use $\deg_{x_i}(f)$ to denote the degree of f with respect to x_i expressed as a univariate polynomial in $\mathbb{K}[x_1, \ldots, \widehat{x_i}, \ldots, x_n][x_i]$.

2.1.2 Ideals and Varieties

If I is an ideal of $\mathbb{K}[x_1, \ldots, x_n]$, we denote by $\mathbf{V}(I)$ the algebraic set (aka variety) consisting of the common zeros to all polynomials in I. An algebraic set $\mathbf{V}$ is irreducible, whenever $\mathbf{V} = \mathbf{V}_1 \cup \mathbf{V}_2$ for some algebraic sets $\mathbf{V}_1, \mathbf{V}_2$, implies $\mathbf{V} = \mathbf{V}_1$ or $\mathbf{V} = \mathbf{V}_2$. The ideal of an algebraic set $\mathbf{V}$, denoted by $\mathbf{I}(\mathbf{V})$, is the set of all polynomials which vanish on all points in $\mathbf{V}$. For $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$, we say $\mathbf{V}(f_1), \ldots, \mathbf{V}(f_n)$ have a common component which passes through $p \in \mathbb{A}^n$ if when we write $\mathbf{V}(f_1, \ldots, f_n)$ as a union of its irreducible components, say $\mathbf{V}_1 \cup \ldots \cup \mathbf{V}_m$, there is a $\mathbf{V}_i$ which contains p. Similarly, we say $f_1, \ldots, f_n$ have a common component through p when $\mathbf{V}(f_1), \ldots, \mathbf{V}(f_n)$ have a common component which passes through p. As we will see, intersection multiplicity is a local notion and hence, throughout this manuscript, we may assume $\mathbf{V}(f_1, \ldots, f_n)$ is equal to an irreducible component containing p.

Definition 2.1.1 (Prime Ideals) *If P is a proper ideal of a ring R, then P is prime if for any $a, b \in R$, if $ab \in P$ then either $a \in P$ or $b \in P$.*

Definition 2.1.2 (Maximal Ideals) *If M is a proper ideal of a ring R, then M is maximal if the only ideal which contains M is R itself.*

It should be noted maximal ideals are prime. Moreover, the quotient ideal of a ring with one of its maximal ideals is a field.

2.1.3 Extension of an Ideal

Definition 2.1.3 *Let A, B be rings and I and ideal of A. Let $\varphi : A \to B$ be a ring homomorphism. Then the extension of I is the ideal in B generated by $\varphi(I)$. That is*

$$\langle \varphi(I) \rangle = \left\{ \sum y_i \varphi(x_i) \mid x_i \in A, y_i \in B \right\}.$$

When $A \subseteq B$, I an ideal of A, and φ is the inclusion map, we will write IB to denote the extension of I in B. This notation will be useful to distinguish between ideals in polynomial rings and ideals in local rings in Sections 3.2 and 4.1.

2.1.4 Radical of an Ideal

Definition 2.1.4 (Radical of an Ideal) *Let I be an ideal of a ring R. The radical of I, written $\sqrt{I}$ is defined as*

$$\sqrt{I} = \{ r \in R \mid r^n \in I \text{ for some } n \in \mathbb{Z}^+ \}.$$

The radical of an ideal I is also the intersection of all primes in R containing I. Thus, an equivalent definition we will use for the radical of an ideal is $\sqrt{I} = \bigcap_{R \supset P \supseteq I, P \text{ prime}}$. An ideal I is said to be radical if $I = \sqrt{I}$.

Theorem 2.1.5 (Hilbert's Nullstellensatz) *Let J be an ideal of $\mathbb{K}[x_1, \ldots, x_n]$ where $\mathbb{K}$ is an algebraically closed field. Then $\mathbf{I}(\mathbf{V}(J)) = \sqrt{J}$.*

2.1.5 Affine Changes of Coordinates

The following definitions are needed to state one of Fulton's properties in Section 2.3 and to generalize this property in Section 4.1.

Definition 2.1.6 (Polynomial Map) *Let $\mathbf{V}$ and $\mathbf{W}$ be irreducible algebraic sets in $\mathbb{A}^n$ and $\mathbb{A}^m$ respectively. A mapping $\phi : \mathbf{V} \to \mathbf{W}$ is called a polynomial map if there are polynomials $T_1, \ldots, T_m \in \mathbb{K}[x_1, \ldots, x_n]$ such that $\phi(a_1, \ldots, a_n) = (T_1(a_1, \ldots, a_n), \ldots, T_m(a_1, \ldots, a_m))$.*

Definition 2.1.7 (Affine Change of Coordinates) *An affine change of coordinates on $\mathbb{A}^n$ is a polynomial map $T = (T_1, \ldots, T_n) : \mathbb{A}^n \to \mathbb{A}^n$ such that each T_i is a polynomial map of degree 1 and T is bijective.*

2.1.6 Modules

The definition of a module is needed to state the definitions and theorems of Section 3.2 in full generality.

Definition 2.1.8 (Module) *Let R be a ring, and 1 is its multiplicative identity. A left R-module M consists of an abelian group $(M, +)$ and an operation $\cdot : R \times M \to M$ such that for all $r, s \in R$ and $x, y \in M$, we have*

- $r \cdot (x + y) = r \cdot x + r \cdot y$
- $(r + s) \cdot x = r \cdot x + s \cdot x$
- $(rs) \cdot x = r \cdot (s \cdot x)$
- $1 \cdot x = x$.

For the modules we will encounter, the ring R will take the role of both the ring and the module. It is easy to check that R is indeed an R-module.

Definition 2.1.9 (Finitely Generated Module) *An R-module M is finitely generated if there exist finitely many elements $x_1, \ldots, x_n \in M$ such that every $m \in M$ can be written as $m = r_1 m_1 + \ldots + r_n m_n$ for some $r_1, \ldots, r_n \in R$.*

Let R be a ring. Consider the R as an R-module. Clearly R is finitely generated as an R-module since the multiplicative identity 1 is in R.

Definition 2.1.10 *Let I be an ideal of R, and let M be an R-module. Then, IM is the set of all finite sums $\sum r_i m_i$ where $r_i \in I$ and $m_i \in M$.*

Definition 2.1.11 (Annihilator) *The Annihilator of an R-module M, is the set*

$$\mathrm{Ann}_R(M) = \{r \in R \mid rm = 0, \text{ for all } m \in M\}.$$

2.1.7 Gröbner and Standard Bases

Following [16, Section 1.6] we will use the term standard basis to denote a Gröbner basis with a local variable ordering. It was shown by Daniel Lazard in [20] that through homogenization, any algorithm which computes a Gröbner basis can be used to compute a standard basis, and similarly any algorithm for computing a standard basis can be used to compute a Gröbner basis. Moreover, an algorithm for computing a Gröbner/Standard Basis, for an arbitrary variable ordering is presented in [16, Section 1.6]. Hence, when we refer to a standard basis-free algorithm we are referring to an algorithm that does not compute a Gröbner or a standard basis for the ideal generated by the input polynomials.

2.2 Local Rings and Intersection Multiplicity

Definition 2.2.1 (Dimension of a Vector Space) *Let V be a vector space over base field F, then the dimension of F is the cardinality of a basis for V over F. We write* $\dim_F(V)$ *to denote the dimension of V.*

Definition 2.2.2 (Local Ring) *Let* $\mathbf{V}$ *be an irreducible algebraic set with $p \in \mathbf{V}$. We define the local ring of* $\mathbf{V}$ *at p as*

$$\mathcal{O}_{\mathbf{V},p} = \left\{ \frac{f}{g} \mid f, g \in \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}(\mathbf{V}) \text{ where } g(p) \neq 0 \right\}.$$

We will often refer to the local ring of $\mathbb{A}^n$ at p in this manuscript, in which case we will simply say the local ring at p and write

$$\mathcal{O}_{\mathbb{A}^n,p} = \left\{ \frac{f}{g} \mid f, g \in \mathbb{K}[x_1, \ldots, x_n] \text{ where } g(p) \neq 0 \right\}.$$

Local rings have a unique maximal ideal. In the case of $\mathcal{O}_{\mathbb{A}^n,p}$ all elements which vanish on p are in the maximal ideal and all of those that do not are units. Hence, given an element $f \in \mathbb{K}[x_1, \ldots, x_n]$ we can test whether f is invertible in $\mathcal{O}_{\mathbb{A}^n,p}$ by testing $f(p) \neq 0$.

Throughout this manuscript we will use the following conventions for ideals. With the exception of Sections 3.2 and 4.1, if $f_1, \ldots, f_n$ are polynomials in $\mathbb{K}[x_1, \ldots, x_n]$, we will write $\langle f_1, \ldots, f_n \rangle$ to denote the ideal generated by $f_1, \ldots, f_n$ in the local ring. In Sections 3.2 and 4.1 we need to distinguish between the ideals generated by $f_1, \ldots, f_n$ in the polynomial ring and in the local ring at a given point. To do this we will use $\langle f_1, \ldots, f_n \rangle$ to denote the ideal generated by $f_1, \ldots, f_n$ in the polynomial ring and $\langle f_1, \ldots, f_n \rangle \mathcal{O}_{\mathbb{A}^n,p}$ to denote the extension of this ideal in the local ring at some point $p \in \mathbb{A}^n$.

Definition 2.2.3 (Intersection Multiplicity) *Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$. We define the intersection multiplicity of $f_1, \ldots, f_n$ at $p \in \mathbb{A}^n$ as the dimension of the local ring at p modulo the ideal generated by $f_1, \ldots, f_n$ in the local ring at p, as a vector space over $\mathbb{K}$. That is,*

$$\mathrm{Im}(p;\, f_1, \ldots, f_n) = \dim_{\mathbb{K}}\Big(\mathcal{O}_{\mathbb{A}^n,p} / \langle f_1, \ldots, f_n \rangle\Big).$$

The following observation allows us to write the intersection multiplicity of a system of polynomials as the intersection multiplicity of a smaller system of polynomials, in fewer variables, when applicable. It follows from an isomorphism between the respective residues of local rings in the definition of intersection multiplicity.

Remark Let $f_1, \ldots f_n \in \mathbb{K}[x_1, \ldots, x_n]$ and $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$. If there are some f_i such that $f_i = x_i - p_i$, say $f_m, \ldots, f_n$ where $1 < m \leq n$, then

$$\mathrm{Im}(p;\, f_1, \ldots, f_n) = \mathrm{Im}((p_1, \ldots, p_{m-1});\, F_1, \ldots, F_{m-1}),$$

where F_j is the image of f_j modulo $\langle x_m - p_m, \ldots, x_n - p_n \rangle$.When p is the origin in $\mathbb{A}^n$ and p' is the origin in $\mathbb{A}^{m-1}$ this result becomes: $\mathrm{Im}(p;\, f_1, \ldots, f_{m-1}, x_m, \ldots, x_n) = \mathrm{Im}(p';\, f_1(x_1, \ldots, x_{m-1}, 0, \ldots, 0), \ldots, f_{m-1}(x_1, \ldots, x_{m-1}, 0, \ldots, 0))$.

In cases where we consider the intersection multiplicity at the origin, we will not distinguish between $p \in \mathbb{A}^n$ and $p' \in \mathbb{A}^{m-1}$. Instead, we will simply write p to represent the origin in both $\mathbb{A}^n$ and $\mathbb{A}^{m-1}$. We will refer to the result given in the above remark frequently in Chapter 4.

2.3 Bivariate Intersection Multiplicity

It is shown in [12, Section 3-3] that the following seven properties characterize intersection multiplicity of bivariate curves. Moreover, these seven properties lead to a constructive procedure which computes the intersection multiplicity of bivariate curves, which is given in Algorithm 1.

Proposition 2.3.1 (Fulton's Properties) *Let $p = (p_1, p_2) \in \mathbb{A}^2$ and $f, g \in \mathbb{K}[x, y]$.*

(2-1) $\mathrm{Im}(p;\ f, g)$ *is a non-negative integer when* $\mathbf{V}(f)$ *and* $\mathbf{V}(g)$ *have no common component at p, otherwise* $\mathrm{Im}(p;\ f, g) = \infty$.

(2-2) $\mathrm{Im}(p;\ f, g) = 0$ *if and only if* $p \notin \mathbf{V}(f) \cap \mathbf{V}(g)$.

(2-3) $\mathrm{Im}(p;\ f, g)$ *is invariant under affine changes of coordinates on* $\mathbb{A}^2$.

(2-4) $\mathrm{Im}(p;\ f, g) = \mathrm{Im}(p;\ g, f)$.

(2-5) $\mathrm{Im}(p;\ f, g) \geq m_f m_g$ *where* m_f *and* m_g *are the respective tailing degrees of f and g expressed in* $\mathbb{K}[x - p_1, y - p_2]$. *Moreover,* $\mathrm{Im}(p;\ f, g) = m_f m_g$ *when* $\mathbf{V}(f)$ *and* $\mathbf{V}(g)$ *intersect transversally, i.e. have no tangent lines in common.*

(2-6) $\mathrm{Im}(p;\ f, gh) = \mathrm{Im}(p;\ f, g) + \mathrm{Im}(p;\ f, h)$ *for any* $h \in \mathbb{K}[x, y]$.

(2-7) $\mathrm{Im}(p;\ f, g) = \mathrm{Im}(p;\ f, g + hf)$ *for any* $h \in \mathbb{K}[x, y]$.

The following proposition was proved by Fulton in [12, Section 3-3]. It is included here for the readers convenience, as we will use similar arguments in later sections.

Proposition 2.3.2 *Algorithm 1 is correct and terminates.*

Proof By (2-3) we may assume p is the origin. Let f, g be polynomials in $\mathbb{K}[x, y]$ with no common component through the origin. By (2-1), $\mathrm{Im}(p;\ f, g)$ is finite. We induct on $\mathrm{Im}(p;\ f, g)$ to prove termination. Suppose $\mathrm{Im}(p;\ f, g) = 0$, then by (2-2), at least one of f or g does not vanish at the origin and Algorithm 1 correctly returns zero.

Now suppose $\mathrm{Im}(p;\ f, g) = n > 0$ for some $n \in \mathbb{N}$. Let r, s be the respective degrees of f, g evaluated at $(x, 0)$. By (2-4) we may reorder f, g to ensure $r \leq s$. Notice $r, s \neq 0$ since f, g vanish at the origin.

If $r < 0$, then f is a univariate polynomial in y which vanishes at the origin, hence f is divisible by y. By (2-6) we have,

$$\mathrm{Im}(p;\ f, g) = \mathrm{Im}(p;\ y, g) + \mathrm{Im}(p;\ \mathrm{quo}(f, y; y), g)\,.$$

Algorithm 1: Fulton's Algorithm

Function $\mathrm{im}(p; f, g)$

Input: Let: $x \succ y$

1. $p \in \mathbb{A}^2$ the origin.
2. $f, g \in \mathbb{K}[x, y]$ such that $\gcd(f, g)(p) \neq 0$.

Output: $\mathrm{Im}(p;\ f, g)$

if $f(p) \neq 0$ **or** $g(p) \neq 0$ **then** /* Red */
  return 0
$r \leftarrow \deg_x(f(x, 0))$
$s \leftarrow \deg_x(g(x, 0))$
if $r > s$ **then** /* Green */
  return $\mathrm{im}(p; g, f)$
if $r < 0$ **then** /* $y \mid f$, Yellow */
  write $g(x, 0) = x^m(a_m + a_{m+1}x + \ldots)$
  return $m + \mathrm{im}(p; \mathrm{quo}(f, y; y), g)$
else /* Blue */
  $g' \leftarrow \mathrm{lc}(f(x, 0)) \cdot g - (x)^{s-r}\mathrm{lc}(g(x, 0)) \cdot f$
  return $\mathrm{im}(p; f, g')$

By definition of intersection multiplicity $\mathrm{Im}(p;\ y, g) = \mathrm{Im}(p;\ y, g(x, 0))$. Since $g(x, 0)$ vanishes at the origin and since g has no common component with f at the origin, $g(x, 0)$ is a non-zero univariate polynomial divisible by x. Write $g(x, 0) = x^m(a_m + a_{m+1}x + \ldots)$ for some $a_m, a_{m+1}, \ldots \in \mathbb{K}$ where m is the largest positive integer such that $a_m \neq 0$. Applying (2-6), (2-5), and (2-2) yields

$$\mathrm{Im}(p;\ f, g) = m + \mathrm{Im}(p;\ \mathrm{quo}(f, y; y), g)\,.$$

Thus, Algorithm 1 returns correctly when $r < 0$. Moreover, we can compute $\mathrm{Im}(p;\ \mathrm{quo}(f, y; y), g) < n$ by induction.

Now suppose $0 < r < s$. By (2-7), replacing g with g' preserves the intersection multiplicity. Notice such a substitution strictly decreases the degree in x of $g(x, 0)$. After finitely many iterations, we will obtain curves F, G such that $\mathrm{Im}(p;\ f, g) = \mathrm{Im}(p;\ F, G)$ and the degree in x of $F(x, 0) < 0$.

Below we work through several examples to illustrate Fulton's algorithm. The first example is rather short but illustrates both key parts of the algorithm, namely rewriting the system and splitting computations. The second example explores the intricacies of the rewriting process. To be concise, both examples were chosen as to terminate after one split, although this is not a behaviour that occurs generically.

Example We will compute the intersection multiplicity of $f = x^2y + x$ and $g = x^2 + 2xy + y$ at p, the origin, using Algorithm 1. First notice $f(x, 0) = x$ and $g(x, 0) = x^2$, so we have $r = 1$

and $s = 2$, thus we must compute g' as described in line 12,

$$g' = g - xf = x^2 + 2xy + y - x(x^2y + x) = 2xy + y - x^3y.$$

Redefining g as g' and reordering f and g gives $f = 2xy + y - x^3y$ and $g = x^2y + x$. Since $f(x, 0) = 0$ and $g(x, 0) = x$, we have $r = -\infty$ and $s = 1$. We set $m = 1$ and compute $\text{quo}(f, y; y) = 2x + 1 - x^3$ and conclude,

$$\text{im}(p; f, g) = \text{im}(p; \text{quo}(f, y; y), g) + m = \text{im}(p; 2x + 1 - x^3, g) + 1 = 1,$$

since $\text{im}(p; 2x + 1 - x^3, g) = 0$ as $2x + 1 - x^3$ does not vanish at the origin.

In the below example, we must perform the computation of line 12 several times before we can compute the intersection multiplicity. Moreover, it illustrates how the computation of line 12 combines with the reordering described in line 7 to reduce the values of r and s over the course of several iterations.

Example We will compute the intersection multiplicity of $f = x^3 + x^2 + y$ and $g = x^4 + y$ at p, the origin, using Algorithm 1. First notice $r = 3$ and $s = 4$, so we will replace g with g' as in line 12,

$$g' = g - xf = x^4 + y - x(x^3 + x^2 + y) = x^4 + y - x^4 - x^3 - xy = y - x^3 - xy.$$

We now have $f = x^3 + x^2 + y$ and $g = y - x^3 - xy$ with $r, s = 3$. We compute g' again, which gives

$$g' = g - (-f) = y - x^3 - xy + x^3 + x^2 + y = x^2 - xy + 2y.$$

Redefining g as g' and reordering f and g we get $f = x^2 - xy + 2y$ and $g = x^3 + x^2 + y$ with $r = 2$ and $s = 3$. Computing g' we get,

$$g' = g - xf = x^3 + x^2 + y - x(x^2 - xy + 2y) = x^3 + x^2 + y - x^3 + x^2y - 2xy = x^2 + y + x^2y - 2xy.$$

Again we define g as g', thus $f = x^2 - xy + 2y$ and $g = x^2 + y + x^2y - 2xy$ and $r, s = 2$. Once more, we compute g' to get,

$$g' = g - f = x^2 + y + x^2y - 2xy - x^2 + xy - 2y = x^2y - xy - y.$$

Redefining g as g' and reordering f and g we get $f = x^2y - xy - y$ and $g = x^2 - xy + 2y$ with $r = -\infty$ and $s = 2$. Notice $g(x, 0) = x^2$, so $m = 2$. Moreover, $\text{quo}(f, y; y) = x^2 - x - 1$. We conclude

$$\text{im}(p; f, g) = \text{im}(p; \text{quo}(f, y; y), g) + m = \text{im}(p; x^2 - x - 1, g) + 2 = 2,$$

since $\text{im}(p; x^2 - x - 1, g) = 0$ as $x^2 - x - 1$ does not vanish at the origin.

Chapter 3

Regular Sequences

This chapter introduces the notion of a regular sequence, an important tool used in our generalization of Fulton's algorithm. We focus first on the dimension of a regular sequence and prove a key result which we will later use to characterize systems which have finite intersection multiplicity at a point. Since the proof of termination of Fulton's algorithm inducts on intersection multiplicity itself, it is crucial we ensure the intersection multiplicity of our input is finite if we wish to generalize Fulton's algorithm. Hence, the result proved in Theorem 3.2.8 lays the foundation for generalizing Fulton's algorithm.

3.1 Krull Dimension

We have already introduced the the notion of dimension for a vector space. In this section we introduce the notion of Krull dimension as well as some related terminology. Throughout this section let R be a ring.

Definition 3.1.1 (Length of a Chain) *Let $P_0, \ldots, P_n \in R$ be prime ideals in R, then we say the chain*
$P_0 \subset P_1 \subset \ldots \subset P_n \neq R$ has length n.

Definition 3.1.2 (Krull Dimension of a Ring) *The Krull dimension of a ring R, written* $\dim(R)$*, is the supremum of the lengths of all chains of prime ideals.*

Definition 3.1.3 (Height of a Prime Ideal) *The height of a prime ideal $P \subset R$, written* $\mathrm{ht}(P)$*, is the supremum of the lengths of all chains of prime ideals such that $P_0 \subset P_1 \subset \ldots \subset P_n = P \neq R$.*

Definition 3.1.4 (Height of an Ideal) *The height of an ideal $I \subset R$, written* $\mathrm{ht}(I)$*, is the smallest height of a prime ideal containing I.*

Definition 3.1.5 (Krull Dimension of an Ideal) *The Krull dimension of an ideal $I \subset R$, written* $\dim(I)$*, is the Krull dimension of the ring R/I.*

Definition 3.1.6 (Krull Dimension of an Irreducible Algebraic Set) *Let I be an ideal in the polynomial ring R. The Krull dimension of an irreducible algebraic set $\mathbf{V} = \mathbf{V}(I)$, written* $\dim(\mathbf{V})$*, is the Krull dimension of $R/\mathbf{I}(\mathbf{V})$.*

Definition 3.1.7 (Krull Dimension of an Algebraic Set) *The Krull dimension of an algebraic set is the maximal Krull dimension of its irreducible components.*

Definition 3.1.8 (Krull Dimension of Module) *The Krull dimension of an R-module M, written* $\dim(M)$*, is the Krull dimension of the ring* $R/\mathrm{Ann}_R(M)$.

3.2 Regular Sequences in a Local Ring

Definition 3.2.1 (Regular Sequence) *Let R be a commutative ring and M an R module. Let* $r_1, \ldots, r_d$ *be a sequence of elements in R. Then* $r_1, \ldots, r_d$ *is an M-regular sequence if* r_i *is not a zero-divisor on* $M/\langle r_1, \ldots, r_{i-1}\rangle M$ *for all* $i = 1, \ldots, d$ *and* $M \neq \langle r_1, \ldots, r_d\rangle M$.

For this manuscript, we are only concerned with regular sequences such that $R, M = \mathcal{O}_{\mathbb{A}^n,p}$. Thus, when stating propositions in full generality, we will use the term M-sequence and when referring to the special case of concern for this manuscript, where $R, M = \mathcal{O}_{\mathbb{A}^n,p}$, we will simply say $f_1, \ldots, f_n$ is a regular sequence, $f_1, \ldots, f_n$ is a regular sequence in the local ring, or $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.

Definition 3.2.2 (Depth) *Let R be a ring,* $I \subset R$ *an ideal and M an R-module. If* $M \neq IM$*, then the maximal length n of an M-sequence* $a_1, \ldots, a_n \in I$ *is called the I-depth of M and denoted by* $\mathrm{depth}(I, M)$*. If R is a local ring with maximal ideal m, then the m–depth of M is simply called the depth of M, that is,* $\mathrm{depth}(M) = \mathrm{depth}(m, M)$

From this definition we see the I–depth of M is 0 if and only if every element of I is a zero-divisor for M. In particular, for a local ring R with maximal ideal m, we have $\mathrm{depth}(m, R/m) = 0$.

Below is Corollary 7.6.12 of [16]

Proposition 3.2.3 *Let R be a Noetherian ring,* $I = \langle f_1, \ldots, f_n\rangle$ *an ideal of R, and M a finitely generated R–module. Assume that* $M \neq IM$*. Then*

1. $f_1, \ldots, f_n$ *is an M–sequence if and only if* $\mathrm{depth}(I, M) = n$*;*
2. *let* $J \subseteq R$ *be an ideal and* $f_1, \ldots, f_n$ *an M–sequence in J, then* $\mathrm{depth}(J, M/\langle f_1, \ldots, f_n\rangle M) = \mathrm{depth}(J, M) - n$.

Definition 3.2.4 (Cohen-Macaulay) *Let R be a local ring with maximal ideal m, M a finitely generated R- module. M is called a Cohen-Macaulay module if* $M = 0$ *or* $M \neq 0$ *and* $\mathrm{depth}(M) = \dim(M)$*. R is called a Cohen-Macaulay ring if it is a Cohen-Macaulay R-module.*

Below is Corollary 7.7.10 of [16]

Proposition 3.2.5 *Let R be a Noetherian local Cohen–Macaulay ring with maximal ideal m, and let* $I \subseteq R$ *be an ideal. Then*

1. $\mathrm{ht}(I) = \mathrm{depth}(I, R)$*;*

2. $\text{ht}(I) + \dim(R/I) = \dim(R)$;

Lemma 3.2.6 *$\mathcal{O}_{\mathbb{A}^n,p}$ is a Cohen-Macaulay ring for $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$.*

Proof Let m be the maximal ideal of $\mathcal{O}_{\mathbb{A}^n,p}$ given by $m = \langle x_1 - p_1, \ldots, x_n - p_n \rangle$. By Proposition 3.2.3 we have

$$\text{depth}\Big(\mathcal{O}_{\mathbb{A}^n,p}\Big) = n + \text{depth}\Big(m, \mathcal{O}_{\mathbb{A}^n,p}/m\Big) = n$$

since all elements in m are zero or zero-divisors in $\mathcal{O}_{\mathbb{A}^n,p}/m$. The Krull dimension of $\mathcal{O}_{\mathbb{A}^n,p}$ is also n and thus, $\mathcal{O}_{\mathbb{A}^n,p}$ is a Cohen-Macaulay ring.

Lemma 3.2.7 *Let I be an ideal in $\mathbb{K}[x_1, \ldots, x_n]$. Let $\mathbf{V} = \mathbf{V}(I)$ and take $p \in \mathbf{V}$. Then,*

$$\dim(\mathcal{O}_{\mathbb{A}^n,p}/\mathbf{I}(\mathbf{V})\,\mathcal{O}_{\mathbb{A}^n,p}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}).$$

Proof Recall $\mathbf{I}(\mathbf{V}) = \sqrt{I}$ by the Nullstellensatz and since $\mathcal{O}_{\mathbb{A}^n,p}$ is a localization of $\mathbb{K}[x_1, \ldots, x_n]$, we have $\sqrt{I}\mathcal{O}_{\mathbb{A}^n,p} = \sqrt{I\mathcal{O}_{\mathbb{A}^n,p}}$ by Proposition 3.11 in [2]. Recall the radical of an ideal is the intersection of all primes containing that ideal. Combining these observations we get $\sqrt{I}\mathcal{O}_{\mathbb{A}^n,p} = \sqrt{I\mathcal{O}_{\mathbb{A}^n,p}} = \bigcap_{\mathcal{O}_{\mathbb{A}^n,p} \supset Q \supseteq I\mathcal{O}_{\mathbb{A}^n,p}, Q \text{ prime}} Q$. Let Q' be the smallest prime in $\mathcal{O}_{\mathbb{A}^n,p}$ which contains $I\mathcal{O}_{\mathbb{A}^n,p}$. Since the height of an ideal is the smallest height of a prime containing the ideal, $\text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big) = \text{ht}(Q')$. By definition, Q' must also contain $\sqrt{I\mathcal{O}_{\mathbb{A}^n,p}}$, thus $\text{ht}\Big(\sqrt{I\mathcal{O}_{\mathbb{A}^n,p}}\Big) = \text{ht}(Q') = \text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big)$. Therefore, $\text{ht}\Big(\mathbf{I}(\mathbf{V})\,\mathcal{O}_{\mathbb{A}^n,p}\Big) = \text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big)$. By Lemma 3.2.6, $\mathcal{O}_{\mathbb{A}^n,p}$ is a Noetherian local Cohen–Macaulay ring, so we may apply Proposition 3.2.5 to get $\dim(\mathcal{O}_{\mathbb{A}^n,p}/\mathbf{I}(\mathbf{V})\,\mathcal{O}_{\mathbb{A}^n,p}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p})$.

The algorithms proposed in this manuscript require the input polynomials form a regular sequence in the local ring at the point we wish to compute the intersection multiplicity at. The following theorem provides a correspondence between being a regular sequence and having a zero-dimensional algebraic set. This theorem holds under the assumption the algebraic set in question is irreducible. Although this is implied by the convention in Section 2.1.2 we state this explicitly to avoid any confusion.

Theorem 3.2.8 *Let I be the ideal generated by $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ in the polynomial ring and define $\mathbf{V} = \mathbf{V}(I)$. Suppose $\mathbf{V}$ is non-empty and irreducible, then for any $p \in \mathbf{V}$, $\dim(\mathbf{V}) = 0$ if and only if $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.*

Proof First we will show $\dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}) = 0$ if and only if $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.

By Lemma 3.2.6, $\mathcal{O}_{\mathbb{A}^n,p}$ is a Noetherian local Cohen–Macaulay ring, so we may apply Proposition 3.2.5. If $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ then by Proposition 3.2.3, $\text{depth}\Big(I\mathcal{O}_{\mathbb{A}^n,p}, \mathcal{O}_{\mathbb{A}^n,p}\Big) = n$. By Proposition 3.2.5, $\text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big) = n$ and

$$\dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}) - \text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big) = n - n = 0.$$

Conversely, supppose $f_1, \ldots, f_n$ is not a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$. Let k be the maximal length of a regular sequence of $I\mathcal{O}_{\mathbb{A}^n,p}$ in $\mathcal{O}_{\mathbb{A}^n,p}$, that is, let $k = \text{depth}\Big(I\mathcal{O}_{\mathbb{A}^n,p}, \mathcal{O}_{\mathbb{A}^n,p}\Big)$. By Proposition 3.2.3, we have $k < n$. By Proposition 3.2.5, $\text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big) = k$ and

$$\dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}) - \text{ht}\Big(I\mathcal{O}_{\mathbb{A}^n,p}\Big) = n - k > 0.$$

To conclude the proof we will show $\dim(\mathbf{V}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p})$. Observe $\dim(\mathbf{V}) = \dim(\mathbb{K}[x_1,\ldots,x_n]/\mathbf{I}(\mathbf{V})) = \dim(\mathcal{O}_{\mathbf{V},p}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}/\mathbf{I}(\mathbf{V})\mathcal{O}_{\mathbb{A}^n,p}) = \dim(\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p})$, where the second equality follows from Theorem 11.25 in [2], the third equality follows from $\mathcal{O}_{\mathbf{V},p} \cong \mathcal{O}_{\mathbb{A}^n,p}/\mathbf{I}(\mathbf{V})\mathcal{O}_{\mathbb{A}^n,p}$, and the last equality follows from Lemma 3.2.7.

Chapter 4

Generalizing Fulton's Algorithm

In this chapter, we develop the main results of this manuscript. We start by proving the generalization of Fulton's properties and several other results needed to generalize Fulton's algorithm[1]. Next, we generalize Fulton's algorithm to systems of 3 polynomials in 3 variables. The trivariate version of Fulton's algorithm looks similar to the bivariate version, and hence, is useful for building the readers intuition before moving to the n-variate algorithm. We also discuss an example for which the algorithm of Vrbik et al. fails to compute the intersection multiplicity where as the trivariate version of Fulton's algorithm succeeds. Next, we introduce the definition of the matrix of modular degrees, as we believe all versions of Fulton's algorithm are best understood when viewed through this lens. Using this notion, we explore two examples of Fulton's algorithm (bivariate and trivariate) and use them to illustrate the pattern we seek to generalize, expressed in terms of the matrix of modular degrees. With this in mind, we provide the generalization of Fulton's algorithm. We don't provide any examples in full of this algorithm since for $n > 3$, explicitly writing all steps is tedious and draws attention away from the focus of this manuscript. We refer the reader to the previous trivariate examples and the examples at the end of Chapter 5 if they wish to further explore the generalization of Fulton's algorithm. Lastly, we conclude this chapter with an important optimization for when the input system forms a triangular regular sequence. We will later see that notion of a regular chain, introduced in Chapter 5, satisfies the conditions of a triangular regular sequence. Hence, when it is known that the input system is a regular chain, one may apply this optimization to compute the intersection multiplicity immediately.

4.1 A Generalization of Fulton's Properties

The following theorem gives a generalization of Fulton's Properties for n polynomials in n variables. This generalization of Fulton's Properties was first discovered by the authors of [22] and first proved in [27].

We have modified the statement of (n-1) from that given in [27] and provide our own proof of (n-1) which follows the proof of (2-1) given in [12, Section 3-3] closely. Below are several propositions used in the proof of Fulton's Properties.

[1] A version of this chapter has been published in [23].

Proposition 4.1.1 *Let $\mathbb{K}$ be an algebraically closed field and I an ideal in $\mathbb{K}[x_1, \ldots, x_n]$ such that $\mathbf{V}(I) = \{p_1, \ldots, p_m\}$ for some $p_i \in \mathbb{A}^n(\mathbb{K})$. Then, $\dim_{\mathbb{K}}(\mathbb{K}[x_1, \ldots, x_n]/I) = \sum_{i=1}^{m} \dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p_i}/I\mathcal{O}_{\mathbb{A}^n,p_i}\right)$.*

Proposition 4.1.2 *Let $\mathbb{K}$ be an algebraically closed field and I an ideal in $\mathbb{K}[x_1, \ldots, x_n]$ such that $\mathbf{V}(I) = \{p\}$ for some $p \in \mathbb{A}^n(\mathbb{K})$. Then, $\mathbb{K}[x_1, \ldots, x_n]/I \cong \mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}$.*

Proposition 4.1.3 *Let I be an ideal in $\mathbb{K}[x_1, \ldots, x_n]$. Then $\mathbf{V}(I)$ is a finite set if and only if $\dim_{\mathbb{K}}(\mathbb{K}[x_1, \ldots, x_n]/I)$ is finite.*

Both Proposition 4.1.1 and Proposition 4.1.2 follow as corollaries to Proposition 6 in [12, Section 2-9]. Proposition 4.1.3 follows as a corollary to the Nullstellensatz [12, Section 1-7].

Proposition 4.1.4 *Let $r_1, \ldots, r_d$ form a regular sequence in a Noetherian local ring R, and suppose all r_i are in the maximal ideal, then any permutation of $r_1, \ldots, r_d$ is a regular sequence in R.*

Proposition 4.1.4 can be found in [18, Section 3-1]. With Proposition 4.1.4, to show $f_1, \ldots, f_n$ is not a regular sequence in the local ring at p, it suffices to show there are coefficients $Q_1, \ldots, Q_n$ in the local ring such that $Q_1 f_1 + \ldots Q_n f_n = 0$ and there is some index r such that $Q_r \notin \langle f_1, \ldots, \widehat{f_r}, \ldots, f_n\rangle \mathcal{O}_{\mathbb{A}^n,p}$.

The below proposition is used to prove a generalization of one of Fulton's properties as well as in our generalization of Fulton's algorithm.

Proposition 4.1.5 *Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ where $f_1, \ldots, f_n$ vanish on some $p \in \mathbb{A}^n$. Suppose for some k we have $f_k = q_1 q_2$ for some $q_1, q_2 \in \mathbb{K}[x_1, \ldots, x_n]$ which are not units in $\mathcal{O}_{\mathbb{A}^n,p}$. Then $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ if and only if both $f_1, \ldots, q_1, \ldots, f_n$ and $f_1, \ldots, q_2, \ldots, f_n$ are regular sequences in $\mathcal{O}_{\mathbb{A}^n,p}$.*

Proof We will consider all ideals as ideals to be in $\mathcal{O}_{\mathbb{A}^n,p}$. Suppose $f_1, \ldots, f_n$ is not a regular sequence. We may assume neither $q_1, q_2 \in \langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$ since otherwise the claim clearly holds. Since $f_1, \ldots, f_n$ is not a regular sequence there exists coefficients $Q_1, \ldots, Q_n$ in the local ring and an index r such that

$$\sum_{i=1}^{n} Q_i f_i = 0,$$

and $Q_r \notin \langle f_1, \ldots, \widehat{f_r}, \ldots, f_n\rangle$. We will consider two cases; where $r = k$ and where $r \neq k$.

If $r = k$ write

$$Q_1 f_1 + \ldots + Q_k q_1 q_2 + \ldots + Q_n f_n = 0,$$

and $Q_k \notin \langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$. If $Q_k q_1 \notin \langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$ then q_2 is a zero-divisor since the image of q_2 is not zero modulo $\langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$. If $Q_k q_1 \in \langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$ then q_1 is a zero-divisor since the images of Q_k and q_1 modulo $\langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$ are not zero. Hence, one of $f_1, \ldots, q_1, \ldots, f_n$ or $f_1, \ldots, q_2, \ldots, f_n$ is not a regular sequence.

Suppose $r \neq k$. Since $Q_r \notin \langle f_1, \ldots, f_k, \ldots, \widehat{f_r}, \ldots, f_n\rangle$, Q_r can not be in both $\langle f_1, \ldots, q_1, \ldots, \widehat{f_r}, \ldots, f_n\rangle$ and $\langle f_1, \ldots, q_2, \ldots, \widehat{f_r}, \ldots, f_n\rangle$. Say $Q_r \notin \langle f_1, \ldots, q_1, \ldots, \widehat{f_r}, \ldots, f_n\rangle$. Defining $Q'_i = Q_i$ for all $i \neq k$ and $Q'_k = Q_k q_2$ gives

$$Q'_1 f_1 + \ldots + Q'_k q_1 + \ldots + Q'_n f_n = 0,$$

and $Q'_r \notin \langle f_1, \ldots, q_1, \ldots, \widehat{f_r}, \ldots, f_n\rangle$, and hence $f_1, \ldots, q_1, \ldots, f_n$ is not a regular sequence.

Conversely, suppose one of $f_1, \ldots, q_1, \ldots, f_n$ or $f_1, \ldots, q_2, \ldots, f_n$ is not a regular sequence, say $f_1, \ldots, q_1, \ldots, f_n$. Then there are $Q_1, \ldots, Q_n \in \mathcal{O}_{\mathbb{A}^n,p}$ such that, $Q_1 f_1 + \ldots + Q_k q_1 + \ldots + Q_n f_n = 0$. We will again consider two cases; where $Q_k \notin \langle f_1, \ldots, \widehat{q_1}, \ldots, f_n\rangle$ and where $Q_k \in \langle f_1, \ldots, \widehat{q_1}, \ldots, f_n\rangle$.

If $Q_k \notin \langle f_1, \ldots, \widehat{q_1}, \ldots, f_n\rangle$, then multiplying by q_2 gives us $q_2 Q_1 f_1 + \ldots + Q_k(q_1 q_2) + \ldots + q_2 Q_n f_n = 0$ and hence $q_2 Q_1 f_1 + \ldots + Q_k f_k + \ldots + q_2 Q_n f_n = 0$. Defining $Q'_i = Q_i q_2$ for all $i \neq k$ and $Q'_k = Q_k$ gives us

$$\sum_{i=1}^{n} Q'_i f_i = 0,$$

and $Q_k \notin \langle f_1, \ldots, \widehat{f_k}, \ldots, f_n\rangle$, hence $f_1, \ldots, f_n$ is not a regular sequence.

If $Q_k \in \langle f_1, \ldots, \widehat{q_1}, \ldots, f_n\rangle$ then we can write $Q'_1 f_1 + \ldots + 0 q_1 + \ldots + Q'_n f_n = 0$ for some $Q'_1, \ldots, Q'_n \in \mathcal{O}_{\mathbb{A}^n,p}$. Since $f_1, \ldots, q_1, \ldots, f_n$ is not a regular sequence, there must be some index r such that $Q'_r \notin \langle F_1, \ldots, \widehat{F_r}, \ldots, F_n\rangle$ where $F_i = f_i$ for all $i \neq k$ and $F_k = q_1$. Hence $Q'_r \notin \langle f_1, \ldots, \widehat{f_r}, \ldots, f_n\rangle$. Moreover, we may replace q_1 with f_k without affecting the sum, hence we write $Q'_1 f_1 + \ldots + 0 f_k + \ldots + Q'_n f_n = 0$. Thus, $f_1, \ldots, f_n$ is not a regular sequence.

The following theorem generalizes Fulton's Properties from Proposition 2.3.1.

Theorem 4.1.6 *Let $f_1, \ldots, f_n$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ and let $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$. The $\mathrm{Im}(p;\, f_1, \ldots, f_n)$ satisfies (n-1) to (n-7) where:*

(n-1) $\mathrm{Im}(p;\, f_1, \ldots, f_n) \in \mathbb{N}$ *when* $\mathbf{V}(f_1, \ldots, f_n)$ *is zero-dimensional and* $\mathrm{Im}(p;\, f_1, \ldots, f_n) = \infty$ *otherwise.*

(n-2) $\mathrm{Im}(p;\, f_1, \ldots, f_n) = 0$ *if and only if* $p \notin \mathbf{V}(f_1, \ldots, f_n)$.

(n-3) $\mathrm{Im}(p;\, f_1, \ldots, f_n)$ *is invariant under affine changes of coordinates on* $\mathbb{A}^n$.

(n-4) $\mathrm{Im}(p;\, f_1, \ldots, f_n) = \mathrm{Im}(p;\, f_{\sigma(1)}, \ldots, f_{\sigma(n)})$ *where σ is any permutation on* $\{1, \ldots, n\}$.

(n-5) $\mathrm{Im}(p;\, (x_1 - p_1)^{m_1}, \ldots, (x_n - p_n)^{m_n}) = m_1 \cdots m_n$ *for any* $m_1, \ldots, m_n \in \mathbb{N}$.

(n-6) $\mathrm{Im}(p;\, f_1, \ldots, f_{n-1}, gh) = \mathrm{Im}(p;\, f_1, \ldots, f_{n-1}, g) + \mathrm{Im}(p;\, f_1, \ldots, f_{n-1}, h)$ *for any* $g, h \in \mathbb{K}[x_1, \ldots, x_n]$ *such that $f_1, \ldots, f_{n-1}, gh$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.*

(n-7) $\mathrm{Im}(p;\, f_1, \ldots, f_n) = \mathrm{Im}(p;\, f_1, \ldots, f_{n-1}, f_n + g)$ *for any* $g \in \langle f_1, \ldots, f_{n-1}\rangle$.

Proof of (n-1)

Suppose $\mathbf{V}(f_1,\ldots,f_n)$ is zero-dimensional. We may apply Proposition 4.1.1. to get,

$$\begin{aligned}\dim_{\mathbb{K}}(\mathbb{K}[x_1,\ldots,x_n]/\langle f_1,\ldots,f_n\rangle) &\geq \dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p}\right)\\ &= \mathrm{Im}(p;\, f_1,\ldots,f_n)\,.\end{aligned}$$

Proposition 4.1.3 tells us $\dim_{\mathbb{K}}(\mathbb{K}[x_1,\ldots,x_n]/\langle f_1,\ldots,f_n\rangle)$ must be finite which concludes this direction.

Suppose $\mathbf{V}(f_1,\ldots,f_n)$ has positive dimension and let $F = \mathbf{I}(\mathbf{V}(f_1,\ldots,f_n))$. Since $\langle f_1,\ldots,f_n\rangle \subseteq F$, there is a surjective homomorphism from $\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p}$ onto $\mathcal{O}_{\mathbb{A}^n,p}/F\mathcal{O}_{\mathbb{A}^n,p}$. Thus, $\mathrm{Im}(p;\, f_1,\ldots,f_n) \geq \dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p}/F\mathcal{O}_{\mathbb{A}^n,p}\right)$.

Since $\mathcal{O}_{\mathbb{A}^n,p}/F\mathcal{O}_{\mathbb{A}^n,p} \cong \mathcal{O}_{\mathbf{V}(f_1,\ldots,f_n),p}$ and $\mathcal{O}_{\mathbf{V}(f_1,\ldots,f_n),p} \supseteq \mathbb{K}[x_1,\ldots,x_n]/F$ we get,

$$\mathrm{Im}(p;\, f_1,\ldots,f_n) \geq \dim_{\mathbb{K}}(\mathbb{K}[x_1,\ldots,x_n]/F)\,.$$

Lastly, Proposition 4.1.3 tells us $\dim_{\mathbb{K}}(\mathbb{K}[x_1,\ldots,x_n]/F)$ is finite if and only if $\mathbf{V}(F) = \mathbf{V}(\mathbf{I}(\mathbf{V}(f_1,\ldots,f_n))) = \mathbf{V}(f_1,\ldots,f_n)$ is zero-dimensional. Since $\mathbf{V}(f_1,\ldots,f_n)$ has positive dimension, $\dim_{\mathbb{K}}(\mathbb{K}[x_1,\ldots,x_n]/F)$ must be infinite, which concludes the proof.

Proof of (n-2)

Suppose $p \notin \mathbf{V}(f_1,\ldots,f_n)$. Then there is at least one f_i which does not vanish on p, hence f_i is a unit in $\mathcal{O}_{\mathbb{A}^n,p}$. Thus, $\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle = \langle 0\rangle$ and $\mathrm{Im}(p;\, f_1,\ldots,f_n) = 0$.

Conversely, suppose $p \in \mathbf{V}(f_1,\ldots,f_n)$. Since for any $f \in \langle f_1,\ldots,f_n\rangle$ we have $f(p) = 0$, so $\langle f_1,\ldots,f_n\rangle$ must be contained in $\langle x_1 - p_1,\ldots,x_n - p_n\rangle$. Thus, there is a surjective homomorphism from $\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p}$ onto $\mathcal{O}_{\mathbb{A}^n,p}/\langle x_1 - p_1,\ldots,x_n - p_n\rangle\mathcal{O}_{\mathbb{A}^n,p}$ which implies

$$\dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p}\right) \geq \dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p}/\langle x_1 - p_1,\ldots,x_n - p_n\rangle\mathcal{O}_{\mathbb{A}^n,p}\right).$$

Notice $\mathbf{V}(x_1 - p_1,\ldots,x_n - p_n) = \{p\}$, hence, applying Proposition 4.1.2 gives $\mathcal{O}_{\mathbb{A}^n,p}/\langle x_1,\ldots,x_n\rangle\mathcal{O}_{\mathbb{A}^n,p} \cong \mathbb{K}[x_1,\ldots,x_n]/\langle x_1,\ldots,x_n\rangle \cong \mathbb{K}$. Therefore, when $p \in \mathbf{V}(f_1,\ldots,f_n)$ the dimension of $\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p}$ as a $\mathbb{K}$ vector space is at least 1, and hence non-zero.

Proof of (n-3)

Let T be an affine change of coordinates such that $T(p) = q$ for some $q \in \mathbb{A}^n$ and let $\phi : \mathcal{O}_{\mathbb{A}^n,q} \to \mathcal{O}_{\mathbb{A}^n,p}$ be such that ϕ maps $\frac{f}{g} \mapsto \frac{f\circ T}{g\circ T}$. By [12, Exercise 2.22], ϕ is an isomorphism, hence

$$\mathcal{O}_{\mathbb{A}^n,q}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,q} \cong \mathcal{O}_{\mathbb{A}^n,p}/\langle f_1\circ T,\ldots,f_n\circ T\rangle\mathcal{O}_{\mathbb{A}^n,p},$$

from which the desired statement follows.

Proof of (n-4)

Since $\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p} = \langle f_{\sigma(1)},\ldots,f_{\sigma(n)}\rangle\mathcal{O}_{\mathbb{A}^n,p}$ we get,

$$\begin{aligned}\mathrm{Im}(p;\, f_1,\ldots,f_n) &= \dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_n\rangle\mathcal{O}_{\mathbb{A}^n,p}\right)\\ &= \dim_{\mathbb{K}}\left(\mathcal{O}_{\mathbb{A}^n,p}/\langle f_{\sigma(1)},\ldots,f_{\sigma(n)}\rangle\mathcal{O}_{\mathbb{A}^n,p}\right)\\ &= \mathrm{Im}(p;\, f_{\sigma(1)},\ldots,f_{\sigma(n)})\,.\end{aligned}$$

Proof of (n-5)

We will show $\mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n)^{m_n}) = m_1\cdots m_n$ for any $m_1,\ldots,m_n\in\mathbb{N}$ for any $m_1,\ldots,m_n\in\mathbb{N}$.

Assume (n-6) holds (we will prove (n-6) shortly). Take any $m_1,\ldots,m_n\in\mathbb{N}$. If one such m_i is zero then $(x_i-p_i)^{m_i}$ does not vanish anywhere, and hence $\mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n)^{m_n}) = 0 = m_1\cdots m_n$ by (n-2). Thus, we may assume all $m_1,\ldots,m_n$ are non-zero.

Applying (n-6) repeatedly yields,

$$\begin{aligned}
\mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n)^{m_n}) &= \mathrm{Im}\Big(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n)^{m_n-1}\Big)\\
&+ \mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n))\\
&=\\
&\vdots\\
&= \mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,1)\\
&+ (m_n)\mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n))\\
&= (m_n)\mathrm{Im}(p;\,(x_1-p_1)^{m_1},\ldots,(x_n-p_n))\\
&=\\
&\vdots\\
&= (m_1\cdots m_n)\mathrm{Im}(p;\,(x_1-p_1),\ldots,(x_n-p_n)).
\end{aligned}$$

Let I be the ideal generated by $x_1-p_1,\ldots,x_n-p_n$ in $\mathbb{K}[x_1,\ldots,x_n]$. It suffices to show $\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}\cong\mathbb{K}$ and hence has dimension 1 as a vector space over $\mathbb{K}$. Notice $\mathbf{V}(I)=\{p\}$, hence applying Proposition 4.1.2 gives $\mathcal{O}_{\mathbb{A}^n,p}/I\mathcal{O}_{\mathbb{A}^n,p}\cong\mathbb{K}[x_1,\ldots,x_n]/I$. Since $I=\langle x_1-p_1,\ldots,x_n-p_n\rangle$, it follows that $\mathbb{K}[x_1,\ldots,x_n]/I\cong\mathbb{K}$, which completes the proof.

Proof of (n-6)

Suppose $f_1,\ldots,f_n$ form a regular sequence. Let,

$$\begin{aligned}
\mathcal{O}_{gh} &= \mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_{n-1},gh\rangle\mathcal{O}_{\mathbb{A}^n,p}\\
\mathcal{O}_{g} &= \mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_{n-1},g\rangle\mathcal{O}_{\mathbb{A}^n,p}\\
\mathcal{O}_{h} &= \mathcal{O}_{\mathbb{A}^n,p}/\langle f_1,\ldots,f_{n-1},h\rangle\mathcal{O}_{\mathbb{A}^n,p}.
\end{aligned}$$

We will show,

$$\dim_{\mathbb{K}}\Big(\mathcal{O}_{gh}\Big) = \dim_{\mathbb{K}}\Big(\mathcal{O}_{g}\Big) + \dim_{\mathbb{K}}(\mathcal{O}_h)$$

by showing there is a short exact sequence,

$$0\to\mathcal{O}_h\xrightarrow{\psi}\mathcal{O}_{gh}\xrightarrow{\phi}\mathcal{O}_g\to 0$$

such that ϕ is the natural homomorphism from $\mathcal{O}_{gh}$ to $\mathcal{O}_g$ and ψ maps $\overline{Z}\mapsto\overline{Zg}$ where the bar denotes respective residues and $Z\in\mathcal{O}_{\mathbb{A}^n,p}$.

The map ϕ is clearly surjective. It remains to show $\mathrm{Im}(\psi)=\mathrm{Ker}(\phi)$ and that ψ is injective.

Clearly the $\mathrm{Im}(\psi) \subseteq \mathrm{Ker}(\phi)$. Now take $\overline{Z} \in \mathrm{Ker}(\phi)$. Thus,

$$Z = A_1 f_1 + \ldots + A_{n-1} f_{n-1} + A_n g,$$

for some $A_1, \ldots, A_n \in \mathcal{O}_{\mathbb{A}^n,p}$ and hence $\overline{Z} = \overline{A_n g}$. But $\psi(\overline{A_n}) = \overline{A_n g} = \overline{Z}$ and hence $\mathrm{Im}(\psi) = \mathrm{Ker}(\phi)$.

Lastly, we must show ψ is injective. Suppose $\overline{Z} \in \mathrm{Ker}(\psi)$ hence,

$$Zg = A_1 f_1 + \ldots + A_{n-1} f_{n-1} + A_n gh,$$

for some $A_1, \ldots, A_n \in \mathcal{O}_{\mathbb{A}^n,p}$. Observe,

$$(Z - A_n h)g \in \langle f_1, \ldots, f_{n-1} \rangle \mathcal{O}_{\mathbb{A}^n,p}.$$

Since $f_1, \ldots, f_{n-1}, gh$ is a regular sequence, both $f_1, \ldots, f_{n-1}, g$ and $f_1, \ldots, f_{n-1}, h$ are regular sequences by Proposition 4.1.5. Since $f_1, \ldots, f_{n-1}, g$ is a regular sequence we must have $(Z - A_n h) \in \langle f_1, \ldots, f_{n-1} \rangle \mathcal{O}_{\mathbb{A}^n,p}$. Therefore, $Z \in \langle f_1, \ldots, f_{n-1}, h \rangle \mathcal{O}_{\mathbb{A}^n,p}$ and thus $\overline{Z} = 0$. Therefore, ψ is injective, which concludes the proof.

Proof of (n-7)

Since $\langle f_1, \ldots, f_n \rangle \mathcal{O}_{\mathbb{A}^n,p} = \langle f_1, \ldots, f_{n-1}, f_n + g \rangle \mathcal{O}_{\mathbb{A}^n,p}$ for any $g \in \langle f_1, \ldots, f_{n-1} \rangle \mathcal{O}_{\mathbb{A}^n,p}$ we get,

$$\begin{aligned} \mathrm{Im}(p;\, f_1, \ldots, f_n) &= \dim_{\mathbb{K}} \left(\mathcal{O}_{\mathbb{A}^n,p} / \langle f_1, \ldots, f_n \rangle \mathcal{O}_{\mathbb{A}^n,p} \right) \\ &= \dim_{\mathbb{K}} \left(\mathcal{O}_{\mathbb{A}^n,p} / \langle f_1, \ldots, f_{n-1}, f_n + g \rangle \mathcal{O}_{\mathbb{A}^n,p} \right) \\ &= \mathrm{Im}(p;\, f_1, \ldots, f_{n-1}, f_n + g)\,. \end{aligned}$$

The following corollaries are essential to our generalization of Fulton's algorithm. The first relates the input constraint of our algorithm (the input must either be a regular sequence or contain a unit) to the generalization of Fulton's properties. In essence, it tells us those systems we can apply our algorithm to are exactly those systems which have finite intersection multiplicity. That is, the constraint that the input polynomials form a regular sequence (or contain a unit) does not limit the applicability of our algorithm. The next corollary tells us that the operations we will perform to rewrite our input in the generalization of Fulton's algorithm will preserve regular sequences, which, when combined with Proposition 4.1.5, is essential in order to satisfy the regular sequence constraint with each recursive call.

Corollary 4.1.7 *Let $f_1, \ldots, f_n$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ and fix $p \in \mathbb{A}^n$. Then $\mathrm{Im}(p;\, f_1, \ldots, f_n)$ is finite exactly when either $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ or when at least one f_i does not vanish at p.*

Proof By (n-2), at least one f_i does not vanish on p exactly when $\mathrm{Im}(p;\, f_1, \ldots, f_n) = 0$. Now suppose that $\mathrm{Im}(p;\, f_1, \ldots, f_n) > 0$. By ($n$-1), $\mathrm{Im}(p;\, f_1, \ldots, f_n)$ is finite when $\mathbf{V}(f_1, \ldots, f_n)$ is zero-dimensional. Since $\mathrm{Im}(p;\, f_1, \ldots, f_n) > 0$, no f_i vanishes at p, we may apply Theorem 3.2.8 which states $\mathbf{V}(f_1, \ldots, f_n)$ is zero-dimensional exactly when $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.

Corollary 4.1.8 *Let $f_1, \ldots, f_n$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ and fix $p \in \mathbb{A}^n$. The operations to the polynomial system $f_1, \ldots, f_n$ which preserve its intersection multiplicity at p as described in Theorem 4.1.6 also preserve the property of being a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$. Namely the operations (n-3), (n-4), and (n-7), preserve regular sequences.*

Proof Applying any of (n-3), (n-4), or (n-7) amounts to rewriting $f_1, \ldots, f_n$ as a system with equal intersection multiplicity at p. Applying (n-1), (n-2), and Theorem 3.2.8 shows the new system is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ exactly when $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.

For property (n-4) in the above proof, it would also suffice to apply Proposition 4.1.4.

4.2 Trivariate Fulton's Algorithm

In this section we show how the n-variate generalization of Fulton's properties can be used to create a procedure to compute intersection multiplicity in the trivariate case. Later we will see this approach generalizes to the n-variate case, although, it is helpful to first understand the algorithm's behaviour in the trivariate case.

This procedure is not complete since the computations of lines 19 and 22, analogous to line 12 of Algorithm 1, do not necessarily preserve intersection multiplicity under (n-7). When this is the case, the procedure returns Fail to signal an error.

When the procedure succeeds, we obtain a powerful tool for computing intersection multiplicities in the trivariate case. This allows us to compute intersection multiplicities that previously could not be computed by other, standard basis-free approaches, namely that of [1] and [27].

Throughout this section we assume $p \in \mathbb{A}^3$ is the origin.

Definition 4.2.1 (Modular Degree) *Let f be in $\mathbb{K}[x, y, z]$ where $x \succ y \succ z$. We define the modular degree of f with respect to a variable $v \in V$ as $\deg_v(f \mod \langle V_{<v} \rangle)$, where $V = \{x, y, z\}$ is the set of variables and $V_{<v}$ is the set of all variables less than v in the given ordering. If $V_{<v} = \emptyset$, we define the modular degree of f with respect to v to be the degree of f with respect to v. Write* $\operatorname{moddeg}(f, v)$ *to denote the modular degree of f with respect to v.*

Remark The definition of modular degree can be generalized to a point $p = (p_1, p_2, p_3) \in \mathbb{A}^3$ by replacing $V_{<v}$ with $V_{<v,p} = \{x - p_1, y - p_2, z - p_3\}$ in Definition 4.2.1.

The modular degree is used to generalize the computation of r, s in Algorithm 1. If we fix some variable v, the modular degree with respect to v is simply the degree in v of a polynomial modulo all variables smaller than v in a given ordering.

Below we formally define cases in terms of the colour they are highlighted with in Algorithm 2. Although not necessary, using a name to distinguish between cases rather than a set of conditions makes the proof far more readable, especially when the set of cases is small, as is the case for trivariate intersection multiplicity.

In the n-variate case, we will see that some of these cases are not distinct and in fact, instances of the same case. We will describe this in more detail later. For now, we make this distinction to illustrate the similarities to Algorithm 1 and to help the reader build intuition for this procedure in a more general setting.

Definition 4.2.2 (Colour Cases) *Consider $f, g, h \in \mathbb{K}[x, y, z]$.*

1. *We say we are in the red case if one of f, g, h does not vanish on p.*
2. *We say we are in the blue case if:*
   (a) *We are not in the red case.*
   (b) *The modular degrees of f, g, h in x are in ascending order.*
   (c) *At least one of f or g has modular degree in x greater than zero.*
3. *We say we are in the orange case if:*
   (a) *We are not in the red case.*
   (b) *The modular degrees of f, g, h in x are in ascending order.*
   (c) *Both f and g have modular degrees in x less than zero.*
4. *We say we are in the yellow case if:*
   (a) *We are in the orange case.*
   (b) *The modular degrees of f, g, h in x and the modular degrees of f, g in y are in ascending order.*
   (c) *The modular degree of f in y is less than zero.*
5. *We say we are in the pink case if:*
   (a) *We are in the orange case.*
   (b) *The modular degrees of f, g, h in x and the modular degrees of f, g in y are in ascending order.*
   (c) *The modular degree of f in y is greater than zero.*

Algorithm 2 generalizes Fulton's approach in the trivariate case. The key to generalizing Fulton's bivariate intersection multiplicity algorithm to the case of 3 polynomials in 3 variables is generalizing the splitting step. When the yellow case holds, we can split the intersection multiplicity computation into the sum of smaller intersection multiplicity computations. Thus, the rest of the algorithm is designed to reduce to the yellow case, or return Fail, in finitely many iterations.

At this time there is no natural way to characterize all cases where Algorithm 2 fails since it is difficult to determine before runtime which cases will be reached after rewriting and splitting. Namely, it is difficult to characterize all inputs which will eventually reach a branch which satisfies the conditions of the pink case. Given an input that does satisfy the conditions of pink case, it is easy to check whether Algorithm 2 fails in that iteration, as we will see in the proof of Theorem 4.2.3.

Theorem 4.2.3 *Let $f, g, h \in \mathbb{K}[x, y, z]$ be either a regular sequence in $\mathcal{O}_{\mathbb{A}^3,p}$ or be such that at least one of f, g, h is a unit in $\mathcal{O}_{\mathbb{A}^3,p}$. Algorithm 2 correctly computes the intersection multiplicity of f, g, h at p or returns Fail.*

Algorithm 2: Trivariate Fulton's Algorithm

Function $\mathrm{im}_3(p; f, g, h)$

Input: Let: $x \succ y \succ z$

1. $p \in \mathbb{A}^3$ the origin.
2. $f, g, h \in \mathbb{K}[x, y, z]$ such that f, g, h form a regular sequence in $\mathcal{O}_{\mathbb{A}^3,p}$ or at least one of f, g, h is a unit in $\mathcal{O}_{\mathbb{A}^3,p}$.

Output: $\mathrm{Im}(p;\ f, g, h)$ or Fail

if $f(p) \neq 0$ **or** $g(p) \neq 0$ **or** $h(p) \neq 0$ **then** /* Red */
  return 0
$r_y \leftarrow \mathrm{moddeg}(f, y)$, $r_x \leftarrow \mathrm{moddeg}(f, x)$
$s_y \leftarrow \mathrm{moddeg}(g, y)$, $s_x \leftarrow \mathrm{moddeg}(g, x)$
$t_y \leftarrow \mathrm{moddeg}(h, y)$, $t_x \leftarrow \mathrm{moddeg}(h, x)$
Reorder f, g, h so that $r_x \leq s_x \leq t_x$ /* Green */
if $r_x < 0$ **and** $s_x < 0$ **then** /* Orange */
  Reorder f, g so that $r_y \leq s_y$ /* Green */
  if $r_y < 0$ **then** /* Yellow */
    $m_h \leftarrow max(m \in \mathbb{N} \mid h \mod \langle y, z\rangle = x^m(a_0 + a_1 x + \ldots))$
    $q_f \leftarrow \mathrm{quo}(f, z; z)$
    $q_g \leftarrow \mathrm{quo}(g(x, y, 0), y; y)$
    return $\mathrm{im}_3(p; q_f, g, h) + \mathrm{im}(p; q_g, h(x, y, 0)) + m_h$
  else /* Pink */
    $L_f \leftarrow \mathrm{lc}(f(x, y, 0); y)$
    $L_g \leftarrow \mathrm{lc}(g(x, y, 0); y)$
    if $L_f(p) \neq 0$ **then**
      $g' \leftarrow L_f g - y^{s_y - r_y} L_g f$
      return $\mathrm{im}_3(p; f, g', h)$
    else if $L_f \mid L_g$ **then**
      $g' \leftarrow g - y^{s_y - r_y} \frac{L_g}{L_f} f$
      return $\mathrm{im}_3(p; f, g', h)$
    else
      return Fail
else /* Blue */
  if $r_x < 0$ **then**
    $h' \leftarrow \mathrm{lc}(g(x, 0, 0); x)h - x^{t_x - s_x}\mathrm{lc}(h(x, 0, 0); x)g$
    return $\mathrm{im}_3(p; f, g, h')$
  else
    $g' \leftarrow \mathrm{lc}(f(x, 0, 0); x)g - x^{s_x - r_x}\mathrm{lc}(g(x, 0, 0); x)f$
    $h' \leftarrow \mathrm{lc}(f(x, 0, 0); x)h - x^{t_x - r_x}\mathrm{lc}(h(x, 0, 0); x)f$
    return $\mathrm{im}_3(p; f, g', h')$

Proof By (n-3) we may assume p is the origin. By Corollary 4.1.7 we have $\mathrm{Im}(p;\ f,g,h) \in \mathbb{N}$.

To prove termination we induct on $\mathrm{Im}(p;\ f,g,h)$ and show that when Algorithm 2 does not fail, we can either compute $\mathrm{Im}(p;\ f,g,h)$ directly or strictly decrease $\mathrm{Im}(p;\ f,g,h)$ through splitting.

Suppose $\mathrm{Im}(p;\ f,g,h) = 0$, then by ($n$-2), one of f,g,h does not vanish on p, hence, Algorithm 2 correctly returns zero. Assume that $\mathrm{Im}(p;\ f,g,h) = N$ for some positive $N \in \mathbb{N}$.

By (n-4) and Corollary 4.1.8, we may reorder f,g,h so that their modular degrees with respect to x are in ascending order.

Suppose f,g, and h satisfy the conditions of the blue case, that is, at most one polynomial has modular degree in x less than zero. Depending on how many polynomials have modular degree in x less than zero, we perform slightly different computations, since there is no need to reduce a modular degree in x of a polynomial that already has modular degree in x less than zero. Notice the substitutions in the blue case preserve intersection multiplicity by (n-7) and regular sequences by Corollary 4.1.8. Since the modular degrees in x of the resulting polynomials are strictly decreasing, we will reach the orange case in finitely many iterations.

By (n-4) and Corollary 4.1.8, we may reorder f,g so that their modular degrees with respect to y are in ascending order.

Suppose f,g, and h satisfy the conditions of the pink case. Define,

$$L_f = \mathrm{lc}(f(x,y,0);y)$$

$$L_g = \mathrm{lc}(g(x,y,0);y).$$

If L_f is not a unit in $\mathcal{O}_{\mathbb{A}^n,p}$ and does not divide L_g, Algorithm 2 returns Fail since (n-7) cannot be applied.

Suppose either $L_f(p) \neq 0$ or $L_f \mid L_g$. Then the respective substitution of g with g' preserves the intersection multiplicity by (n-7) and regular sequences by Corollary 4.1.8. Moreover, if g' is the polynomial resulting from either of the respective computations, then $\mathrm{moddeg}(g',y) < \mathrm{moddeg}(g,y)$ and $\mathrm{moddeg}(g',x) < 0$. The latter statement follows from both f and g having modular degree in x less than zero as a result of being in the orange case. Since the modular degree of g' with respect to y strictly decreases, we will reach the yellow case or return Fail in finitely many iterations.

Suppose f,g, and h satisfy the conditions of the yellow case. Since $\mathrm{moddeg}(f,x) < 0$, $\mathrm{moddeg}(f,y) < 0$, f is non-zero, and f vanishes at the origin, we have $z \mid f$.

By Proposition 4.1.5, the sequence z,g,h is regular, hence $g(x,y,0)$ is non-zero and vanishes at the origin. Since $\mathrm{moddeg}(g,x) < 0$ holds, we have $y \mid g(x,y,0)$.

Let $q_f = \mathrm{quo}(f,z;z), q_g = \mathrm{quo}(g(x,y,0),y;y), m_h = max(m \in \mathbb{Z}^+ \mid h(x,0,0) \equiv 0 \mod \langle x^m \rangle)$ and write $f = zq_f$, $g(x,y,0) = yq_g$. By (n-6) and Proposition 4.1.5, it is correct to compute:

$$\begin{aligned}
\mathrm{Im}(p;\, f,g,h) &= \mathrm{Im}\Big(p;\, q_f,g,h\Big) + \mathrm{Im}(p;\, z,g,h)\\
&= \mathrm{Im}\Big(p;\, q_f,g,h\Big) + \mathrm{Im}(p;\, z,g(x,y,0),h(x,y,0))\\
&= \mathrm{Im}\Big(p;\, q_f,g,h\Big) + \mathrm{Im}\Big(p;\, z,q_g,h(x,y,0)\Big) + \mathrm{Im}(p;\, z,y,h(x,y,0))\\
&= \mathrm{Im}\Big(p;\, q_f,g,h\Big) + \mathrm{Im}\Big(p;\, z,q_g,h(x,y,0)\Big) + \mathrm{Im}(p;\, z,y,h(x,0,0))\\
&= \mathrm{Im}\Big(p;\, q_f,g,h\Big) + \mathrm{Im}\Big(p;\, q_g,h(x,y,0)\Big) + m_h.
\end{aligned}$$

Since m_h is a positive integer, we have:

$$\mathrm{Im}\Big(p;\, q_f,g,h\Big), \mathrm{Im}\Big(p;\, q_g,h(x,y,0)\Big) < \mathrm{Im}(p;\, f,g,h) = N.$$

Thus, when Algorithm 2, called on the input q_f, g, h, does not fail, termination follows from induction.

To illustrate the utility of this approach we will work through an example where the available standard basis-free techniques (the algorithm of Vrbik et al.) used to compute intersection multiplicity fail. A full description of these techniques can be found in [1] and [27], although we give a brief overview below.

Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ such that $\mathbf{V}(f_1, \ldots, f_n)$ is a zero-dimensional, that is, $\mathrm{Im}(p;\, f_1, \ldots, f_n) \in \mathbb{N}$, and at least one of $f_1, \ldots, f_n$, say f_n is non-singular at p. Theorem 1 of [1], states that when the above conditions hold, and under an additional transversality constraint between $\mathbf{V}(f_1, \ldots, f_{n-1})$ and $\mathbf{V}(f_n)$, an n-variate intersection multiplicity can be reduced to an $(n-1)$-variate intersection multiplicity computation.

In [27], the above reduction is combined with an additional reduction procedure referred to as cylindrification. The idea behind this second reduction procedure is to use pseudo-division by a polynomial, say f_n, to reduce the degree of $f_1, \ldots, f_{n-1}$ with respect to some variable, say x_n. The cylindrification procedure assumes that f_n has a term containing x_n with a non-zero coefficient invertible in $\mathcal{O}_{\mathbb{A}^n,p}$.

The following example contains 3 polynomials which are singular at p, hence the above reduction cannot be applied. Moreover, one can check that applying cylindrification does not reduce the input in a way that the first reduction criterion holds. Hence, the current standard basis-free techniques fail. Additionally, this can be verified using the Maple implementation of the techniques in [27].

Example Compute $\mathrm{Im}\Big(p;\, zy^2, y^5 - z^2, x^5 - y^2\Big)$ using Algorithm 2.

Notice, $zy^2, y^5 - z^2, x^5 - y^2$ form a regular sequence. We compute the modular degrees with respect to x: $r_x < 0, s_x < 0, t_x = 5$, hence, we begin in the orange case. Since additionally, $r_y < 0$, we are in the yellow case and the computation reduces to:

$$\mathrm{Im}\Big(p;\, zy^2, y^5 - z^2, x^5 - y^2\Big) = \mathrm{Im}\Big(p;\, y^2, y^5 - z^2, x^5 - y^2\Big) + \mathrm{Im}\Big(p;\, y^4, x^5 - y^2\Big) + 5.$$

Start with $\mathrm{Im}\left(p;\, y^4, x^5 - y^2\right)$, applying Fulton's bivariate algorithm we get,

$$\begin{aligned}\mathrm{Im}\left(p;\, y^4, x^5 - y^2\right) &= \mathrm{Im}\left(p;\, y^3, x^5 - y^2\right) + 5\\ &= \mathrm{Im}\left(p;\, y^2, x^5 - y^2\right) + 10\\ &= \mathrm{Im}\left(p;\, y, x^5 - y^2\right) + 15\\ &= 20.\end{aligned}$$

Next we compute $\mathrm{Im}\left(p;\, y^2, y^5 - z^2, x^5 - y^2\right)$. Here we have modular degrees in x: $r_x < 0, s_x < 0, t_x = 5$, thus we are in the orange case. Computing the modular degrees in y we get: $r_y = 2, s_y = 5$, hence we enter the pink case. The leading coefficient in y of y^2 evaluated at $z = 0$ is a unit, hence the pink case computation is valid. Thus, let $g' = (y^5 - z^2) - y^3(y^2) = -z^2$ and compute $\mathrm{Im}\left(p;\, y^2, -z^2, x^5 - y^2\right)$.

Computing the modular degrees with respect to y we get: $r_y = 2, s_z < 0$, hence we reorder y^2 and $-z^2$. Again, we enter the yellow case and the computation reduces to

$$\mathrm{Im}\left(p;\, -z^2, y^2, x^5 - y^2\right) = \mathrm{Im}\left(p;\, -z, y^2, x^5 - y^2\right) + \mathrm{Im}\left(p;\, y, x^5 - y^2\right) + 5.$$

Clearly $\mathrm{Im}\left(p;\, y, x^5 - y^2\right) = 5$ by Fulton's bivariate algorithm. The computation $\mathrm{Im}\left(p;\, -z, y^2, x^5 - y^2\right)$ immediately satisfies the yellow case, hence we may split,

$$\begin{aligned}\mathrm{Im}\left(p;\, -z, y^2, x^5 - y^2\right) &= \mathrm{Im}\left(p;\, -1, y^2, x^5 - y^2\right) + \mathrm{Im}\left(p;\, y, x^5 - y^2\right) + 5\\ &= 0 + 5 + 5\\ &= 10\end{aligned}$$

Combining the intermediate computations, we get,

$$\mathrm{Im}\left(p;\, zy^2, y^5 - z^2, x^5 - y^2\right) = 45.$$

4.3 The Matrix of Modular Degrees

Before we extend Fulton's algorithm to the n-variate setting, it is helpful to view both the bivariate and trivariate versions of Fulton's algorithm through the more familiar lens of matrices. In this section, we generalize the notion of modular degree and define the matrix of modular degrees, which we then use it to illustrate the pattern we seek to generalize in the n-variate version of the algorithm.

Throughout this section we assume $p \in \mathbb{A}^n$ is the origin

Definition 4.3.1 (Modular Degree) *Let f be in $\mathbb{K}[x_1, \ldots, x_n]$ where $x_1 \succ \ldots \succ x_n$. We define the modular degree of f with respect to a variable $v \in V$ as* $\deg_v(f \mod \langle V_{<v}\rangle)$*, where $V = \{x_1, \ldots, x_n\}$ is the set of variables and $V_{<v}$ is the set of all variables less than v in the given ordering. If $V_{<v} = \emptyset$, we define the modular degree of f with respect to v to be the degree of f with respect to v. Write* $\mathrm{moddeg}(f, v)$ *to denote the modular degree of f with respect to v.*

Remark The definition of modular degree can be generalized to a point $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$ by replacing $V_{<v}$ with $V_{<v,p} = \{x_i - p_i \mid x_i < v, x_i \in V\}$ in Definition 4.3.1.

Definition 4.3.2 *The matrix of modular degrees of* $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ *is the matrix whose i-th, j-th entry is* $\operatorname{moddeg}(f_i, x_j)$.

The following examples use the matrix of modular degrees to illustrate how the rewriting processes of Algorithms 1 and 2 are used to obtain a system of polynomials which satisfy the respective conditions for splitting.

Example Let $f, g \in \mathbb{K}[x, y]$ be given by $f = x + y, g = x^3$. Suppose we wish to compute the intersection multiplicity at p, the origin by calling $\operatorname{im}(f, g)$. Let f_1 and f_2 in the definition of modular degree be equal to f and g respectively. Moreover, let x_1 and x_2 in the definition of modular degree be equal to x, y, z respectively. Then the matrix of modular degrees corresponding to f, g is:

$$r = \begin{bmatrix} 1 & 1 \\ 3 & 0 \end{bmatrix},$$

where the i-th row corresponds to the polynomial f_i and the j-th column corresponds to the variable x_j. Hence, (i, j)-th entry is the modular degree of f_i with respect to x_j.

Fulton's algorithm begins by computing

$$g' = g - x^2 f = x^3 - x^2(x + y) = -x^2 y.$$

By (2-7) we may redefine g as g'. Hence, we consider the system given by $f = x + y, g = -x^2 y$. After reordering f and g by modular degree in x, the matrix of modular degrees is now:

$$r = \begin{bmatrix} -\infty & 1 \\ 1 & 1 \end{bmatrix}.$$

Since the modular degree of f with respect to x is now less than zero, Fulton's algorithm splits as follows.

$$\begin{aligned} &\operatorname{Im}(p;\ f, g) \\ &= \operatorname{Im}\Big(p;\ -x^2 y, x + y\Big) \\ &= \operatorname{Im}\Big(p;\ -x^2, x + y\Big) + \operatorname{Im}(p;\ y, x + y) \\ &= \operatorname{Im}\Big(p;\ -x^2, x + y\Big) + 1. \end{aligned}$$

Proceeding in this way it is easy to see $\operatorname{Im}\Big(p;\ -x^2, x + y\Big) = 2$ and hence $\operatorname{Im}(p;\ f, g) = 3$.

Example Let $f, g, h \in \mathbb{K}[x, y, z]$ be given by $f = x^2, g = (x + 1)y + x^3, h = y^2 + z + x^3$. Suppose we wish to compute the intersection multiplicity at p, the origin by calling $\operatorname{im}_3(f, g, h)$. Let f_1, f_2, f_3 in the definition of modular degree be equal to f, g, h respectively. Moreover, let x_1, x_2, x_3 in the definition of modular degree be equal to x, y, z respectively. Then the matrix of modular degrees corresponding to f, g, h is:

$$r = \begin{bmatrix} 2 & 0 & 0 \\ 3 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix},$$

where the i-th row corresponds to the polynomial f_i and the j-th column corresponds to the variable x_j. Hence, (i, j)-th entry is the modular degree of f_i with respect to x_j.

We begin in the blue case. Since f has minimal modular degree with respect to x, it is chosen as a pivot and is used to reduce the modular degrees of g and h with respect to x. Write

$$g' = g - xf = (x+1)y + x^3 - x^3 = (x+1)y,$$

and

$$h' = h - xf = y^2 + z + x^3 - x^3 = y^2 + z.$$

By (n-7) we may redefine g as g' and h as h'. Hence, we consider the system given by $f = x^2, g = (x+1)y, h = y^2 + z$. The matrix of modular degrees is now:

$$r = \begin{bmatrix} 2 & 0 & 0 \\ -\infty & 1 & 0 \\ -\infty & 2 & 1 \end{bmatrix},$$

and after reordering f, g, h by modular degree with respect to x, we have $f = (x+1)y, g = y^2 + z, h = x^2$, with matrix of modular degrees:

$$r = \begin{bmatrix} -\infty & 1 & 0 \\ -\infty & 2 & 1 \\ 2 & 0 & 0 \end{bmatrix}.$$

We now satisfy all conditions necessary to enter the orange case since both f and g have modular degree with respect to x less than zero. Since the modular degree of f with respect to y is greater than zero, we are in the pink case.

Since f has minimal modular degree with respect to y, and moreover the leading coefficient of $f \mod \langle z \rangle$ is $x + 1$ which is invertible in the local ring at the origin, we may choose f as a pivot and use it to reduce the modular degree of g with respect to y. Write

$$g' = (x+1)g - yf = (x+1)y^2 + (x+1)z - (x+1)y^2 = (x+1)z.$$

Redefining g as g' and reordering by modular degree in y, we have $f = (x+1)z, g = (x+1)y, h = x^2$, and the matrix of modular degrees is now:

$$r = \begin{bmatrix} -\infty & -\infty & 1 \\ -\infty & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix}.$$

Since the modular degree of f and g with respect to x and the modular degree of f with respect to y are all less than zero, we enter the yellow case. Thus, we may split computations and conclude.

$$\begin{aligned} &\mathrm{Im}(p;\ f, g, h) \\ &= \mathrm{Im}(p;\ x+1, g, h) + \mathrm{Im}(p;\ z, g, h) \\ &= \mathrm{Im}(p;\ x+1, g, h) + \mathrm{Im}(p;\ z, x+1, h) + \mathrm{Im}(p;\ z, y, h) \\ &= 0 + 0 + 2. \end{aligned}$$

Notice in each of the above examples, we reduce to the splitting case when the matrix of modular degrees has all entries above the anti-diagonal equal to negative infinity. When this occurs, we can write the intersection multiplicity as the sum of smaller intersection multiplicity computations and hence, make progress towards termination. Thus, when we explore the n-variate case, it is useful to think of the rewriting steps in terms of the matrix of modular degrees. In order to reduce to the splitting case, we will iterate through the columns of the matrix of modular degrees, applying the generalization Fulton's properties to reduce all entries above the anti-diagonal to negative infinity (when possible). For the trivariate version of Fulton's algorithm, the blue case indicates we are working in the first column and the pink case indicates we are working in the second column. We also note, as shown in the proof of Theorem 4.2.3, that our modifications to a column may change entries in the columns to its right but will never modify entries in any column to its left, hence why we iterate left to right. We will describe this pattern in more detail for the n-variate algorithm, in the following section.

4.4 Generalized Fulton's Algorithm

In this section, we give a generalization of Algorithm 1 using properties (n-1) to (n-7). Unfortunately, the natural generalization using these properties does not characterize intersection multiplicities as in the bivariate case. This is because the substitutions in lines 18 and 20 of Algorithm 3 do not necessarily preserve intersection multiplicity in the n-variate case. Namely, if a particular leading coefficient used in the substitution is not invertible in the local ring, (n-7) may not be applicable. In the bivariate case, this was not an issue since all leading coefficients considered in such a computation were units in the local ring. When such a case arises, other techniques must be used to complete the computation, and hence, our generalization will signal a failure. Throughout this section we assume $p \in \mathbb{A}^n$ is the origin.

Unlike in the trivariate case, it is no longer practical to divide the algorithm into cases. Moreover, we will see that dividing the algorithm into such cases does not accurately reflect the structure of the procedure. The main reason for this, as alluded to in Section 4.3, is that several of the cases we encountered in the past are instances of the same, more general case.

Roughly speaking, Algorithm 3 can be divided into 2 key parts. The first is the main loop which rewrites the input using (n-4) and (n-7). The second is the splitting part, which occurs as a result of the main loop successfully terminating.

The purpose of the main loop, in the j-th iteration, is to create $n - j$ polynomials with modular degrees less than zero in x_j and in any variable larger than x_j. When we examine Algorithm 2 in this context, the requirements to enter the orange or yellow case were simply conditions necessary to move forward an iteration in the main loop. Moreover, the substitutions used in the blue and pink case were separate instances of the same process, which was used to reduce modular degrees for different iterations of the main loop. We highlight line 9 of Algorithm 3 with the colour orange to illustrate the similarities between moving forward an iteration in the loop and satisfying the orange case in Algorithm 2.

Recall in Algorithm 2 there were several possible substitutions that could be performed in the blue case, the deciding factor being, how many of the input polynomials had modular degree in x less than zero. Extending this to the context of the n-variate algorithm, in each iteration of the main loop, we check how many polynomials already satisfy the condition required to

move forward an iteration. As in the blue case, this determines how many substitutions to perform, and which polynomial will be used as a pivot to reduce the modular degrees of the remaining polynomials. To illustrate these similarities, we highlight line 13 of Algorithm 3 with the colour blue.

When the main loop terminates, assuming the procedure did not fail, our input system will have a of triangular shape with respect to modular degrees. That is, upon successful termination of the main loop, any entry in the matrix of modular degrees which lies above the anti-diagonal will be negative infinity. Since there are no entries above the anti-diagonal in the last column of the matrix of modular degrees, the procedure need only modify the first $n-1$ columns. Hence, Algorithm 3 computes only an $n \times (n-1)$ matrix of modular degrees, omitting the last column to avoid unnecessary computations. Lemma 4.4.1 describes the implications of this triangular shape in terms of splitting intersection multiplicity computations. To illustrate the similarities between this splitting procedure, and the procedure used in the yellow case of Algorithm 2, we highlight line 24 of Algorithm 3 with the colour yellow.

As in the trivariate case, we cannot clearly characterize all cases for which Algorithm 3 fails before runtime, due to the difficulty in determining how an input will be rewritten and split. Nonetheless, it is still easy to determine whether an input will cause Algorithm 3 to fail in a given iteration of the main loop, as described in the proof of Theorem 4.4.3.

In the following lemma, the map J maps the index of a row to the column index of the corresponding entry on the anti-diagonal of a $n \times n$ matrix. That is, for any $n \times n$ matrix M, the entry $M_{i,J(i)}$ lies on the anti-diagonal of M where $i \in \{1, \ldots, n\}$.

Lemma 4.4.1 *Let $f_1, \ldots, f_n$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ which form a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ where p is the origin. Let $V = \{x_1, \ldots, x_n\}$ and let $V_{>v} = \{x_i \in V \mid x_i > v\}$. Define the map $J : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that $J(i) = n - i + 1$.*

Suppose for all $i = 1, \ldots, n-1$ we have $\operatorname{moddeg}(f_i, v) < 0$ for all $v \in V_{>x_{J(i)}}$. Then, we have $x_{J(i)} \mid f_i(x_1, \ldots, x_{J(i)}, 0, \ldots, 0)$. Moreover, if we define $q_i = quo(f_i(x_1, \ldots, x_{J(i)}, 0, \ldots, 0), x_{J(i)}; x_{J(i)})$ then,

$$\begin{aligned} \operatorname{Im}(p;\, f_1, \ldots, f_n) &= \operatorname{Im}(p;\, q_1, f_2, \ldots, f_n) + \operatorname{Im}(p;\, x_n, q_2, \ldots, f_n) \\ &+ \ldots + \operatorname{Im}(p;\, x_n, \ldots, x_{J(i)+1}, q_i, f_{i+1}, \ldots, f_n) + \ldots \\ &+ \operatorname{Im}(p;\, x_n, x_{n-1}, \ldots, q_{n-1}, f_n) + m_n \end{aligned}$$

where $m_n = max(m \in \mathbb{Z}^+ \mid f_n(x_1, 0, \ldots, 0) \equiv 0 \mod \langle x_1^m \rangle)$.

Proof First we will show that we can write $f_i(x_1, \ldots, x_{J(i)}, 0, \ldots, 0) = x_{J(i)} q_i$ for all $i = 1, \ldots, n-1$.

Suppose $x_n, \ldots, x_{J(i)+1}, f_i, \ldots, f_n$ is a regular sequence for some $1 \leq i < n$. The hypothesis $\operatorname{moddeg}(f_i, x_1), \ldots, \operatorname{moddeg}(f_i, x_{J(i)-1}) < 0$ and the fact that f_i is regular modulo $\langle x_{J(i)+1}, \ldots, x_n \rangle$ and vanishes at the origin implies $x_{J(i)}$ divides $f_i(x_1, \ldots, x_{J(i)}, 0, \ldots, 0)$.

To show $x_n, \ldots, x_{J(i)+1}, f_i, \ldots, f_n$ is a regular sequence for all $1 \leq i < n$, it suffices to show $x_n, f_2, \ldots, f_n$ is a regular sequence, since repeated applications of Proposition 4.1.5, and the above implication will yield the desired result.

Algorithm 3: Generalized Fulton's Algorithm

Function $\mathrm{im}_n(p; f_1, \ldots, f_n)$

Input: Let: $x_1 \succ \ldots \succ x_n$.

1. $p \in \mathbb{A}^n$ the origin.
2. $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ such that $f_1, \ldots, f_n$ form a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ or at least one such f_i is a unit in $\mathcal{O}_{\mathbb{A}^n,p}$.

Output: $\mathrm{Im}(p;\ f_1, \ldots, f_n)$ or Fail

if $f_i(p) \neq 0$ *for any* $i = 1, \ldots, n$ **then** /* Red */
  return 0
if $n = 1$ **then** /* Compute multiplicity */
  return $\max(m \in \mathbb{Z}^+ \mid f_n \equiv 0 \mod \langle x_1^m \rangle)$
for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, n-1$ **do**
    $r_j^{(i)} \leftarrow \mathrm{moddeg}(f_i, x_j)$
for $j = 1, \ldots, n-1$ **do** /* Orange */
  Reorder $f_1, \ldots, f_{n-j+1}$ so that $r_j^{(1)} \leq \ldots \leq r_j^{(n-j+1)}$ /* Green */
  $m \leftarrow \min(i \mid r_j^{(i)} > 0)$ or $m \leftarrow \infty$ if no such i exists
  if $m \leq (n-j)$ **then**
    for $i = m+1, \ldots, n-j+1$ **do** /* Blue */
      $d \leftarrow r_j^{(i)} - r_j^{(m)}$
      $L_m \leftarrow \mathrm{lc}(f_m(x_1, \ldots, x_j, 0, \ldots, 0); x_j)$
      $L_i \leftarrow \mathrm{lc}(f_i(x_1, \ldots, x_j, 0, \ldots, 0); x_j)$
      if $L_m(p) \neq 0$ **then**
        $f_i' \leftarrow L_m f_i - x_j^d L_i f_m$
      else if $L_m \mid L_i$ **then**
        $f_i' \leftarrow f_i - x_j^d \frac{L_i}{L_m} f_m$
      else
        return Fail
    return $\mathrm{im}_n(p; f_1, \ldots, f_m, f_{m+1}', \ldots, f_{n-j+1}', \ldots, f_n)$
/* Yellow */
for $i = 1, \ldots, n-1$ **do**
  $q_i \leftarrow \mathrm{quo}(f_i(x_1, \ldots, x_{n-i+1}, 0, \ldots, 0), x_{n-i+1}; x_{n-i+1})$
return
$\mathrm{im}_n(p; q_1, f_2, \ldots, f_n)$
$+\ \mathrm{im}_{n-1}(p; q_2(x_1, \ldots, x_{n-1}, 0), \ldots, f_n(x_1, \ldots, x_{n-1}, 0))$
$+$
$\vdots$
$+\mathrm{im}_2(p; q_{n-1}(x_1, x_2, 0, \ldots, 0), f_n(x_1, x_2, 0, \ldots, 0))$
$+\mathrm{im}_1(p; f_n(x_1, 0, \ldots, 0))$

Observe $\text{moddeg}(f_1, x_1), \ldots, \text{moddeg}(f_1, x_{n-1}) < 0$ and f_1 is a non-zero polynomial which vanishes at the origin, and hence, must be divisible by x_n. By applying Proposition 4.1.5 we get $x_n, f_2, \ldots, f_n$ is a regular sequence.

Since $f_1, \ldots, f_n$ is a regular sequence we may apply (n-6) to get

$$\text{Im}(p;\ f_1, \ldots, f_n) = \text{Im}(p;\ x_n, f_2, \ldots, f_n) + \text{Im}(p;\ q_1, f_2, \ldots, f_n)\,.$$

By definition of intersection multiplicity,

$$\text{Im}(p;\ x_n, f_2, \ldots, f_n) = \text{Im}(p;\ x_n, f_2(x_1, \ldots, x_{n-1}, 0), \ldots, f_n(x_1, \ldots, x_{n-1}, 0))\,.$$

Continuing in this way we get,

$$\begin{aligned}\text{Im}(p;\ f_1, \ldots, f_n) &= \text{Im}(p;\ q_1, f_2, \ldots, f_n) + \text{Im}(p;\ x_n, q_2, \ldots, f_n) + \ldots \\ &+ \text{Im}(p;\ x_n, x_{n-1}, \ldots, q_{n-1}, f_n) + \text{Im}(p;\ x_n, \ldots, x_2, f_n)\,.\end{aligned}$$

By definition of intersection multiplicity,

$$\text{Im}(p;\ x_n, \ldots, x_2, f_n) = max(m \in \mathbb{Z}^+ \mid f_n(x_1, 0, \ldots, 0) \equiv 0 \mod \langle x_1^m \rangle),$$

which completes the proof.

Corollary 4.4.2 *When the conditions of Lemma 4.4.1 hold,*

$$\begin{aligned}\text{Im}(p;\ f_1, \ldots, f_n) &= \text{Im}(p;\ q_1, f_2, \ldots, f_n) \\ &+ \text{Im}(p;\ q_2(x_1, \ldots, x_{n-1}, 0), \ldots, f_n(x_1, \ldots, x_{n-1}, 0)) + \\ &\vdots \\ &+ \text{Im}(p;\ q_{n-1}(x_1, x_2, 0, \ldots, 0), f_n(x_1, x_2, 0, \ldots, 0)) \\ &+ m_n.\end{aligned}$$

Proof Follows from Lemma 4.4.1 and the definition of intersection multiplicity.

Theorem 4.4.3 *Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ be either a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ or be such that at least one of $f_1, \ldots, f_n$ is a unit in $\mathcal{O}_{\mathbb{A}^n,p}$. Algorithm 3 correctly computes the intersection multiplicity of $f_1, \ldots, f_n$ at p or returns Fail.*

Proof By (n-3) we may assume p is the origin. By Corollary 4.1.7 we have $\text{Im}(p;\ f_1, \ldots, f_n) \in \mathbb{N}$.

To prove termination we induct on $\text{Im}(p;\ f_1, \ldots, f_n)$, and show that when Algorithm 3 does not return Fail, we can either compute $\text{Im}(p;\ f_1, \ldots, f_n)$ directly or strictly decrease $\text{Im}(p;\ f_1, \ldots, f_n)$ through splitting.

Suppose $\text{Im}(p;\ f_1, \ldots, f_n) = 0$, then by ($n$-2), one of $f_1, \ldots, f_n$ does not vanish at p, hence Algorithm 3 correctly returns zero. Thus, we may assume $\text{Im}(p;\ f_1, \ldots, f_n) = N$ for some positive $N \in \mathbb{N}$, and moreover, we may now assume $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$.

First, we claim that either Algorithm 3 returns Fail or the input polynomials can be modified while preserving intersection multiplicity such that they satisfy the conditions of Lemma

4.4.1. Moreover, we claim such modifications can be performed in finitely many iterations. To modify the input polynomials such that they satisfy the conditions of Lemma 4.4.1, we proceed iteratively.

Fix some x_j where $1 \leq j \leq n-1$, and suppose $f_1, \ldots, f_{n-j+k}$ all have modular degree in x_{j-k} less than zero for any $1 \leq k < j$ whenever $j > 1$. Notice $f_1, \ldots, f_{n-j+1}$ are the polynomials which must have modular degree less than zero in all variables greater then x_j. By (n-4) and Corollary 4.1.8 we may rearrange $f_1, \ldots, f_{n-j+1}$ so that their modular degrees with respect to x_j are ascending.

To satisfy the conditions of Lemma 4.4.1, in the j-th iteration we must have $n-j$ polynomials in $\{f_1, \ldots, f_{n-j+1}\}$ with modular degree in x_j less than zero. It should be noted at this stage that no polynomial in $\{f_1, \ldots, f_{n-j+1}\}$ has modular degree in x_j equal to zero. When $j = 1$ this follows from $f_1, \ldots, f_n$ being a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$. When $j > 1$, having some $f_i \in \{f_1, \ldots, f_{n-j+1}\}$ with modular degree equal to zero would contradict either being a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ or the hypothesis that $\operatorname{moddeg}(f_i, x_{j-k}) < 0$ for all $1 \leq k < j$. Since the modular degrees are in ascending order we may compute,

$$m = \begin{cases} min(i \mid \operatorname{moddeg}(f_i, x_j) > 0) & \text{if such an } i \text{ exists,} \\ \infty & \text{otherwise.} \end{cases}$$

If $m > n-j$ then $f_1, \ldots, f_{n-j}$ satisfy the conditions of lemma 4.4.1 for the variable x_j and hence we are done.

Suppose $m \leq n-j$. We will use f_m as a pivot to reduce the modular degree with respect to x_j of f_i, for all $i = m+1, \ldots, n-j+1$. Define,

$$L_m = \operatorname{lc}(f_m(x_1, \ldots, x_j, 0, \ldots, 0); x_j),$$
$$L_i = \operatorname{lc}(f_i(x_1, \ldots, x_j, 0, \ldots, 0); x_j),$$

and

$$d = \operatorname{moddeg}(f_i, x_j) - \operatorname{moddeg}(f_m, x_j).$$

If $L_m(p) = 0$ and there is an i such that $L_i \nmid L_m$, then (n-7) cannot be applied not preserve intersection multiplicity since L_m is not a unit in the local ring. When this case occurs, we return Fail.

Suppose either $L_m(p) \neq 0$ or for all i we have $L_m \mid L_i$. In which case, (n-7) allows us to replace f_i with $f_i' = L_m f_i - x^d L_i f_m$ or $f_i' = f_i - x^d \frac{L_i}{L_m} f_m$ respectively. Moreover, Corollary 4.1.8 tells us such a substitution preserves regular sequences.

Notice if $j > 1$, then $\operatorname{moddeg}(f_i', x_{j-k}) < 0$ for all $1 \leq k < j$, since, by assumption, both f_i and f_m have modular degree in x_{j-k} less than zero. Thus, making such a substitution preserves the assumptions of our hypothesis. Lastly, since $\operatorname{moddeg}(f_i', x_j) < \operatorname{moddeg}(f_i, x_j)$, we will have $n-j$ polynomials with modular degree in x_j less than zero or return Fail, in finitely many iterations.

Thus we may now assume $f_1, \ldots, f_n$ satisfy the conditions of Lemma 4.4.1, hence the algorithm correctly splits computations by Lemma 4.4.1 and Corollary 4.4.2.

To show termination, we may suppose none of the computations following the split returns Fail, since in such a case, termination is immediate. Since $\operatorname{Im}(p;\ f_n(x_1, 0, \ldots, 0))$ is a positive integer, as in Lemma 4.4.1, each term has intersection multiplicity strictly less than $\operatorname{Im}(p;\ f_1, \ldots, f_n) = N$ and hence termination follows by induction.

4.5 Triangular Regular Sequences

In this section we consider systems of polynomial equations which form triangular sets in the sense of Section 5.1.2. We observe that under a mild constraint, such a system is also a regular sequence. We will refer to such systems as triangular regular sequences.

We will show for triangular regular sequences, the intersection multiplicity can be computed immediately through means of evaluation. Hence, although Algorithm 3 will always compute the intersection multiplicity for such systems, triangular regular sequences allow us to take advantage of property (n-5) to avoid excessive recursive calls. Thus, when it is known that the input system is a triangular regular sequence, the results in this section give an optimization which can be implemented alongside Algorithm 3.

The ability to compute the intersection multiplicity immediately through means of evaluation also suggests the use of triangular decomposition techniques for computing intersection multiplicities. That is, if a procedure for triangular decomposition was discovered which could preserve the intersection multiplicity (and the property of being a regular sequence) of a system of polynomial equations, then the following observation could lead to a complete algorithm for computing intersection multiplicity. Multiplicity preserving triangular decompositions were explored but not solved in [21], as the proposed algorithm cannot guarantee the output is triangular, and solved for the bivariate case in [6].

Theorem 4.5.1 (McCoy's Theorem) *Let f be a non-zero polynomial in $R[x]$ where R is a commutative ring. Then f is a regular element of $R[x]$ if and only if every non-zero $s \in R$ is such that $sf \neq 0$.*

McCoy's Theorem is a well-known result proven in [24].

Corollary 4.5.2 *Consider a sequence $t_1, \ldots, t_n$ such that for $i = 1, \ldots, n$, each t_i is a non-zero polynomial in $\mathbb{K}[x_i, \ldots, x_n]$ with main variable x_i.*

If at least one non-zero coefficient of t_{i-1} is invertible modulo $\langle t_i, \ldots, t_n \rangle$ for all $1 < i \leq n$, then $t_1, \ldots, t_n$ is a regular sequence in $\mathbb{K}[x_1, \ldots, x_n]$. If $t_1, \ldots, t_n$ also vanish on $p \in \mathbb{A}^n$ then $t_1, \ldots, t_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n, p}$.

Proof Follows from Theorem 4.5.1.

Corollary 4.5.2 tells us regular chains, as described in Section 5.1, are triangular regular sequences, assuming all equations of the regular chain vanish at p.

Proposition 4.5.3 *Consider a sequence $t_1, \ldots, t_n$ such that for $i = 1, \ldots, n$, each t_i is a non-zero polynomial in $\mathbb{K}[x_i, \ldots, x_n]$ with main variable x_i.*

Suppose each $t_1, \ldots, t_n$ vanish at the origin, which we denote by p, and suppose at least one non-zero coefficient of t_{i-1} is invertible modulo $\langle t_i, \ldots, t_n \rangle$ for all $1 < i \leq n$.

Then we may write $t_i(x_i, 0, \ldots, 0)$ as $x_i^{m_i} f_i$ where m_i is the least positive integer such that $f_i \in \mathbb{K}[x_i]$ is a unit in $\mathcal{O}_{\mathbb{A}^n, p}$. Moreover,

$$\mathrm{Im}(p;\, t_1, \ldots, t_n) = m_1 \cdot \ldots \cdot m_n.$$

Proof The result is trivial for $n = 1$, so we may assume $n > 1$. Since $t_i(x_i, 0, \ldots, 0)$ is a non-zero univariate polynomial in $\mathbb{K}[x_i]$ which vanishes at the origin, we may write $t_i(x_i, 0, \ldots, 0) = x_i^{m_i} f_i$ for a positive integer m_i and f_i a unit in the local ring at p.

By Corollary 4.5.2, $t_1, \ldots, t_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$. Hence, we may apply ($n$-6) and Proposition 4.1.5 repeatedly and finally (n-5) to get,

$$\begin{aligned}
\mathrm{Im}(p;\, t_1, \ldots, t_n) &= \mathrm{Im}(p;\, t_1, \ldots, t_{n-1}, x_n^{m_n} f_n) \\
&= \mathrm{Im}(p;\, t_1, \ldots, t_{n-1}, x_n^{m_n}) + \mathrm{Im}(p;\, t_1, \ldots, f_n) \\
&= m_n \mathrm{Im}(p;\, t_1, \ldots, t_{n-1}, x_n) + 0 \\
&= m_n \mathrm{Im}(p;\, t_1(x_1, \ldots, x_{n-1}, 0), \ldots, t_{n-1}(x_{n-1}, 0), x_n) \\
&= m_n \mathrm{Im}\Big(p;\, t_1(x_1, \ldots, x_{n-1}, 0), \ldots, x_{n-1}^{m_{n-1}} f_{n-1}, x_n\Big) \\
&= m_n m_{n-1} \mathrm{Im}(p;\, t_1(x_1, \ldots, 0, 0), \ldots, x_{n-1}, x_n) + 0 \\
&\vdots \\
&= m_1 \cdot \ldots \cdot m_n \mathrm{Im}(p;\, x_1, \ldots, x_n) \\
&= m_1 \cdot \ldots \cdot m_n.
\end{aligned}$$

Chapter 5

The Generalization of Fulton's Algorithm Using Regular Chains

In this chapter we extend the generalization of Fulton's algorithm to handle a regular chain as input rather than a point[1]. This is desirable as the generalization of Fulton's algorithm is practically limited to only those points with rational coordinates. Modifying the generalization of Fulton's algorithm to handle regular chains as input therefore relaxes this practical constraint. We start by reviewing concepts in the theory of regular chains and then define what it means to compute the intersection multiplicity and modular degree at a regular chain. We then provide the modified algorithm and explain how each line corresponds to a section of Algorithm 3, as it's analogue in the theory of regular chains. Lastly, we discuss failure cases for both Algorithm 3 and Algorithm 4; particularly, what causes a failure and how to handle a failure if one is encountered.

5.1 Regular Chain Preliminaries

This section is a short review of concepts from the theory of regular chains and triangular decompositions of polynomial systems. Details can be found in [5].

In order to accurately describe concepts in our review of the theory of regular chains, we need to distinguish between fields and their algebraic closure. Thus, for this section, we relax the assumption that $\mathbb{K}$ is algebraically closed and instead assume $\mathbb{K}$ is a perfect field, that is every irreducible polynomial over $\mathbb{K}$ has distinct roots. Let $\overline{\mathbb{K}}$ be the algebraic closure of $\mathbb{K}$. Let $\mathbb{K}[\underline{X}]$ be the polynomial ring with over $\mathbb{K}$ and with n ordered variables $\underline{X} = X_1 \succ \ldots \succ X_n$. For $F \subseteq \mathbb{K}[\underline{X}]$, we denote by $\langle F \rangle$ and $V(F)$ the ideal generated by F in the polynomial ring $\mathbb{K}[\underline{X}]$ and the algebraic set of $\mathbb{A}^n(\overline{\mathbb{K}})$ consisting of the common roots of the polynomials of F, respectively.

5.1.1 Notations for Polynomials

Let $a, b \in \mathbb{K}[\underline{X}]$ be polynomials, with $b \notin \mathbb{K}$. Denote by mvar(b), init(b) and mdeg(b) respectively, the greatest variable appearing in b (called the *main variable* of b), the leading coefficient

[1] A version of this chapter has been accepted for publication in [13].

of b with respect to mvar(b) (called the *initial* of b) and the degree of b with respect to mvar(b) (called the *main degree* of b). We denote by prem(a, b) and pquo(a, b) the pseudo-remainder and pseudo-quotient in the pseudo-division of a by b; those are, respectively the (uniquely defined) polynomials r and q such that $h^e a = qb + r$ and $r = 0$ or $\deg(r, v) < \text{mdeg}(b)$ where $v = \text{mvar}(b)$, $h = \text{init}(b)$ and $e = \max(0, \deg(a, v) - \text{mdeg}(b) + 1)$.

5.1.2 Triangular Sets

Let $T \subseteq \mathbb{K}[\underline{X}]$ be a *triangular set*, that is, a set of non-constant polynomials with pairwise distinct main variables. Denote by mvar(T) the set of main variables of the polynomials in T. A variable $v \in \underline{X}$ is called *algebraic* with respect to T if $v \in \text{mvar}(T)$, otherwise it is said to be *free* with respect to T. For $v \in \text{mvar}(T)$, we denote by T_v and T_v^- (respectively T_v^+) the polynomial $f \in T$ with $\text{mvar}(f) = v$ and the polynomials $f \in T$ with $\text{mvar}(f) < v$ (respectively $\text{mvar}(f) > v$). Let h_T be the product of the initials of the polynomials of T. We denote by sat(T) the *saturated ideal* of T: if $T = \emptyset$ holds, then sat(T) is defined as the trivial ideal $\langle 0 \rangle$, otherwise it is the ideal $\langle T \rangle : h_T^\infty$. The *quasi-component* $W(T)$ of T is defined as $V(T) \setminus V(h_T)$. The Zariski closure of $W(T)$ in $\mathbb{A}^n(\overline{\mathbb{K}})$, denoted by $\overline{W(T)}$, is the intersection of all algebraic sets $V \subseteq \mathbb{A}^n(\overline{\mathbb{K}})$ such that $W(T) \subseteq V$ holds; moreover we have $\overline{W(T)} = V(\text{sat}(T))$.

5.1.3 Regular Chain

A triangular set $T \subseteq \mathbb{K}[\underline{X}]$ is a *regular chain* if either T is empty, or if v is the largest variable occurring in T, the set T_v^- is a regular chain, and the initial of T_v is regular (that is, neither zero nor zero-divisor) modulo sat(T_v^-). The *dimension* of T, denoted by dim(T), is by definition, the dimension of its saturated ideal and, as a property, equals $n - |T|$, where $|T|$ is the number of elements of T. If T has dimension zero, then T generates sat(T) and we have $V(T) = W(T)$. A regular chain T is *square-free* if for all $t \in T$, the polynomial der(t) is regular with respect to sat(T), where $\text{der}(t) = \frac{\partial t}{\partial v}$ and $v = \text{mvar}(t)$. When the regular chain T is *square-free*, then the ideal sat(T) is radical.

5.1.4 Normalized Regular Chain

The regular chain $T \subseteq \mathbb{K}[\underline{X}]$ is said *normalized* if for every $v \in \text{mvar}(T)$, none of the variables occurring in init(T_v) is algebraic with respect to T_v^-. Denote by d the dimension of T. Let $\underline{Y}$ and $\underline{U} = U_1, \ldots, U_d$ stand respectively for mvar(T) and $\underline{X} \setminus \underline{Y}$. Then, the fact that T is normalized means that for every $t \in T$ we have $\text{init}(t) \in \mathbb{K}[\underline{U}]$. It follows that if T is normalized, then T is a lexicographical Gröbner basis of the ideal that T generates in $(\mathbb{K}[\underline{U}])[\underline{Y}]$ (that is, over the field $(\mathbb{K}[\underline{U}])$ of rational functions), and we denote NF(p, T) the normal form a polynomial $p \in (\mathbb{K}[\underline{U}])[\underline{Y}]$ with respect to T as this Gröbner basis. In particular, if T is normalized and has dimension zero, then for every $t \in T$ we have $\text{init}(t) \in \mathbb{K}$.

5.1.5 Regular GCD

Let i be an integer with $1 \leq i \leq n$, let $T \subseteq \mathbb{K}[\underline{X}]$ be a regular chain, let $p, t \in \mathbb{K}[\underline{X}] \setminus \mathbb{K}$ be polynomials with the same main variable X_i, and $g \in \mathbb{K}$ or $g \in \mathbb{K}[\underline{X}]$ with $\text{mvar}(g) \leq X_i$.

Assume that

1. $X_i > X_j$ holds for all $X_j \in \text{mvar}(T)$, and
2. both $\text{init}(p)$ and $\text{init}(t)$ are regular with respect to $\text{sat}(T)$.

Denote by A the total ring of fractions of the residue class ring $\mathbb{K}[X_{i+1}, \ldots, X_n]/\sqrt{\text{sat}(T)}$. Note that A is isomorphic to a direct product of fields. We say that g is a *regular GCD* of p, t with respect to T whenever the following conditions hold:

(G_1) the leading coefficient of g in X_i is a regular element of A;

(G_2) g belongs to the ideal generated by p and t in $A[X_i]$; and

(G_3) if $\deg(g, X_i) > 0$, then g divides both p and t in $A[X_i]$, that is, both $\text{prem}(p, g)$ and $\text{prem}(t, g)$ belong to $\sqrt{\text{sat}(T)}$.

Assume from now on that T has as many polynomials as variables. Assume also that X_i is the only variable occurring in p or t which is not algebraic in T. Therefore, the three triangular sets T, $T \cup \{p\}$ and $T \cup \{t\}$ can be regarded as zero-dimensional regular chains. Then, with this configuration, Conditions (G_1), (G_2), (G_3) imply the following properties:

1. if $\deg(g, X_i) = 0$ holds then p is regular (actually invertible) modulo $\langle T \cup t \rangle$,
2. if $\deg(g, X_i) > 0$ and $\text{mdeg}(g) = \text{mdeg}(t)$ both hold, then $\sqrt{\langle T \cup t \rangle} = \sqrt{\langle T \cup g \rangle}$ holds and thus we have $V(T \cup t) = V(T \cup g)$,
3. if $\deg(g, X_i) > 0$ and $\text{mdeg}(g) < \text{mdeg}(t)$ both hold, let $q = \text{pquo}(t, g)$, then $T \cup q$ is a regular chain and the following two relations hold:
   (a) $\sqrt{\langle T \cup t \rangle} = \sqrt{\langle T \cup g \rangle} \cap \sqrt{\langle T \cup q \rangle}$,
   (b) $V(T \cup t) = V(T \cup g) \cup V(T \cup q)$.

5.1.6 The Algorithm RegularGCD

Let T, p, t be as in the previous section. In particular, we assume that T, $T \cup \{p\}$ and $T \cup \{t\}$ are zero-dimensional regular chains. Then, the function call $\text{RegularGcd}(p, t, T)$ returns a a set of pairs $(g_1, T_1), \ldots, (g_e, T_e)$ where

1. $T_1, \ldots, T_e \subseteq \mathbb{K}[\underline{X}]$ are regular chains such that $V(T) = V(T_1) \cup \cdots \cup V(T_e)$,
2. $g_1, \ldots, g_e \in \mathbb{K}[\underline{X}]$ are polynomials such that for every $i = 1 \cdots e$, the polynomial g_i is a regular GCD of p, t with respect to T_i, and
3. if T is square-free (respectively normalized) then all regular chains $T_1, \ldots, T_e$ are square-free (respectively normalized).

For convenience, we extend the specifcations of $\text{RegularGcd}(p, t, T)$. With T, t as above, we allow p to be any non-zero polynomial in $\mathbb{K}[\underline{X}]$, with either $p \in \mathbb{K}$ or $\text{mvar}(p) < X_i$, as long as p is regular w.r.t. $\text{sat}(T)$, in which case, $\text{RegularGcd}(p, t, T)$ simply returns the pair (p, T).

5.1.7 The Algorithm Regularize

Let T, p be as in the previous section. The function call Regularize(p, T) computes a set of regular chains $T_1, \ldots, T_e \subseteq \mathbb{K}[\underline{X}]$ such that:

1. for each $i = 1, \ldots, e$, either $p \in \langle T_i \rangle$ holds or p is regular with respect to $\langle T_i \rangle$,
2. we have $V(T) = V(T_1) \cup \cdots \cup V(T_e)$,
3. moreover, if T is square-free (respectively normalized) then all regular chains $T_1, \ldots, T_e$ are square-free (respectively normalized).

For $F \subseteq \mathbb{K}[\underline{X}]$, the function call RegularizeList(F, T) computes a pair of sets of regular chains $T_1, \ldots, T_e \subseteq \mathbb{K}[\underline{X}]$ such that:

1. for each $i = 1, \ldots, e$, for each $p \in F$ either $p \in \langle T_i \rangle$ holds or p is regular with respect to $\langle T_i \rangle$,
2. we have $V(T) = V(T_1) \cup \cdots \cup V(T_e)$,
3. moreover, if T is square-free (respectively normalized) then all regular chains $T_1, \ldots, T_e$ are square-free (respectively normalized).

In practice RegularizeList(F, T) will separate the regular chains for which p is regular from those which generate an ideal containing p, for each $p \in F$. That is, if RegularizeList(F, T) returns a pair U, V, we will let U denote the set of regular chains for which all $p \in F$ are regular and V the set of regular chains for which $p \in \langle T_i \rangle$ for all $T_i \in V$ and $p \in \langle F \rangle$.

5.1.8 Triangular Decomposition

Let $F \subseteq \mathbb{K}[\underline{X}]$. Regular chains $T_1, \ldots, T_e$ of $\mathbb{K}[\underline{X}]$ form a *triangular decomposition* of $V(F)$ in the sense of Kalkbrener (respectively Wu and Lazard) whenever we have $V(F) = \cup_{i=1}^{e} \overline{W(T_i)}$ (respectively $V(F) = \cup_{i=1}^{e} W(T_i)$). Hence, a triangular decomposition of $V(F)$ in the sense of Wu and Lazard is necessarily a triangular decomposition of $V(F)$ in the sense of Kalkbrener, while the converse is not true. Triangular decompositions, both Kalkbrener and Wu-Lazard, can be computed efficiently using the `Triangularize` command in the `RegularChains` Library.

It should be noted that `Triangularize` does not preserve the intersection multiplicity of the system it decomposes, and hence, cannot be used with the observation of Section 4.5 to build a complete intersection multiplicity algorithm. However, `Triangularize` can be used to find the solutions to a given polynomial system, for which the intersection multiplicity of the system at each solution could then be computed, see Section 6.8.

5.2 Extending the Generalization of Fulton's Algorithm

Both Fulton's algorithm and its generalization assume the point p is the origin. When p is rational, both algorithms can easily be adapted to handle this case directly rather than applying property (n-3) and performing an affine change of coordinates. When p is not rational,

encoding p symbolically presents practical challenges that should be addressed in our implementation of the generalization of Fulton's algorithm.

One natural way of encoding these non-rational points, is as a solution to a system of polynomial equations. This can be done using a zero-dimensional regular chain to encode the point of interest. Since intersection multiplicity is defined for a point p, we must explain what it means to compute the intersection multiplicity at a zero-dimensional regular chain. Namely, since a zero-dimensional regular chain can be thought of as encoding a finite group of points in its vanishing set, we are in a sense, defining what it means to compute the intersection multiplicity at a group of points.

Definition 5.2.1 (Intersection Multiplicity at a Regular Chain) *Let* $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_N]$ *and* $T \subset \mathbb{K}[x_1, \ldots, x_N]$ *a zero-dimensional regular chain where* $N \geq n$. *If* $N = n$ *we say* $\mathrm{Im}(T; f_1, \ldots, f_n) = m$, *if* $\mathrm{Im}(p; f_1, \ldots, f_n) = m$ *for every* $p \in \mathbf{V}(T)$, *where* $m \in \mathbb{N} \cup \{\infty\}$. *If* $N > n$ *we say* $\mathrm{Im}(T; f_1, \ldots, f_n) = m$, *if* $\mathrm{Im}(p; T_{x_N}, \ldots, T_{x_{n+1}}, f_1, \ldots, f_n) = m$ *for every* $p \in \mathbf{V}(T)$, *where* $m \in \mathbb{N} \cup \{\infty\}$.

It is worth noting that Definition 5.2.1 only pertains to a regular chain encoding points with the same intersection multiplicity. If a regular chain T encodes points with different intersection multiplicities Definition 5.2.1 would not be applicable to T. This is a natural definition as intersection multiplicity is a local notion, and hence, it is only meaningful to talk about the intersection multiplicity of a group of points if the local behaviour of the given polynomial system is similar at each point. In general, a system of polynomial equations will not have the same intersection multiplicity at each of its solutions, in such a case we can use operations defined on regular chains to decompose the solution set. Since we will only consider zero-dimensional regular chains, after finitely many decompositions, we will eventually reach a case where each regular chain in the decomposition contains only points with the same intersection multiplicity. For this reason, Algorithm 4 returns a collection of regular chains and their corresponding intersection multiplicities in the sense of Definition 5.2.1.

In order to extend the generalization of Fulton's algorithm to handle a zero-dimensional regular chain as input, rather than a point, we must also redefine the notion of modular degree with respect to a regular chain.

Definition 5.2.2 (Modular Degree at a Regular Chain) *Let* f *be a polynomial in* $\mathbb{K}[x_1, \ldots, x_n]$ *and* $T \subseteq \mathbb{K}[x_1, \ldots, x_n]$ *a strongly normalized, square-free, zero-dimensional regular chain with variable ordering* $x_1 \succ \ldots \succ x_n$. *Suppose* $\mathrm{lc}(\mathrm{NF}\left(f, T_{x_i}^{-}\right); x_i)$ *is regular modulo* T *for some* x_i. *Then the modular degree of* f *at* T *with respect to* x_i *is the degree in* x_i *of* $\mathrm{NF}\left(f, T_{x_i}^{-}\right)$.

We shall explain why Algorithm 4 together with Algorithm 5 form a generalization of Algorithm 3 from intersection multiplicity at a point to intersection multiplicity at a (zero-dimensional) regular chain. One short explanation would be invoking the celebrated D5 Principle [11]. But, since Algorithm 3 may already split computations, more details are needed to convince the reader.

We first observe that the specifications of Algorithm 4 generalize that of Algorithm 3, thanks to Definition 5.2.1. For the pseudo-code, we will explain below how each key sequence of lines of Algorithm 3 is adapted to Algorithm 4.

Lines 2-3: In the regular-chain adaptation, Lines 2-5 of Algorithm 4, one must separate the points of $V(T)$ at which all polynomials $f_1, \ldots, f_n$ vanish from those at which one of $f_1, \ldots, f_n$ does not vanish; this task is achieved with RegularizeList(W, T), see Section 5.1.7; the construction of the set W can be seen as an optimization: indeed if all $f_1, \ldots, f_n$ have a null normal form with respect to T then all $f_1, \ldots, f_n$ vanish at every point of $V(T)$ and the call RegularizeList(W, T) is not needed.

Lines 4-5: these two lines in Algorithm 3 determine the trailing degree of f_1, that is, the number of times that x_1 divides f_1; in the regular-chain adaptation, Lines 6-9 of Algorithm 4, the role of x_1 is taken by T_{x_1} and the "divisibility test" is replaced by a Regular GCD computation. Because each call to RegularGCD may split the computations (thus decomposing $V(T)$) we have dedicated an algorithm to that task, namely Algorithm 5. A complete proof of that latter algorithm follows from the properties of RegularGCD given in Section 5.1.5. We note that ensuring that all regular chains involved in the computations are square-free is essential: indeed, at Line 13 of Algorithm 5, we need to make sure that the division of p by g removes once and only once every root common to p and T_{x_1}, which allows us at Line 14 to increment by 1 the current value of m.

Lines 6-8: In the regular-chain adaptation, Lines 10-17 of Algorithm 4, one needs to compute modular degrees in the sense of Definition 5.2.2. Indeed, the leading (or trailing) degree of a polynomial with respect to some variable at a regular chain must be the same at every point solution of that regular chain. This explains the call RegularizeList(C, T) together with the test $|U| + |V| > 1$; indeed, if $|U| + |V| > 1$ holds then there exists a polynomial in the list C which vanishes at some points of $V(T)$ while not vanishing at the others, that is, one modular degree is not well-defined, an issue which is resolved by splitting the computations at Line 17 of Algorithm 4.

Lines 9-23: the regular-chain adaptation, Lines 18-32 of Algorithm 4, is essentially isomorphic to its counterpart in Algorithm 3; indeed, because of the work done in Lines 10-17 of Algorithm 4, no splitting (of the zero set $V(T)$) is needed. However, it is worth noting the switch from regular division to pseudo-division, since all divisions must now occur modulo a regular chain. Consequently, we must now multiply f_i by $\text{init}(L_m)^e$ on line 29 in order to cancel the desired terms, and moreover, we must ensure $\text{init}(L_m)$ is invertible in the local ring at every point in T in order to apply (n-7), where $e = \max(0, \deg(L_i, \text{mvar}(L_m)) - \text{mdeg}(L_m) + 1)$.

Lines 24-33: In the regular-chain adaptation, Lines 33-48 of Algorithm 4, one must perform all required calls to $\text{im}_1, \ldots, \text{im}_n$ at the same regular chains in order to calculate the sums of values returned by those calls. Furthermore, it is worth noting that the exact quotients used in the splitting step of Algorithm 3 have been replaced with pseudo-quotients. Since each regular chain H is strongly normalized and zero-dimensional, the initials of its defining polynomials are in $\mathbb{K}$. Thus, the initial of the pseudo-divisor $\text{NF}\left(H_{x_i}, H_{x_i}^-\right)$ will also be in $\mathbb{K}$ for any i. As such, replacing quotients with pseudo-quotients will not affect the correctness ofAlgorithm 4.

Algorithm 4: Generalized Fulton's Algorithm for Regular Chains

Function $\mathrm{im}_n(T; f_1, \ldots, f_n)$

Input:

1. $T \subset \mathbb{K}[x_1, \ldots, x_N]$ is a zero-dimensional, square-free, strongly normalized regular chain in variables $x_1 \succ \ldots \succ x_N$ where $N \geq n$.

2. $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_N]$ such that for each $p \in \mathbf{V}(T)$ either $f_1, \ldots, f_n$ form a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$, or at least one f_i is a unit in $\mathcal{O}_{\mathbb{A}^n,p}$.

Output: A set of pairs $[m_i, T_i]$ such that: (i) $\mathbf{V}(T) = \bigcup \mathbf{V}(T_i)$, and (ii) m_i is either $\mathrm{Im}(T_i;\ f_1, \ldots, f_n)$ or Fail

$W \leftarrow \{f_i \,|\, \mathrm{NF}(f_i,\ T) \neq 0\}$

if $W \neq \emptyset$ **then**

  $U, V \leftarrow \mathsf{RegularizeList}(W, T)$

  return $\{[0, H] \,|\, H \in U\} \cup \bigcup_{H \in V} \mathrm{im}_n(H; f_1, \ldots, f_n)$ `/* Red */`

if $n = 1$ **then** `/* Compute multiplicity */`

  $U \leftarrow \mathsf{Regularize}(f_1, T^-_{x_1})$

  $U' \leftarrow \bigcup_{H \in U} \{T_{x_1}\} \cup H$

  return $\bigcup_{H \in U'} \mathrm{valuation}(f_1, H)$

for $i = 1, \ldots, n$ **do**

  for $j = 1, \ldots, n-1$ **do** `/* Compute modular degrees */`

    $F[i][j] \leftarrow \mathrm{NF}\left(f_i,\ T^-_{x_j}\right)$

    $C[(i-1)(n-1)+j] \leftarrow \mathrm{lc}(F[i][j], x_j)$

    $R[i][j] \leftarrow \deg_{x_j}(F[i][j])$

$U, V \leftarrow \mathsf{RegularizeList}(C, T)$

if $|U| + |V| > 1$ **then**

  return $\bigcup_{H \in U \cup V} \mathrm{im}_n(H; f_1, \ldots, f_n)$

for $j = 1, \ldots, n-1$ **do** `/* Orange */`

  Reorder $f_1, \ldots, f_{n-j+1}$ so that $R[1][j] \leq \ldots \leq R[n-j+1][j]$ `/* Green */`

  $m \leftarrow \min(i \mid R[i][j] > 0)$ or $m \leftarrow \infty$ if no such i exists

  if $m \leq (n-j)$ **then**

    for $i = m+1, \ldots, n-j+1$ **do** `/* Blue */`

      $d \leftarrow R[i][j] - R[m][j]$

      $L_m \leftarrow C[(m-1)(n-1)+j]$

      $L_i \leftarrow C[(i-1)(n-1)+j]$

      if $\mathrm{NF}(L_m,\ T) \neq 0$ **then**

        $f'_i \leftarrow L_m f_i - x_j^d L_i f_m$

      else if $\mathrm{NF}(\mathrm{prem}(L_i, L_m),\ T^-_n) = 0$ **and** $\mathrm{NF}(\mathrm{init}(L_m),\ T) \neq 0$ **then**

        `/* Where` $e = \max(0, \deg(L_i, \mathrm{mvar}(L_m)) - \mathrm{mdeg}(L_m) + 1)$ `*/`

        $f'_i \leftarrow \mathrm{init}(L_m)^e f_i - x_j^d \mathrm{pquo}(L_i, L_m) f_m$

      else

        return $\{[Fail, T]\}$

    return $\mathrm{im}_n(T; f_1, \ldots, f_m, f'_{m+1}, \ldots, f'_{n-j+1}, \ldots, f_n)$

33
34 /* Yellow */
35 tasks ← $\mathrm{im}_1(T; F[n][1])$
36 **for** $i = 2, \ldots, n-1$ **do**
37 newTasks ← ∅
38 **for** task *in* tasks **do** /* each task is of the form $[m, H]$, where H is a regular chain */
39 $m, H \leftarrow$ task
40 $q \leftarrow \mathrm{pquo}(\mathrm{NF}\left(f_{n-i+1}, H^-_{x_i}\right), \mathrm{NF}\left(H_{x_i}, H^-_{x_i}\right))$
41 newTasks ← newTasks∪$\{[m+m', H'] \mid [m', H'] \in \mathrm{im}_{\mathrm{i}}(H; q, F[n-i+2][i], \ldots, F[n][i])\}$
42 tasks ← newTasks
43 results ← ∅
44 **for** task *in* tasks **do**
45 $m, H \leftarrow$ task
46 $q \leftarrow \mathrm{pquo}(\mathrm{NF}\left(f_1, H^-_{x_n}\right), \mathrm{NF}\left(H_{x_n}, H^-_{x_n}\right))$
47 results ← results ∪ $\{[m+m', H'] \mid [m', H'] \in \mathrm{im}_{\mathrm{n}}(H; q, f_2, \ldots, f_n)\}$
48 **return** results

5.3 Failure Cases

Since the algorithms developed are only partial algorithms, this section will discuss cases where the algorithms presented do not succeed in computing the intersection multiplicity. We will focus mainly on Algorithm 4 but will occasionally make comparisons to its analogue, Algorithm 3, which computes the intersection multiplicity at a point.

Fix some $n > 2$ and j such that $1 < j < n$. Suppose we are in the j-th iteration of the loop on line 18 of Algorithm 4 and we are computing the intersection multiplicity of polynomials $f_1, \ldots, f_n$ at a regular chain T. Let L_m be as in Algorithm 4, that is let $L_m = \mathrm{lc}(\mathrm{NF}\left(f_m, T^-_{x_j}\right), x_j)$ for some m. The first condition we check is whether $\mathrm{NF}(L_m, T) \neq 0$. If this condition holds then L_m is a unit in the local ring at any point in T, hence we may apply property (n-7) to rewrite the system without increasing the intersection multiplicity. When this condition does not hold, we check whether $\mathrm{NF}(\mathrm{init}(L_m), T) = 0$ and whether there is no pseudo-remainder modulo the regular chain $T^-_{x_n}$, after a pseudo-division between L_m and L_i, for all $i = m+1, \ldots, n-j+1$, where $L_i = \mathrm{lc}(\mathrm{NF}\left(f_i, T^-_{x_j}\right), x_j)$ and $e = \max(0, \deg(L_i, \mathrm{mvar}(L_m)) - \mathrm{mdeg}(L_m) + 1)$. In this case we can make a similar substitution and apply (n-7) to preserve the intersection multiplicity. When neither of these conditions hold, we return Fail, as we cannot further simplify our input system to compute the intersection multiplicity. In fact, when $\mathrm{NF}(L_m, T) = 0$, substituting as in line 27, will increase the intersection multiplicity at any point $p \in \mathbf{V}(T)$ for which L_m is not a unit in $\mathcal{O}_{\mathbb{A}^n, p}$. Moreover, such a p will exist whenever $\mathrm{NF}(L_m, T) = 0$.

The following example illustrates an input system which successfully computes the intersection multiplicity at one regular chain in the decomposition but fails to compute the intersec-

Algorithm 5: Valuation

Function valuation(f, T)

Input:

1. T a zero-dimensional, square-free, strongly normalized regular chain in variables $x_1 \succ \ldots \succ x_N$ where $N \geq n$.
2. $f \in \mathbb{K}[x_1, \ldots, x_N]$ with main variable x_1. Moreover, $\mathrm{NF}\left(f,\ T_{x_1}^{-}\right) \neq 0$.

Output: A set of pairs $[m_i, T_i]$ such that: (i) $\mathbf{V}(T) = \bigcup \mathbf{V}(T_i)$, and (ii) $m_i = \mathrm{Im}\left(T_i;\ \mathrm{NF}\left(f,\ T_{i,x_1}^{-}\right)\right)$.

```
tasks ← {[f, T, 0]}
results ← ∅
while tasks ≠ ∅ do
    p, T, m ← removeElem(tasks)
    L ← RegularGcd(p, T_{x1}, T_{x1}^-)
    for g, C ∈ L do
        d ← deg_{x1}(g)
        H ← {T_{x1}} ∪ C
        if d = 0 then
            results ← results ∪ {[m, H]}
        else
            q ← NF(pquo(p, g), C)
            tasks ← tasks ∪ {[q, H, m + 1]}
return results
```

tion multiplicity for another.

Example Consider the regular chain in variables $x \succ y \succ z$ given by $T = x(x+1), y, z$. The regular chain T encodes the two points $(0,0,0)$ and $(-1,0,0)$. Calling $\mathrm{im}_n(T; f_1, f_2, f_3)$ where $f_1 = xy + z, f_2 = y^2 + z^2, f_3 = x(x+1) + y + z$ splits T into two regular chains $t_1 = x+1, y, z$ and $t_2 = x, y, z$. Since both t_1 and t_2 encode only a point, the procedure of computing the intersection multiplicity at all points in t_1 or t_2 is almost identically to Algorithm 3.

Consider what happens when we compute the intersection multiplicity at t_2, or analogously, call Algorithm 3 with arguments $p = (0,0,0)$ and f_1, f_2, f_3. We can begin at $j = 2$ since f_1 and f_2 have modular degree in x less than zero. Since f_1 has smaller modular degree in y than f_2, the pivot index m is set to 1. We compute $L_m = x$ and $L_i = 1$. Since $\mathrm{NF}(L_m, t_2) = 0$ the conditional statement on line 26 of Algorithm 4 is not satisfied. Analogously, L_m does not vanish at $p = (0,0,0)$ so the conditional statement of line 17 in Algorithm 3 is not satisfied. Next see $\mathrm{NF}(\mathrm{prem}(L_i, L_m), t_2) \neq 0$ or analogously 1 is not divisible by x in the polynomial ring. We have exhausted all ways of rewriting the input system provided in the algorithm and hence we must now add the pair $[FAIL, t_2]$ to the output set.

This issue is avoided when we compute the intersection multiplicity at t_1 since the leading polynomial of t_1 is simply $x + 1$. Thus, when $j = 2$, $\mathrm{NF}(L_m, t_1) = \mathrm{NF}(x, \{x+1, y, z\}) = x$ and hence we may apply (n-7) to preserve intersection multiplicity upon substitution. Continuing the algorithm leads to an output of $[2, t_1]$, and hence the final output returned is $\{[2, t_1], [FAIL, t_2]\}$.

As we will discuss in Section 6.5, the success of failure on the generalization of Fulton's algorithm can often come down to the choice of variable ordering. In the above example we used the ordering $x \succ y \succ z$ and could compute the intersection multiplicity at one of the two regular chains returned. If we instead consider the same system under the ordering $y \succ z \succ x$, we can compute the intersection multiplicity at both regular chains. This presents an interesting question as to what constitutes an optimal variable ordering for a given system of polynomials. We hope to address this question in a future paper.

Chapter 6

Implementing the Generalization of Fulton's Algorithm

In this chapter we provide details on the MAPLE implementation of the generalization of Fulton's algorithm, for both a point and a regular chain as input[1]. Moreover, we describe the integration of this implementation with the implementation of the algorithm of Vrbik's PhD thesis, in the form of a hybrid procedure.

6.1 Overloading the `IntersectionMultiplicity` Command

In our implementation, we overload the `IntersectionMultiplicity` command to handle several different calling sequences. The first calling sequence occurs when one wishes to compute the intersection multiplicity at a point p but knows the system of polynomial equations forms a regular chain. In which case, the observation made in Section 4.5 allows us to compute the intersection multiplicity immediately by means of evaluation. The second calling sequence applies the generalization of Fulton's algorithm at a point p. The third calling sequence seeks to apply the algorithm of Vrbik et al. along side the adaptation of the generalization of Fulton's algorithm to regular chains. That is, the third calling sequence takes a regular chain and a system of polynomial equations and applies first, the generalization of Fulton's algorithm and then, upon detecting a failure, applies the algorithm of Vrbik et al. Lastly, the final calling sequence is used to return meaningful error messages when none of the previous calling sequences are satisfied.

6.2 `IntersectionMultiplicity` at a Point

The second calling sequence computes the intersection multiplicity at a point p. This calling sequence takes advantage of the cache option in its helper functions to compute the image of polynomials and modular degrees efficiently. Moreover, this calling sequence provides some support for coordinates and coefficients which are not rational given by `RootOf`. Although algebraic coordinates can be handled by encoding them in a regular chain, allowing algebraic

[1] A version of this chapter has been accepted for publication in [13].

coordinates specified by RootOf can simplify the calling sequence in some cases. When the system of polynomial equations, or the point of interest contain non-rational coordinates, the normalizer environment variable is set to evala and the algorithm proceeds as expected. Reducible RootOf errors are caught, in which case it is recommended the user uses the calling sequence which handles regular chains.

6.3 IntersectionMultiplicity at a Regular Chain

The third calling sequence invokes a hybrid algorithm which calls first the generalization of Fulton's algorithm, and then if necessary, the algorithm of Vrbik et al. We apply the generalization of Fulton's algorithm first in this hybrid algorithm as it is often faster and can compute more examples than the current implementation of the algorithm of Vrbik et al. as suggested by the results in the next chapter. It is possible however, that the algorithm of Vrbik et al. can succeed in some cases where the generalization of Fulton's algorithm fails, as we will see in the next chapter; hence, a hybrid algorithm which combines the two approaches is desirable. Both the generalization of Fulton's algorithm and the algorithm of Vrbik et al. can be accessed individually in the IntersectionMultiplicity command by setting the optional method keyword equal to fulton or tangentcone respectively.

6.4 Non-Regular Sequences

The input constraints for the generalization of Fulton's algorithm require that the system of polynomials, $f_1, \ldots, f_n$, is a regular sequence at the point p, or in the case of a zero-dimensional regular chain T, that $f_1, \ldots, f_n$ is a regular sequence at all points in $\mathbf{V}(T)$ (of course, this assumption is only required when no f_i is a unit in any of the respective local rings). Testing for this constraint is not practical for a standard basis-free algorithm as it requires the use of standard bases, see the description of is_reg and is_regs in [16, Section 7.6]. Moreover, this constraint is essential to the proof of termination, hence we provide several heuristics for testing for non-regular sequences in our implementation. The key observation behind the heuristics is that by applying Proposition 4.1.5 and Corollary 4.1.8, it suffices to test for a non-regular sequence in any branch of computation. As the size of the branch decreases, which occurs during the splitting stage, testing for non-regular sequences becomes easier. For example, branches of size $n = 1$ will be a non-regular sequence only when $f_n = 0$. When $n = 2$ it suffices to check $\gcd(f_1, f_2)(p) \neq 0$, as we did in Fulton's algorithm. Applying similar heuristics during the start of each recursive call allows our implementation of the generalization of Fulton's algorithm to catch many non-regular sequences and return an error indicating the input was invalid.

6.5 Changes of Coordinates

The generalization of Fulton's algorithm requires a variable ordering to be specified before runtime. Although this ordering is needed for the algorithm, it is independent of the geometry of the input system. Hence, the choice of a good or bad ordering can cause the algorithm to

succeed or fail. Since it is impractical to try all possible variable orderings, we leave the choice of variable ordering up to the user. In our implementation, the variable ordering can be changed manually by the user by modifying one or more of the required parameters. Additionally, setting the `maxshift` option equal to $k \in \mathbb{N}$ will apply a circular left shift to the variable ordering upon detecting a failure, for up to k failures.

6.6 Pivot Selection

The implementation of the generalization of Fulton's algorithm, at both a point and regular chain, also improves upon the pivot selection process of Algorithm 3 and Algorithm 4. In line 11 of Algorithm 3 and line 20 of Algorithm 4, the index m is defined as the index of the polynomial with minimal modular degree with respect to some variable. Such an m can be thought of as the index to a pivot element, as f_m may be used to reduce the modular degrees of some polynomials within the current iteration of the algorithm. It is possible however, for multiple polynomials to share the same minimal modular degree with respect to some variable. Since the success of this procedure is often dependent on the invertibility of a particular leading coefficient in the local ring, namely that given on line 15 of Algorithm 3 and line 24 of Algorithm 4, having multiple viable choices for a pivot element increases the algorithm's chance of succeeding.

Consider the system $xy - z, (x + 1)y, x \in \mathbb{K}[x, y, z]$ at the origin. Both $xy - z$ and $(x + 1)y$ have modular degree 1 with respect to y. The leading coefficients of $xy - z$ and $(x + 1)y$ modulo $\langle z \rangle$ with respect to y are x and $x + 1$ respectively. The generalization of Fulton's algorithm, as presented in Algorithm 3, would only consider the first polynomial as a pivot, and hence, would return `Fail` as x is not invertible in the local ring at the origin. But clearly this need not be the case as $x + 1$ is invertible in the local ring and $(x + 1)y$ has minimal modular degree with respect to y. Hence, extending the pivot selection process as to consider all polynomials with minimal modular degree can further strengthen the generalization of Fulton's algorithm and is therefore, included in our implementation.

6.7 Expression Swell

The rewriting process used in the generalization of Fulton's algorithm decreases the modular degrees with respect to a given variable in each successful iteration. In doing so, it is possible that it increases the degrees of the polynomials being rewritten with respect to some other variables. Similarly, the rewriting process may also cause the size of a polynomial's coefficients to grow. In some cases, this process causes severe expression swell which can act as a bottleneck for the performance of this procedure.

In our implementation we try to mitigate the effects of expression swell by considering only primitive polynomials. Namely, in each iteration we divide each polynomial by the greatest common divisor of its coefficients, therefore replacing each polynomial with its primitive part. This does not affect the intersection multiplicity since the greatest common divisor of coefficients in $\mathbb{K}$ is also in $\mathbb{K}$, hence we are dividing by an invertible element.

6.8 TriangularizeWithMultiplicity

In order to compute a non-zero intersection multiplicity, one first needs a solution to a given polynomial system. Throughout this manuscript we have assumed the user already has a solution or regular chain encoding solutions, to the given polynomial system. In practice one may not always know the solutions of a polynomial system. When this is the case, it would be desirable to have an algorithm which can first solve a system of polynomial equations and second compute the intersection multiplicity at each solution.

The AlgebraicGeometryTools sub-package of the RegularChains library's TriangularizeWithMultiplicity command addresses this problem by combining the Triangularize algorithm, recall Section 5.1.8, with MAPLE's IntersectionMultiplicity command. The TriangularizeWithMultiplicity solves the system of polynomial equations using Triangularize and then maps the IntersectionMultiplicity command to each regular chain in the output. Thus, we extend TriangularizeWithMultiplicity to support our modifications to the IntersectionMultiplicity command. Namely, TriangularizeWithMultiplicity now supports the calling sequence of IntersectionMultiplicity which takes a regular chain as input and supports all optional arguments this calling sequence of IntersectionMultiplicity supports.

Chapter 7

Experiments

In this chapter we provide experimental results for the generalization of Fulton's algorithm[1]. We first use the `IntersectionMultiplicity` command and `TriangularizeWithMultiplicity` command to compare the generalization of Fulton's algorithm to the algorithm of Vrbik et al. (recall both commands can access either algorithm individually using the `method` option). Next, we compare the generalization of Fulton's algorithm to SINGULAR's `iMult` command, although, the breadth of systems we can compare are greatly limited as `iMult` is restricted to only those systems which have a solution at the origin. Lastly, we summarize the results and discuss several ways one may be able to improve on our implementation by taking advantage of different variable orderings.

7.1 Benchmarking Against The Algorithm of Vrbik et al.

In this section, we benchmark the implementation of the generalization of Fulton's algorithm relative to the implementation of the algorithm of Vrbik et al. in MAPLE. All tests were run on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz machine with two processors using MAPLE 2021.2. For these experiments, `kernelopts(numcpus)` was set to one in order to run computations in serial. Additionally, we set `kernelopts(cpulimit)` to 2000, terminating experiments that take over 2000 seconds of CPU time. We will use NR, denoting "No Result", when experiments exceed the given time limit. Timings were performed using the `Usage` command of MAPLE's `CodeTools` library. For experiments that produced errors, the `IntersectionMultiplicity` command was wrapped in the `traperror` command to time how long the algorithm took to throw an error.

Since the algorithm of Vrbik et al. can only handle points encoded in a regular chain as input, we will use the calling sequence for the generalization of Fulton's algorithm which handles regular chains as input. That is, we will run experiments using our implementation of Algorithm 4. Doing so ensures that both algorithms will experience any overhead inherent in using regular chains, which we believe will lead to more accurate comparisons.

All systems of polynomials are described in Appendix A. The format of the below tables is as follows:

[1] A version of this chapter has been accepted for publication in [13].

1. The column n denotes the size of the square system, that is the number of variables and number of polynomials in the system.

2. The point column represents the point encoded by the zero-dimensional regular chain passed as input[2].

3. The ordering column denotes the variable ordering given to the regular chain. This variable ordering is the same ordering used by the generalization of Fulton's algorithm in our implementation.

4. Columns under the heading Fulton denote results obtained from calling the generalization of Fulton's algorithm and columns under the heading Vrbik denote results obtained from calling the algorithm of Vrbik et al.

5. The Im column denotes the intersection multiplicity computed using the given algorithm and CPU time denotes the CPU time elapsed during the respective computation.

Table 7.1 compares the two algorithms on systems chosen by the author, at a regular chain encoding the origin. Tests 1-8 consider a simple family of systems that demonstrate how each algorithm scales as the number of variables increases. In particular, we see the generalization of Fulton's algorithm often runs 1-2 orders of magnitude faster than the algorithm of Vrbik et al. yielding a speed up of almost 17 minutes when $n = 25$. In tests 9-11, we see the algorithm of Vrbik et al. either returns an error or does not return before timeout occurs. Conversely, the generalization of Fulton's algorithm can compute all of these examples. Also, worth noting, the system in test 11 contains polynomials which are all singular at the origin. Since the algorithm of Vrbik et al. requires at least one polynomial to be singular in order to apply the reduction criterion, it will always fail in such cases. Hence, the generalization of Fulton's algorithm becomes particularly attractive when all polynomials are singular at the point of interest, as it relies on a different set of conditions to succeed, unrelated to the geometry of the input. Test 12 illustrates that this difference becomes even more pronounced when the polynomials themselves become more complex. Here, we observe that the generalization of Fulton's algorithm runs in roughly the same time as test 3, the test with 7 variables, whereas the algorithm of Vrbik et al. takes approximately 50 seconds longer than it did in test 3. This suggests that both the size of the polynomial system and complexity of the polynomials themself, will cause a significant discrepancy in the performance of the two algorithms, making the generalization of Fulton's algorithm an attractive option for large or complex systems.

In Table 7.2 we consider the systems described in [9, 3, 26] at regular chains encoding just a point. In the cases where both algorithms succeed, we observe a speedup of 1-4 seconds. Moreover, in both mth191 and DZ2 we see the generalization of Fulton's algorithm is able to compute the intersection multiplicity whereas the algorithm of Vrbik et al. cannot. In Ojika4, we see an example where the algorithm of Vrbik et al. succeeds and the generalization of Fulton's algorithm does not, which justifies the implementation of a hybrid algorithm, combining the two approaches. Although, it is worth noting that upon selecting a better variable ordering,

[2]All regular chains, with the exception of the regular chain used in Carpasse, encode just a point. For the Carpasse system, the regular chain must encode an additional point since the desired point contains algebraic coordinates.

Table 7.1: `IntersectionMultiplicity` Using the Author's Tests

	System Specifications			Fulton		Vrbik	
System	n	Ordering	Point	Im	CPU Time	Im	CPU Time
test 1	3	$x_1 \succ \ldots \succ x_3$	origin	2	15.00ms	2	329.00ms
test 2	5	$x_1 \succ \ldots \succ x_5$	origin	2	31.00ms	2	2.95s
test 3	7	$x_1 \succ \ldots \succ x_7$	origin	2	94.00ms	2	8.84s
test 4	9	$x_1 \succ \ldots \succ x_9$	origin	2	188.00ms	2	20.69s
test 5	11	$x_1 \succ \ldots \succ x_{11}$	origin	2	359.00ms	2	40.39s
test 6	13	$x_1 \succ \ldots \succ x_{13}$	origin	2	610.00ms	2	71.58s
test 7	15	$x_1 \succ \ldots \succ x_{15}$	origin	2	1.17s	2	118.52s
test 8	25	$x_1 \succ \ldots \succ x_{25}$	origin	2	7.81s	2	17.32m
test 9	3	$x \succ y \succ z$	origin	5	47.00ms	NR	NR
test 10	3	$x \succ y \succ z$	origin	24	109.00ms	ERROR	16.00ms
test 11	3	$x \succ y \succ z$	origin	45	93.00ms	ERROR	15.00ms
test 12	6	$x_1 \succ \ldots \succ x_6$	origin	2	125.00ms	2	18.09s

the generalization of Fulton's algorithm can compute the intersection multiplicity of all points in Ojika4. We alo include two bivariate systems, namely decker1 and decker2, to show our implementation in the bivariate case is also slightly faster.

As mentioned earlier our implementation of the generalization of Fulton's algorithm can compute the intersection multiplicity of a group of points encoded by a zero-dimensional regular chain. So far, we have only benchmarked examples where the regular chain encodes a single point (with the exception of the Caprasse system). This is because the implementation of the algorithm of Vrbik et al. does not provide sufficient support for regular chains encoding a group of points and may throw an error or return an incorrect intersection multiplicity when this is the case. Instead, it is suggested to use the `TriangularizeWithMultiplicity` command to compute the intersection multiplicity of a group of points.

Since `TriangularizeWithMultiplicity` may return several regular chains and their corresponding intersection multiplicities, Table 7.3 replaces the column Im with a new column, Success Ratio. The Success Ratio column denotes the number of intersection multiplicities successfully computed over the number of regular chains returned, using the generalization of Fulton's algorithm and the algorithm of Vrbik et al. respectively. That is, the Success Ratio column tells us how many solutions we were able to compute the intersection multiplicity at. We also note that the implementation of the algorithm of Vrbik et al. returns an error any time it cannot compute all intersection multiplicities, hence the Success Ratio column computed using the algorithm of Vrbik et al. will contain only full fractions, errors, and NR.

The polynomial systems and points used in Table 7.2 all had intersection multiplicity greater than one. Moreover, all systems used in Table 7.3 all had at least one solution with intersection multiplicity greater than one. The reason we selected systems which have solutions with intersection multiplicity greater than 1 is due to an optimization implemented with the algorithm of Vrbik et al. This optimization uses Jacobians to quickly compute the intersec-

Table 7.2: `IntersectionMultiplicity` Using Examples from the Literature

	System Specifications			Fulton		Vrbik	
System	n	Ordering	Point	Im	CPU Time	Im	CPU Time
cbms1	3	$x \succ y \succ z$	$(0,0,0)$	FAIL	32.00ms	ERROR	16.00ms
cbms2	3	$x \succ y \succ z$	$(0,0,0)$	FAIL	31.00ms	ERROR	31.00ms
mth191	3	$x \succ y \succ z$	$(0,1,0)$	4	31.00ms	ERROR	1.47s
decker1	2	$x \succ y$	$(0,0)$	3	15.00ms	3	171.00ms
decker2	2	$x \succ y$	$(0,0)$	4	14.00ms	4	187.00ms
Ojika2	3	$x \succ y \succ z$	$(0,0,1)$	2	63.00ms	2	1.53s
Ojika2	3	$x \succ y \succ z$	$(1,0,0)$	2	62.00ms	2	1.48s
Ojika3	3	$x \succ y \succ z$	$(0,0,1)$	4	31.00ms	4	1.62s
Ojika3	3	$x \succ y \succ z$	$(-\frac{5}{2},\frac{5}{2},1)$	2	31.00ms	2	1.02s
Ojika4	3	$x \succ y \succ z$	$(0,0,1)$	FAIL	16.00ms	ERROR	657.00ms
Ojika4	3	$x \succ y \succ z$	$(0,0,10)$	FAIL	16.00ms	3	2.00s
Ojika4	3	$x \succ z \succ y$	$(0,1,0)$	3	46.00ms	ERROR	2.50s
Ojika4	3	$x \succ z \succ y$	$(0,10,0)$	3	63.00ms	3	4.02s
Caprasse	4	$x_1 \succ \ldots \succ x_4$	$(2,-i\sqrt{3},2,i\sqrt{3})$	FAIL	94.00ms	NR	>2000s
KSS	5	$x_1 \succ \ldots \succ x_5$	$(1,1,1,1,1)$	FAIL	43.00ms	ERROR	56.94s
DZ1	4	$x_1 \succ \ldots \succ x_4$	$(0,0,0,0)$	FAIL	31.00ms	ERROR	16.00ms
DZ2	3	$x \succ z \succ y$	$(0,0,-1)$	16	78.00ms	ERROR	16.00ms
Solotarev	4	$x_1 \succ \ldots \succ x_4$	$(\frac{5}{3},-1,5,-\frac{47}{27})$	2	110.00ms	2	1.36s
Solotarev	4	$x_1 \succ \ldots \succ x_4$	$(-1,-1,5,3)$	2	93.00ms	2	1.36s

Table 7.3: `TriangularizeWithMultiplicity` Using Examples from the Literature

	System Specifications		Fulton		Vrbik	
System	n	Ordering	Success Ratio	CPU Time	Success Ratio	CPU Time
cbms1	3	$x \succ y \succ z$	10/11	641.00ms	ERROR	218.00ms
cbms2	3	$x \succ y \succ z$	1/2	11.55s	ERROR	422.00ms
mth191	3	$x \succ y \succ z$	8/8	609.00ms	ERROR	2.45s
decker1	2	$x \succ y$	3/3	47.00ms	3/3	140.00ms
decker2	2	$x \succ y$	3/3	63.00ms	3/3	250.00ms
Ojika2	3	$x \succ y \succ z$	4/4	219.00ms	4/4	5.38s
Ojika3	3	$x \succ y \succ z$	2/2	62.00ms	2/2	5.02s
Ojika4	3	$x \succ y \succ z$	3/5	438.00ms	ERROR	906.00ms
Ojika4	3	$x \succ z \succ y$	5/5	500.00ms	ERROR	3.41s
Caprasse	4	$x_1 \succ x_2 \succ \ldots$	4/15	1.58s	NR	>2000s
Caprasse	4	$x_4 \succ x_2 \succ x_1 \succ x_3$	12/15	4.48s	ERROR	14.09s
KSS	5	$x_1 \succ x_2 \succ \ldots$	16/17	3.34s	ERROR	71.16s
DZ1	4	$x_1 \succ x_2 \succ \ldots$	NR	>2000s	ERROR	313.00ms
DZ2	3	$x \succ z \succ y$	2/2	172.00ms	ERROR	47.00ms
Solotarev	4	$x_1 \succ x_2 \succ \ldots$	4/4	453.00ms	4/4	3.47s

Table 7.4: `TriangularizeWithMultiplicity` Using Examples from the Literature with Multiplicity 1

	System Specifications	Fulton	
System	n	Success Ratio	CPU Time
eco5	5	4/4	609.00ms
eco6	6	5/5	2.52s
eco7	7	6/6	97.17s
trinks	6	2/2	516.00ms
Czapor Geddes 1	3	1/1	281.00ms
A Bifurcation Problem	3	3/3	2.95m
quadfor2	4	1/1	109.00ms
Lorentz	4	6/6	485.00ms
S9_1	8	2/2	1.80s
cyclic3	3	2/2	110.00ms

tion multiplicity when systems have an intersection multiplicity of one, see [27].

In Table 7.4, we show several experiments for the `TriangularizeWithMultiplicity` command using other examples from the literature[3]. All solutions to these systems have an intersection multiplicity of one, and hence, we only provide results for `TriangularizeWithMultiplicity` using the `method=fulton`. The ordering of the variables for systems in this table is included in Appendix A. Systems such as eco7 and A Bifurcation Problem illustrate the utility of this optimization, and hence, we consider this optimization desirable and wish to integrate it into our implementation of the generalization of Fulton's algorithm in the future.

7.2 Benchmarking Against `iMult`

In this section we provide experimental results comparing the generalization of Fulton's algorithm to SINGULAR's `iMult` algorithm. All tests were run on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz machine with two processors. Since `iMult` works only at points with rational coordinates, we will use the calling sequence for our implementation of the generalization of Fulton's algorithm that takes a point as input, rather than a regular chain. All other testing parameters (for tests run using MAPLE) remain unchanged from Section 7.1.

For tests run using SINGULAR, we used the following settings for all experiments. Tests were run on SINGULAR version 4.2.1 and timings were calculated using SINGULAR's `timer` command. Before running any tests, the commands `system("--ticks-per-sec",1000);` and `system("--min-time","0.001");` were called, to set the unit of measurement to milliseconds and minimal return time as 1 millisecond. Additionally, the commands

[3]The implementation of the `RegularChains` library restricts polynomials to those of integer coefficients. Hence, to compute the intersection multiplicity we multiply the systems A Bifurcation Problem and quadfor2 by a factor of 64 and 3 respectively to eliminate any fractions. This does not change the intersection multiplicity since 64 and 3 are units.

`system("--cpus","1");` and `setcores(1);` were called to ensure experiments run in serial. Lastly, as was the convention in the last section, we stop experiments after 2000 seconds, unfortunately, due to difficulties with Singular's `abort` system option, we were unable to stop computations at 2000 seconds of CPU time. To work past this, we approximated this timeout by measuring 2000 seconds of real time instead. All rings used in these experiments had characteristic zero and used the negative degree reverse lexicographical local ordering.

This section focuses on two main families of polynomial systems, the nql-*n*-*d* family and the simple-nql-*n*-*d* family. Although we keep the name the same, we have modified these systems from their traditional presentation, which we describe in Appendix A. The reason for this modification, and the reason we focus on only two families of systems, rather than a diverse test pool, is that `iMult` is limited to only systems which have a solution at the origin (or systems which can be made to have a solution at the origin after an affine change of coordinates). This resulted in difficulty finding large, zero-dimensional systems, which had a solution at the origin, and which did not have a trivial standard basis (in order to provide meaningful comparisons). Indeed, we only focus on large examples since the purpose of a standard basis-free intersection multiplicity algorithm is to provide an alternative for when methods which use standard bases fail to terminate in a reasonable amount of time. Hence, although most of the systems used in Section 7.1 can be made to have a solution at the origin, and hence can be used in comparisons between `iMult` and the generalization of Fulton's algorithm, these comparisons are less meaningful as both techniques succeed relatively quickly.

The constraint imposed by `iMult`, limiting intersection multiplicity computations to only systems which have a solution at the origin, further justifies our goal of designing an intersection multiplicity algorithm that works at any point, not just the origin. As a result of this limitation, we are unable to provide meaningful comparisons on many large examples in the literature, hence, relaxing this constraint in our implementation with the use of regular chains is indeed desirable.

Of course, the downside of using only these two families of systems for testing is that the intersection multiplicity can be computed by hand. So, although the tables included in this section can be used to contrast the performance of the two algorithms, these are not examples in which one would need a computer algebra system to compute the intersection multiplicity. Nonetheless, they do serve as a proof of concept, illustrating that the generalization of Fulton's algorithm can be a viable alternative to intersection multiplicity algorithms which use standard bases.

All systems of polynomials are described in Appendix A. The format of the below tables is as follows:

1. Columns under the heading `iMult` denote results obtained from calling the `iMult` command and columns under the heading `IntersectionMultiplicity` denote results obtained from calling the generalization of Fulton's algorithm. The algorithm of Vrbik et al. was excluded from these tests as it returns an error on every system studied.

2. The ordering column denotes the variable ordering given to either the generalization of Fulton's algorithm in the case of `IntersectionMultiplicity`, or the variable ordering given to the polynomial ring in the case of `iMult`.

3. The Im column denotes the intersection multiplicity computed using the given algorithm

and CPU time denotes the CPU time elapsed during the respective computation.

4. Timings from SINGULAR were converted from milliseconds to the expected units corresponding to that of MAPLE's `CodeTools:-Usage` command for consistency with MAPLE's results.

5. For tests run using either MAPLE or SINGULAR, if we experience a timeout or error, we populate all larger systems in the same family of experiments with that result. That is, all cells in the given column with the same parameter d and larger n, will be automatically populated with the notation for either a timeout or error, respectively. For example, if the system simple-nql-7-8 exceeds the 2000 second time limit, then the larger system, simple-nql-8-8 will automatically be assigned a result which indicates it too experienced a timeout.

In Table 7.5 and Table 7.6, the choice of variable ordering dramatically affects the performance of both algorithms. For this discussion, we will refer to the variable ordering $x_1 \succ \ldots \succ x_n$ as the descending variable ordering and we will refer to the ordering $x_1 \prec \ldots \prec x_n$ as the ascending variable ordering. It is clear in both tables that both algorithms perform far better under the ascending variable ordering.

In Table 7.5, `iMult` outperforms the generalization of Fulton's algorithm for small values of n. As n increases, the performance of `iMult` is quickly hindered. Moreover, as the parameter d increases, the change in n required for `iMult` to experience significant delays decreases. This is best illustrated by comparing `iMult`'s performance on nql-3-8 and nql-4-8 for either the ascending or descending variable ordering. By increasing n from 3 to 4 the algorithm, `iMult` goes from terminating in a matter of milliseconds to exceeding the timeout limit. With the exception of the nql-n-4 family under the ascending ordering, `iMult` exceeds the timeout limit on most families studied in this table. It should also be noted, the error which occurs in nql-10-4 is a memory error thrown by SINGULAR which we were unable to work around.

For both variable orderings, the generalization of Fulton's algorithm experiences many of the same issues as `iMult` to a lesser degree. For the descending variable ordering, the generalization of Fulton's algorithm resists exceeding the timeout limit longer than `iMult` as n and d increase. For the systems nql-6-4, nql-5-6, and nql-4-8, the generalization of Fulton's algorithm is able to terminate in a matter of seconds while `iMult` under the same variable ordering experiences a timeout. For the generalization of Fulton's algorithm, the ascending ordering far outperforms the descending ordering. Under the ascending ordering, the generalization of Fulton's algorithm does not experience any timeout and can compute most examples in a matter of seconds. For an algorithm which seeks to serve as an alternative for when standard basis computations are not feasible, this is extremely promising. Under a bad variable ordering, the generalization of Fulton's algorithm is more resistant to timeout and long computations than `iMult`. Under a good variable ordering, the generalization far outperforms `iMult`, even as n and d increase. The reason for this is the elagent method of the generalization of Fulton's algorithm, rewriting the input system using only standard operations on polynomials, which can, in some cases, avoid long, tedious computations.

The simple-nql-n-d system, described in Table 7.6 is not as clear cut. Overall, `iMult` performs far better on the simple-nql-n-d systems than it did on the nql-n-d systems. Under the descending ordering, `iMult` also outperforms the generalization of Fulton's algorithm.

Table 7.5: iMult on nql-*n*-*d*

	iMult				IntersectionMultiplicity			
Order	$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$		$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$	
System	Im	CPU Time	Im	CPU Time	Im	CPU Time	Im	CPU Time
nql-3-4	16	<1.00ms	16	<1.00ms	16	109.00ms	16	16.00ms
nql-4-4	32	<1.00ms	32	<1.00ms	32	625.00ms	32	16.00ms
nql-5-4	64	48.16s	64	20.00ms	64	4.55s	64	47.00ms
nql-6-4	NR	>2000s	128	50.00ms	128	37.75s	128	63.00ms
nql-7-4	NR	>2000s	256	310.00ms	256	5.51m	256	109.00ms
nql-8-4	NR	>2000s	512	3.2s	NR	>2000s	512	203.00ms
nql-9-4	NR	>2000s	1024	79.5s	NR	>2000s	1024	422.00ms
nql-10-4	NR	>2000s	NR	ERROR	NR	>2000s	2048	828.00ms
nql-3-6	54	<1.00ms	54	<1.00ms	54	469.00ms	54	31.00ms
nql-4-6	162	19.21s	162	3.71m	162	5.38s	162	62.00ms
nql-5-6	NR	>2000s	NR	>2000s	486	86.11s	486	125.00ms
nql-6-6	NR	>2000s	NR	>2000s	NR	>2000s	1458	313.00ms
nql-7-6	NR	>2000s	NR	>2000s	NR	>2000s	4374	1.66s
nql-8-6	NR	>2000s	NR	>2000s	NR	>2000s	13122	3.11s
nql-9-6	NR	>2000s	NR	>2000s	NR	>2000s	39366	9.11s
nql-10-6	NR	>2000s	NR	>2000s	NR	>2000s	118098	27.86s
nql-3-8	128	30.00ms	128	70.00ms	128	1.09s	128	31.00ms
nql-4-8	NR	>2000s	NR	>2000s	512	33.28s	512	78.00ms
nql-5-8	NR	> 2000s	NR	>2000s	NR	>2000s	2048	281.00ms
nql-6-8	NR	> 2000s	NR	>2000s	NR	>2000s	8192	1.11s
nql-7-8	NR	>2000s	NR	>2000s	NR	>2000s	32768	4.33s
nql-8-8	NR	>2000s	NR	>2000s	NR	>2000s	131072	20.39s
nql-9-8	NR	> 2000s	NR	>2000s	NR	>2000s	524288	92.62s
nql-10-8	NR	> 2000s	NR	>2000s	NR	>2000s	2097152	5.87m

Table 7.6: `iMult` on simple-nql-*n*-*d*

	iMult				IntersectionMultiplicity			
Order	$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$		$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$	
System	Im	CPU Time	Im	CPU Time	Im	CPU Time	Im	CPU Time
simple-nql-4-4	256	<1.00ms	256	<1.00ms	256	547.00ms	256	63.00ms
simple-nql-5-4	1024	<1.00ms	1024	<1.00ms	1024	3.09s	1024	250.00ms
simple-nql-6-4	4096	10.00ms	4096	10.00ms	4096	16.62s	4096	437.00ms
simple-nql-7-4	16384	250.00ms	16384	200.ms	16384	79.55s	16384	2.05s
simple-nql-8-4	NR	>2000s	NR	>2000s	NR	ERROR	65536	7.50s
simple-nql-4-8	4096	<1.00ms	4096	<1.00ms	4096	8.16s	4096	219.00ms
simple-nql-5-8	32768	180ms	32768	160.00ms	32768	93.30s	32768	1.56s
simple-nql-6-8	262144	13.78s	262144	13.65s	NR	ERROR	262144	12.69s
simple-nql-7-8	NR	>2000s	NR	>2000s	NR	ERROR	12097152	99.95s
simple-nql-8-8	NR	>2000s	NR	>2000s	NR	ERROR	16777216	12.18m

Part of the reason for this is that the simple-nql-8-4 and simple-nql-6-8 systems return errors. The errors are thrown by the generalization of Fulton's algorithm after performing too many levels of recursion, exceeding the hard limit allowed by MAPLE. Because of this, it is difficult to predict how the generalization of Fulton's algorithm would perform against `iMult`, with both algorithms using the descending variable ordering, as n increases. It is possible, without this error, that the generalization of Fulton's algorithm could be faster than `iMult` under this ordering for sufficiently large n, but in practice, since there is no work around for this error, `iMult` is clearly the better choice.

Under the ascending variable ordering, the "too many levels of recursion" error is avoided, and we obtain a more accurate comparison. As expected, for small n, `iMult` outperforms the generalization of Fulton's algorithm, even more so than with the nql-*n*-*d* system. It also takes longer than with the nql-*n*-*d* system, to increase n to the point where `iMult` experiences time-outs. For sufficiently large n, the generalization of Fulton's algorithm once again outperforms `iMult`, similar to Table 7.5. This is best illustrated by the simple-nql-8-4 and simple-nql-7-8 systems, where `iMult` experiences a timeout but the generalization of Fulton's algorithm terminates in 7.5 and 99.95 seconds respectively.

7.3 Concluding Remarks for Experimental Results

In both tables, the generalization of Fulton's algorithm performs best under the ascending variable ordering. For large n, the generalization of Fulton's algorithm outperforms `iMult` under the ascending variable ordering, and in the case of the nql-*n*-*d* systems, under the descending variable ordering as well. This behaviour is indeed desirable for a standard basis-free intersection multiplicity algorithm. Unfortunately, the performance of the generalization of Fulton's algorithm is conditional upon the variable order chosen. So, although the generalization of Fulton's algorithm outperformed `iMult` for large n in our experiments, this performance was dependent on the selection of a good variable ordering. This again leads us back to the discussion of Section 6.5, and the question " what constitutes an ideal variable ordering?". Since,

the performance of the generalization varies so greatly by the choice of variable ordering, it is important to know what constitutes a good variable ordering before running the generalization of Fulton's algorithm. Moreover, as discussed in Section 5.3, the success of the generalization of Fulton's algorithm may also depend on the variable ordering. Although we do not have an answer to these questions at this time, we have two ideas of how to approach this problem, which may be of interest to the curious reader.

First, since the generalization uses operations on polynomials to rewrite the system, we speculate that any ordering for a given system, which improves the performance of polynomial system solvers, would be a strong candidate to improve the performance of the generalization of Fulton's algorithm on that system. This of course assumes the generalization of Fulton's algorithm succeeds on the given ordering. If one wishes to choose a variable ordering with instead the goal of obtaining an ordering for which the generalization of Fulton's algorithm succeeds, we speculate an ordering which minimizes modular degrees may be helpful. Additionally, consider an ordering which, for each variable and each polynomial, reduces the number of terms which are not units in the local ring, for the image of that polynomial modulo all variables below the given variable. Or, put bluntly, consider an ordering which reduces the number of terms which would fail to satisfy the condition in line 18 of Algorithm 3. Since these are the terms which could cause the generalization of Fulton's algorithm to fail, we speculate reducing the number of such terms with the choice of a variable ordering could cause the generalization of Fulton's algorithm to succeed. At this point, this is only speculation as more work is needed to make these ideas concrete, although, the intuition behind these observations is indeed quite natural and hence, we believe there is merit in these techniques for finding an ideal variable ordering.

Chapter 8

Conclusion

The generalization of Fulton's algorithm provides a powerful means of computing intersection multiplicities without the use of standard bases. In its simplest form, as presented in Algorithm 3, the generalization of Fulton's algorithm uses only standard polynomial operations to rewrite and split the input system. Aside from the benefits gained by avoiding standard bases, the generalization of Fulton's algorithm has the added benefit of being easily implementable. Since, the generalization of Fulton's algorithm only requires support for standard polynomial operations, it can be easily supported in most computer algebra systems. Indeed, no specialized libraries or engines are needed to run the generalization of Fulton's algorithm, making it highly portable between computer algebra systems.

On the other hand, the more complex presentation of the generalization of Fulton's algorithm (Algorithm 4), requires support for regular chains. This more powerful version can be applied to compute the intersection multiplicity at any point, rational or not. To our knowledge, the generalization of Fulton's algorithm and the algorithm of Vrbik et al. are the only algorithms which support n-variate intersection multiplicity computations at any point. Since, we have combined both of these algorithms under the `IntersectionMultiplicity` command as a hybrid algorithm, the resulting procedure constitutes the only procedure which supports n-variate intersection multiplicity computations at any point, greatly increasing the breadth of systems that can be handled from other intersection multiplicity algorithms in the literature.

Our experimental testing indicates that the proposed algorithms can indeed be used as an alternative to intersection multiplicity algorithms which use standard bases. Under a good variable ordering, the generalization of Fulton's algorithm can compute in seconds what competing intersection multiplicity algorithms take over half an hour to compute. Although this behaviour, nor the success of the partial algorithm, is guaranteed, the generalization of Fulton's algorithm nonetheless provides a viable alternative to intersection multiplicity algorithms which use standard bases, which was previously not possible.

8.1 Future Work

In this section we briefly discuss possible research directions and improvements pertaining to this manuscript.

8.1.1 Triangular Regular Sequences

As mentioned in Section 4.5, the optimization provided for triangular regular sequences could lead to a complete, standard basis-free, intersection multiplicity algorithm if a triangular decomposition algorithm, which preserves intersection multiplicity and regular sequences, were to be discovered. By applying such an algorithm, one could decompose a given system and apply the optimization discussed in Section 4.5 to compute the intersection multiplicity at each point in its solution set. Although finding such an algorithm would be far from trivial, we believe this direction could be promising.

8.1.2 Jacobian Optimization

The implementation of the algorithm of Vrbik et al. is optimized for systems which have intersection multiplicity one. Given a system and a point p, it suffices to test whether the Jacobian Matrix of the system is invertible at p. This optimization is implemented with the algorithm of Vrbik et al. and amounts to great speedups when applicable. The results of Table 7.4, which show the performance of `TriangularizeWithMultiplicity` using the generalization of Fulton's algorithm on systems with intersection multiplicity one, indicate this optimization is worth implementing. Indeed, Table 7.4 contains several systems for which it takes minutes to compute their intersection multiplicities using the generalization of Fulton's algorithm, even though all systems have intersection multiplicity one at all points in their intersection. By implementing this optimization for systems with intersection multiplicity one, we can reduce these times to fractions of a second.

8.1.3 Reimplementing the Algorithm of Vrbik et al.

In Chapter 7, we saw the algorithm of Vrbik et al. throw an error for many of the tests. Although the implementation throws an error whenever the criterion it seeks to apply does not hold, many of the errors we received were not due to this. In fact, most errors were due to an issue with the implementation. Hence, reimplementing the algorithm of Vrbik et al. would not only lead to a more accurate comparison, but also, a stronger overall hybrid algorithm. We also note, that in this implementation, it would be desirable to return Fail rather than throwing an error when the algorithm does not succeed, as the user may still want to know the intersection multiplicity at branches which did succeed.

8.1.4 Optimal Variable Ordering

One of the most important conclusions which follows from our experiments is the importance of the variable ordering. In Chapter 7 we observed the generalization of Fulton's algorithm succeed and fail on the same system under different variable orderings. Moreover, we observed the generalization of Fulton's algorithm perform extremely well under some orderings while performing slower under others. These observations motivate the study of different variable orderings as they pertain to the generalization of Fulton's algorithm. Some properties of variable orderings which we believe will help improve the generalization of Fulton's algorithm were discussed in Section 7.3.

8.2 Summary

To conclude, we summarize the main benefits from the developments described in this manuscript.

1. The generalization of Fulton's algorithm provides a viable alternative to intersection multiplicity algorithms which rely on standard bases. Experimentally, the generalization of Fulton's algorithm outperforms intersection multiplicity algorithms which use standard bases on large polynomial systems as well as other standard basis-free intersection multiplicity algorithms.

2. By allowing points encoded by zero-dimensional regular chains as input, our modified version of the `IntersectionMultiplicity` command can compute the intersection multiplicity at any point, rational or not.

3. The optimization provided for triangular regular sequences allows one to avoid long stacks of recursive calls, computing the intersection multiplicity immediately when applicable.

4. The simplicity of the generalization of Fulton's algorithm (at a point) makes it easily implementable in almost any computer algebra system.

Index

Bibliography

[1] Parisa Alvandi, Marc Moreno Maza, Éric Schost, and Paul Vrbik. A standard basis free algorithm for computing the tangent cones of a space curve. In Vladimir P. Gerdt, Wolfram Koepf, Werner M. Seiler, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 45–60, Cham, 2015. Springer International Publishing.

[2] Michael F. Atiyah and Ian G. Macdonald. *Introduction to commutative algebra*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1969.

[3] Dario A. Bini and Bernard Mourrain. Polynomial test suite. `http://www-sop.inria.fr/saga/POL/`. Accessed: 2022.

[4] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[5] Changbo Chen and Marc Moreno Maza. Algorithms for computing triangular decomposition of polynomial systems. *J. Symb. Comput.*, 47(6):610–642, 2012.

[6] Jin-San Cheng and Xiao-Shan Gao. Multiplicity-preserving triangular set decomposition of two polynomials. *J. Syst. Sci. Complex.*, 27(6):1320–1344, 2014.

[7] Robert M. Corless, Patrizia M. Gianni, and Barry M. Trager. A reordered schur factorization method for zero-dimensional polynomial systems with multiple roots. In Bruce W. Char, Paul S. Wang, and Wolfgang Küchlin, editors, *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC '97, Maui, Hawaii, USA, July 21-23, 1997*, pages 133–140. ACM, 1997.

[8] David A. Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. Graduate Text in Mathematics, 185. Springer-Verlag, New-York, 1998.

[9] Barry H. Dayton and Zhonggang Zeng. Computing the multiplicity structure in solving polynomial systems. In Manuel Kauers, editor, *Symbolic and Algebraic Computation, International Symposium ISSAC 2005, Beijing, China, July 24-27, 2005, Proceedings*, pages 116–123. ACM, 2005.

[10] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 4-3-0 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2022.

[11] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. About a new method for computing in algebraic number fields. In B. F. Caviness, editor, *EUROCAL '85, European Conference on Computer Algebra, Linz, Austria, April 1-3, 1985, Proceedings Volume 2: Research Contributions*, volume 204 of *Lecture Notes in Computer Science*, pages 289–290. Springer, 1985.

[12] William Fulton. *Algebraic curves - an introduction to algebraic geometry (reprint vrom 1969)*. Advanced book classics. Addison-Wesley, 1989.

[13] Jürgen Gerhard, Marc Moreno Maza, and Ryan Sandford. Computing intersection multiplicities with regular chains. In *Maple Transactions*, 2022. To appear.

[14] Gert-Martin Greuel, Santiago Laplagne, and Gerhard Pfister. Singular manual, imult. `https://www.singular.uni-kl.de/Manual/4-0-3/sing_1277.htm`. Accessed: 2022.

[15] Gert-Martin Greuel, Santiago Laplagne, and Gerhard Pfister. `normal.lib`. A SINGULAR 4-2-0 library for computing the normalization of affine rings (2020). `http://www.singular.uni-kl.de`, 2022.

[16] Gert-Martin Greuel and Gerhard Pfister. *A Singular introduction to commutative algebra*. Springer Science & Business Media, 2012.

[17] Jan Hilmar and Chris Smyth. Euclid meets bezout: Intersecting algebraic plane curves with the euclidean algorithm. *American mathematical monthly*, 117(3):250–260, March 2010.

[18] Irving Kaplansky. *Commutative rings*. The University of Chicago Press, Chicago, Ill.-London, revised edition, 1974.

[19] Hidetsune Kobayashi, Hideo Suzuki, and Yoshihiko Sakai. Numerical calculation of the multiplicity of a solution to algebraic equations. *Mathematics of computation*, 67(221):257–270, 1998.

[20] Daniel Lazard. Gröbner-bases, gaussian elimination and resolution of systems of algebraic equations. In J. A. van Hulzen, editor, *Computer Algebra, EUROCAL '83, European Computer Algebra Conference, London, England, March 28-30, 1983, Proceedings*, volume 162 of *Lecture Notes in Computer Science*, pages 146–156. Springer, 1983.

[21] Yinglin Li, Bican Xia, and Zhihai Zhang. Zero decomposition with multiplicity of zero-dimensional polynomial systems. *CoRR*, abs/1011.1634, 2010.

[22] Steffen Marcus, Marc Moreno Maza, and Paul Vrbik. On fulton's algorithm for computing intersection multiplicities. In Vladimir P. Gerdt, Wolfram Koepf, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 14th International Workshop, CASC 2012, Maribor, Slovenia, September 3-6, 2012. Proceedings*, volume 7442 of *Lecture Notes in Computer Science*, pages 198–211. Springer, 2012.

[23] Marc Moreno Maza and Ryan Sandford. Towards extending fulton's algorithm for computing intersection multiplicities beyond the bivariate case. In François Boulier, Matthew England, Timur M. Sadykov, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 23rd International Workshop, CASC 2021, Sochi, Russia, September 13-17, 2021, Proceedings*, volume 12865 of *Lecture Notes in Computer Science*, pages 232–251. Springer, 2021.

[24] Neal H. McCoy. Remarks on divisors of zero. *Amer. Math. Monthly*, 49:286–295, 1942.

[25] Hans J Stetter. *Numerical polynomial algebra*. SIAM, 2004.

[26] Jan Verschelde. Phcpack demonstration database. `http://homepages.math.uic.edu/~jan/demo.html`. Accessed: 2022.

[27] Paul Vrbik. *Computing Intersection Multiplicity via Triangular Decomposition*. PhD thesis, The University of Western Ontario, 2014.

Appendix A

Polynomial Systems

A.1 The Author's Examples

test 1-8:

- Variables: $x_1, \ldots, x_n$ for $n = 3, 5, 7, 9, 11, 13, 15, 25$ respectively.
- Equations: $x_1, x_2^2, x_3, \ldots, x_n$

test 9:

- Variables: x, y, z
- Equations: $xy - z, x^2y^3 - z, x^4 - y$

test 10:

- Variables: x, y, z
- Equations: $x^3, -x^6 + y^2, z^4$

test 11:

- Variables: x, y, z
- Equations: $zy^2, y^5 - z^2, x^5 - y^2$

test 12:

- Variables: $x_1, \ldots, x_6$
- Equations: $x_1^2 + x_2, x_2^2 + x_3, x_3^2 + x_1^2, x_4^2 + x_5, x_5^2 + x_6, x_6^2 + x_4$

A.2 Systems From The Literature

This section contains a collection of polynomial systems from [9],[3], and [26].

cbms1:

- Variables: x, y, z
- Equations: $x^3 - yz, y^3 - xz, z^3 - xy$

cbms2:

- Variables: x, y, z
- Equations: $x^3 - 3x^2y + 3xy^2 - y^3 - z^2, z^3 - 3z^2x + 3zx^2 - x^3 - y^2, y^3 - 3y^2z + 3yz^2 - z^3 - x^2$

mth191:

- Variables: x, y, z
- Equations: $x^3 + y^2 + z^2 - 1, x^2 + y^3 + z^2 - 1, x^2 + y^2 + z^3 - 1$

decker1:

- Variables: x, y
- Equations: $x^3 + xy, y^2 + y$

decker2:

- Variables: x, y
- Equations: $x + y^3, x^2y - y^4$

Ojika2:

- Variables: x, y, z
- Equations: $x^2 + y + z - 1, x + y^2 + z - 1, x + y + z^2 - 1$

Ojika3:

- Variables: x, y, z
- Equations: $x + y + z - 1, 2x^3 + 5y^2 - 10z + 5z^3 + 5, 2x + 2y + z^2 - 1$

Ojika4:

- Variables: x, y, z
- Equations:
  $6x^4z^2 - 3x^2y^2z^2 - x^2z^2 + 28x^2z - 3y^4z^3 + 2y^2z^2 + 7y^2z + z^2 - 11z + 10,$
  $x + x^3z + xy^2z - xz,$
  $10y - 2x^2yz - y^3z - yz$

Carpasse:

- Variables: x_1, x_2, x_3, x_4
- Equations:
  $x_2^2x_3 + 2x_1x_2x_4 - 2x_1 - x_3$,
  $-x_1^3x_3 + 4x_1x_2^2x_3 + 4x_1^2x_2x_4 + 2x_2^3x_4 + 4x_1^2 - 10x_2^2 + 4x_1x_3 - 10y_2x_4 + 2$,
  $2x_2x_3x_4 + x_1x_4^2 - x_1 - 2x_3$,
  $-x_1x_3^3 + 4x_2x_3^2x_4 + 4x_1x_3x_4^2 + 2x_2x_4^3 + 4x_1x_3 + 4x_3^2 - 10x_2x_4 - 10x_4^2 + 2$

KSS:

- Variables: x_1, x_2, x_3, x_4, x_5
- Equations: $f_\sigma(x_1, \ldots, x_5) = x_\sigma^2 + \sum_{v=1}^5 x_v - 2x_\sigma - 4$ for $\sigma = 1, \ldots, 5$.

DZ1:

- Variables: x_1, x_2, x_3, x_4
- Equations: $x_1^4 - x_2x_3x_4, x_2^4 - x_1x_3x_4, x_3^4 - x_1x_2x_4, x_4^4 - x_1x_2x_3$

DZ2:

- Variables: x, y, z
- Equations: $x^4, x^2y + y^4, z + z^2 - 7x^3 - 8x^2$

Solotarev:

- Variables: x_1, x_2, x_3, x_4
- Equations: $3x_1^2 - 2x_1 - x_3, x_1^3 - x_1^2 - x_1x_3 + x_3 - 2x_4 - 2, 3x_2^2 - 2x_2 - x_3, x_2^3 - x_2^2 - x_2x_3 - x_3 + 2$

For systems which appear in Table 7.4 we also include the variable ordering used in the `TriangularizeWithMultiplicity` computations.

eco5:

- Variables: x_1, x_2, x_3, x_4, x_5
- Ordering: $x_1 \succ x_2 \succ x_3 \succ x_4 \succ x_5$
- Equations:
  $(x_1 + x_1x_2 + x_2x_3 + x_3x_4)x_5 - 1$,
  $(x_2 + x_1x_3 + x_2x_4)x_5 - 2$,
  $(x_3 + x_1x_4)x_5 - 3$,
  $x_4x_5 - 4$,
  $x_1 + x_2 + x_3 + x_4 + 1$

eco6:

- Variables: $x_1, x_2, x_3, x_4, x_5, x_6$
- Ordering: $x_1 \succ x_2 \succ x_3 \succ x_4 \succ x_5 \succ x_6$
- Equations:
  $(x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5)x_6 - 1$,
  $(x_2 + x_1x_3 + x_2x_4 + x_3x_5)x_6 - 2$,
  $(x_3 + x_1x_4 + x_2x_5)x_6 - 3$,
  $(x_4 + x_1x_5)x_6 - 4$,
  $x_5x_6 - 5$,
  $x_1 + x_2 + x_3 + x_4 + x_5 + 1$

eco7:

- Variables: $x_1, x_2, x_3, x_4, x_5, x_6, x_7$
- Ordering: $x_1 \succ x_2 \succ x_3 \succ x_4 \succ x_5 \succ x_6 \succ x_7$
- Equations:
  $(x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6)x_7 - 1$,
  $(x_2 + x_1x_3 + x_2x_4 + x_3x_5 + x_4x_6)x_7 - 2$,
  $(x_3 + x_1x_4 + x_2x_5 + x_3x_6)x_7 - 3$,
  $(x_4 + x_1x_5 + x_2x_6)x_7 - 4$,
  $(x_5 + x_1x_6)x_7 - 5$,
  $x_6x_7 - 6$,
  $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + 1$

trinks:

- Variables: x, y, z, t, u, v
- Ordering: $x \succ y \succ z \succ t \succ u \succ v$
- Equations: $45y + 35u - 165v - 36$,
  $35y + 25z + 40t - 27u$,
  $25yu - 165v^2 + 15x - 18z + 30t$,
  $15yz + 20tu - 9x$,
  $-11v^3 + xy + 2zt$,
  $-11uv + 3v^2 + 99x$

Czapor Geddes 1:

- Variables: x_1, x_2, x_3, x_4, x_5
- Ordering: $x \succ y \succ z$

- Equations: $8x^2 - 2xy - 6xz + 3x + 3y^2 - 7yz + 10y + 10z^2 - 8z - 4$,
  $10x^2 - 2xy + 6xz - 6x + 9y^2 - yz - 4y - 2z^2 + 5z - 9$,
  $5x^2 + 8xy + 4xz + 8x + 9y^2 - 6yz + 2y - z^2 - 7z + 5$

A Bifurcation Problem:

- Variables: x_1, x_2, x_3
- Ordering: $x_1 \succ x_2 \succ x_3$
- Equations: $5x_1^9 - 6x_1^5x_2 + x_1x_2^4 + 2x_1x_3, -2x_1^6x_2 + 2x_1^2x_2^3 + 2x_2x_3, x_1^2 + x_2^2 - \frac{17}{64}$

quadfor2:

- Variables: x_1, x_2, w_1, w_2
- Ordering: $x_1 \succ x_2 \succ w_1 \succ w_2$
- Equations: $w_1 + w_2 - 1, w_1x_1 + w_2x_2, w_1x_1^2 + w_2x_2^2 - \frac{2}{3}, w_1x_1^3 + w_2x_2^3$

Lorentz:

- Variables: x_1, x_2, x_3, x_4
- Ordering: $x_1 \succ x_2 \succ x_3 \succ x_4$
- Equations: $x_1x_2 - x_1x_3 - x_4 + 1, x_2x_3 - x_2x_4 - x_1 + 1, -x_1x_3 + x_3x_4 - x_2 + 1, x_1x_4 - x_2x_4 - x_3 + 1$

S9_1:

- Variables: a, b, c, d, e, f, g, h
- Ordering: $a \succ b \succ c \succ d \succ e \succ f \succ g \succ h$
- Equations:
  $-eg - 2dh$,
  $9e + 4b$,
  $-4ch - 2ef - 3dg$,
  $-7c + 9a - 8f$,
  $-4df - 5cg - 6h - 3e$,
  $-5d - 6cf - 7g + 9b$,
  $9d + 6a - 5b$,
  $9c - 7a + 8$

cyclic3:

- Variables: x_1, x_2, x_3
- Ordering: $x_1 \succ x_2 \succ x_3$
- Equations: $x_1 + x_2 + x_3, x_1x_2 + x_1x_3 + x_2x_3, x_1x_2x_3 - 1$

nql-n-d:

- Variables: $x_1, \ldots, x_n$
- Equations: $x_1^d, x_i^d + x_i^{\frac{d}{2}} - x_{i-1}$ for $i = 2, \ldots, n$.

simple-nql-n-d:

- Variables: $x_1, \ldots, x_n$
- Equations: $x_1^d, x_i^d - x_{i-1}$ for $i = 2, \ldots, n$.

Since the `iMult` command only computes the intersection multiplicity at the origin, the nql-*n*-*d* and simple-nql-*n*-*d* systems have been modified from their traditional presentation. Traditionally, the first equation in both systems would be $x_1^d - 2$ rather than x_1^d. This modification is not negligible since it does change the intersection multiplicity of the traditional systems have intersection multiplicity 1 at all points for all positive values n, d.

Curriculum Vitae

Name:	Ryan Sandford
Post-Secondary Education and Degrees:	University of Western Ontario MSc Computer Science, Computer Algebra 2020 - 2022
	University of Western Ontario BSc Honours Specialization in Mathematics, Major in Computer Science 2016 - 2020
Honours and Awards:	MITACS Accelerate Fellowship (CAN$16,250) 2021-2022
	Western USRI (CAN$6,500) 2020
	NSERC USRA (CAN$8,000) 2019
Related Work Experience:	Teaching Assistant The University of Western Ontario 2020 - 2022

Publications:

1. Carranza, Daniel, Chang, Jonathan, Kapulkin, Chris, and **Sandford**, **Ryan**, "2-adjoint equivalences in homotopy type theory", *Logical Methods in Computer Science* Volume 17, Issue 1, pp. 3:1–3:9

2. Moreno Maza, Marc and **Sandford**, **Ryan**, "Towards Extending Fulton's Algorithm for Computing Intersection Multiplicities Beyond the Bivariate Case", *Computer Algebra in Scientific Computing - 23rd International Workshop, Proceedings 2021 Lecture Notes in Computer Science*, Volume 12865, pp. 232-251

3. Gerhard, Jürgen, Moreno Maza, Marc and **Sandford**, **Ryan**, "Computing Intersection Multiplicities with Regular Chains", *Maple Conference, Proceedings 2021* to appear in Maple Transactions

Talks:

1. Gerhard, Jürgen, Moreno Maza, Marc and **Sandford**, **Ryan**, "Computing Intersection Multiplicities with Regular Chains", *Maple Conference*

2. Moreno Maza, Marc and **Sandford**, **Ryan**, "Towards Extending Fulton's Algorithm for Computing Intersection Multiplicities Beyond the Bivariate Case", *Computer Algebra in Scientific Computing - 23rd International Workshop*

Technology Transfers:

1. Gerhard, Jürgen and **Sandford**, **Ryan**, development and implementation of new algorithms resulted in substantial improvements to core commands in the `AlgebraicGeometryTools` sub-package of the `RegularChains` Library, Maple 2022

2. 2020 Carranza, Daniel, Chang, Jonathan, Kapulkin, Chris, and **Sandford**, **Ryan**, contributed formalized proofs of new and existing theorems to the "2_adj directory" in the Lean 3 Homotopy Type Theory Library

Peer Review

1. Mathematics in Computer Science, Journal

2. Computer Algebra in Scientific Computing, Conference