THESIS TITLE

A Thesis

Presented to the Faculty of

Lake Forest College

in Partial Fulfillment of the Requirements for the Degree of

B.A. in Computer Science

by

Sepehr Akbari

April 2026

ABSTRACT

Your abstract goes here. Make sure it sits inside the brackets. If not, your bios-ketch page may not be roman numeral iii, as required by the graduate school.

BIOGRAPHICAL SKETCH

Your bio-sketch goes here. Make sure it sits inside the brackets.

This document is dedicated to all Cornell graduate students.

ACKNOWLEDGEMENTS

Your acknowledgements go here. Make sure it sits inside the brackets.

TABLE OF CONTENTS

Biographical Sketch iii
Dedication iv
Acknowledgements v
Table of Contents vi
List of Tables vii
List of Figures viii

1 Introduction **1**

2 Gröbner Bases **2**
2.1 Polynomial Rings and Ideals 3
2.2 Multivariate Division 5
2.3 Buchberger's Algorithm 11
2.4 Improvements to Buchberger's Algorithm 17
2.4.1 Pair Elimination 17
2.4.2 Selection Strategies 17
2.4.3 Signature-Based Algorithms 17
2.4.4 Symbolic Pre-processing 17
2.5 Complexity and Performance Metrics 17

Bibliography **18**

A Algorithm Implementation **20**

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Intro

CHAPTER 2

GRÖBNER BASES

This chapter provides a concise introduction to Gröbner bases and Buchberger's algorithm. The discussion is tailored for an audience of beginning graduate students or advanced undergraduates in mathematics and computer science. Accordingly, we assume a familiarity with standard undergraduate curricula, like calculus and linear algebra, but while a background in abstract algebra, commutative algebra, or algebraic geometry is advantageous, it is not a prerequisite for following the exposition provided here. No original work presented.

For those seeking a more comprehensive treatment, the first two chapters of [7] offer an excellent and widely recognized alternative introduction. Additional foundational perspectives can be found in [6, 12, 14], while [2] provides a more advanced theoretical framework. From a historical perspective, this work draws upon Bruno Buchberger's seminal doctoral thesis [4, 3] and his subsequent survey [5]. We also acknowledge modern optimizations that have fundamentally shaped the efficiency of the algorithm, such as the work of [10, 11, 16], which are well contextualized among other relevant works in the introductory material of [9].

The primary objective of this chapter is to establish an understanding of Buchberger's algorithm, with a particular emphasis on the role that S-pair selection strategies play in its computational performance. As the ultimate goal of this thesis is to leverage machine learning to optimize these selection processes, establishing this mathematical foundation is a critical first step.

2.1 Polynomial Rings and Ideals

The study of systems of multivariate polynomial equations is a cornerstone of both computational algebra and algebraic geometry. Consider a typical system of equations

$$\begin{cases} f_1(x,y) = x^2 - y^3 = 0 \\ f_2(x,y) = x^2 - 4y + 3 = 0 \end{cases} \tag{2.1}$$

which arise naturally across many diverse fields, like robotics [1], signal processing [13], and statistics [8, 15]. The most fundamental question regarding such a system is the *consistency* problem, which asks whether there exists an exact common solution $(x_0, y_0) \in \mathbb{C}^2$ such that all equations are simultaneously satisfied.

To approach this problem algebraically, we work within a formal structure known as a polynomial ring.

Definition 2.1.1 (Polynomial Ring)**.** The polynomial ring $R = k[x_1, \ldots, x_n]$ is the set of all polynomials in variables $x_1, \ldots, x_n$ with coefficients from a field k (typically $\mathbb{Q}, \mathbb{R}$, or $\mathbb{C}$). R is an associative algebra equipped with standard addition and multiplication operations for polynomials.

One powerful way to show that a system has no solution is to find a contradiction through a linear combination of its generators. If we can find polynomials $a_1, a_2 \in R$ such that:

$$a_1(x,y)f_1(x,y) + a_2(x,y)f_2(x,y) = 1, \tag{2.2}$$

then the system must be inconsistent. Indeed, any hypothetical solution (x_0, y_0) would force the left-hand side of (2.2) to zero, contradicting the fact that the right-hand side is the constant 1. According to Hilbert's Nullstellensatz, over an algebraically closed field like $\mathbb{C}$, the converse is also true; as in, a system has no common zeros if and only if 1 can be expressed as a polynomial combination of the generators.

This observation shifts our focus from the equations themselves to the set of all possible polynomial combinations they can generate. This set is known as an ideal.

Definition 2.1.2 (Ideal)**.** Let $f_1, \ldots, f_s$ be polynomials in R. The ideal generated by these polynomials, denoted $\langle f_1, \ldots, f_s \rangle$, is the set

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} a_i f_i \mid a_i \in R \right\}. \tag{2.3}$$

Algebraically, an ideal is a sub-module of R that is closed under addition and under multiplication by any element of R. Conceptually, an ideal I represents the collection of all polynomial consequences of its generators; if a point is a zero for every generator of I, it is necessarily a zero for every polynomial in I.

The consistency of the system in (2.1) is therefore equivalent to the ideal membership problem; determining whether the constant polynomial 1 lies within the ideal $I = \langle x^2 - y^3, x^2 - 4y + 3 \rangle$. While it may not be immediately obvious for this specific example, the system (2.1) does in fact possess solutions (for instance, $(-1, 1)$), meaning $1 \notin I$. We could easily imagine more complex systems of polynomial equations, where assessing membership is far less straightforward. Thus, developing a systematic algorithmic test for such membership is the primary motivation for the theory of Gröbner bases.

2.2 Multivariate Division

In the previous section, we established that the solvability of a system of polynomial equations reduces to the ideal membership problem, which states, given a polynomial f and an ideal $I = \langle f_1, \ldots, f_s \rangle$, is $f \in I$?

In the univariate case, where $R = k[x]$, this problem is trivialized by the Euclidean division algorithm. Since $k[x]$ is a principal ideal domain, any ideal is generated by a single greatest common divisor, say g. To check if $f \in \langle g \rangle$, one simply computes the remainder of f divided by g. If the remainder is 0, then $f \in I$; otherwise, it is not. Let's illustrate this with a simple example.

Example 2.2.1. Consider the ideal $I = \langle x^2 - x \rangle$ in $\mathbb{Q}[x]$, and lets check if $h_1(x) = x^3 - x^2$ and $h_2(x) = x^3 - 1$ are elements of I. Dividing h_1 by $x^2 - x$ using polynomial long division

$$x^3 - x^2 = (x)(x^2 - x) + 0$$

yields a remainder of 0, confirming that $h_1 \in I$. Conversely, for h_2, we have

$$x^3 - 1 = (x + 1)(x^2 - x) + (x - 1)$$

with a non-zero remainder of $x - 1$, indicating that $h_2 \notin I$. This is a direct and trivial application of the division algorithm in one variable.

However, generalizing this logic to the multivariate ring $k[x_1, \ldots, x_n]$ introduces complications. Consider another example.

Example 2.2.2. Let $I = \langle x^2y - x, x^2 + y^3 \rangle$ in the ring $\mathbb{Q}[x, y]$, and let's check if $h_3(x, y) = x^2y + y^4$ is an element of I. Here, I is generated by two polynomials, its elements can be expressed as combinations of the generators: $a_1(x^2y - x) +$

$a_2(x^2 + y^3)$ for some $a_1, a_2 \in \mathbb{Q}[x, y]$. This example follows the same principle as before, but considering that a_1 and a_2 can be any polynomials in two variables, the search space is vastly larger. Other than impractical trial and error, there is no straightforward method to determine if such a_1 and a_2 exist that satisfy the equation $h_3 = a_1(x^2y - x) + a_2(x^2 + y^3)$.

Our goal is to apply our solution for Example 2.2.1 to Example 2.2.2 by developing a multivariate division algorithm. However, two main issues arise in this generalization.

Firstly, in Example 2.2.1, there only exists one polynomial divisor to be used in the division algorithm, but in Example 2.2.2, there are two generators in the ideal, leading to multiple possible divisors to choose from at each step. We can overcome this problem by simply choosing one of the divisors in each step and keeping track of a quotient for each divisor. The final result in the univariate case is expressed as $f = aq + r$, where a is the quotient, q is the divisor, and r is the remainder. In the multivariate case with multiple divisors, we can express the result as $f = a_1q_1 + \cdots + a_sq_s + r$, where each a_i is the quotient corresponding to divisor q_i. This expression still captures the essence of the division algorithm, but now accommodates multiple divisors.

Secondly, notice that in the division algorithm we divide the leading term of the dividend by the leading term of the divisor, giving importance to what we consider the leading term, or the largest term in a polynomial. In the univariate case, this is straightforward since there is a natural ordering of terms by degree ($x^2 > x$). In the multivariate case, however, there is no inherent way to order terms like x^2y and xy^2. To address this, we need to follow a systematic rule for ordering monomials in multiple variables to ensure consistency in determining leading terms. We denote a monomial $x_1^{\alpha_1} \dots x_n^{\alpha_n}$ as x^α, where $\alpha \in \mathbb{Z}^n_{\geq 0}$.

Definition 2.2.1 (Monomial Ordering)**.** A monomial ordering on $k[x_1, \dots, x_n]$ is a relation $>$ on the set of monomials such that $>$ is a total ordering (any two monomials are comparable), $>$ is compatible with multiplication, as in, if $x^\alpha > x^\beta$, then $x^\alpha x^\gamma > x^\beta x^\gamma$ for all γ, and $>$ is a well-ordering (every non-empty set of monomials has a least element).

The conditions of monomial ordering are set to guarantee consistency and termination of the division algorithm. The total ordering condition ensures that we can always identify a leading term in any polynomial. The compatibility condition guarantees that the ordering behaves predictably under multiplication, and the well-ordering condition is specially critical for computer science applications, as it guarantees that any algorithm reducing the size of a polynomial based on this order will eventually terminate. While there are infinitely many such orderings, two are very frequently used in computational algebra; lexicographic order (lex) and graded reverse lexicographic order (grevlex).

Definition 2.2.2 (Lexicographic Order)**.** Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. The lexicographic order is defined by $x^\alpha >_{lex} x^\beta$ if the leftmost non-zero entry of the vector difference $\alpha - \beta$ is positive.

Intuitively, Lex behaves like a dictionary sort. For variables $x > y > z$, we have $x > y^5 z^3$ because the x-exponent dominates. On the other hand, $y^5 z^3 > y^4 z^{10}$ because the y-exponent is larger.

Definition 2.2.3 (Graded Reverse Lexicographic Order)**.** Graded reverse lexicographic order is defined by $x^\alpha >_{grevlex} x^\beta$ if $|\alpha| > |\beta|$, or if $|\alpha| = |\beta|$ and the rightmost non-zero entry of $\alpha - \beta$ is negative.

Grevlex compares total degree first. For example, $x > y > z$, $y^2 > xz$ since $2 > 1$. If total degrees are equal, it favors monomials that have fewer variables from the end of the ordering. Thus, $xy > xz$ because the y-exponent is larger than the z-exponent. For computation, grevlex is often more efficient than lex as it tends to keep coefficients smaller.

With a fixed monomial order, we can define the leading term of a polynomial f, denoted $\mathrm{LT}(f)$, as well as its coefficient $\mathrm{LC}(f)$ and monomial $\mathrm{LM}(f)$.

Given a monomial order and a set of divisors $F = \{f_1, \ldots, f_s\}$, we can formulate a multivariate polynomial long division algorithm with the goal to write f as

$$f = a_1 f_1 + \cdots + a_s f_s + r$$

where no term in the remainder r is divisible by any $\mathrm{LT}(f_i)$. Algorithm 1 outlines this process.

Definition 2.2.4 (Reduction)**.** When a polynomial h is divided by a set of polynomials $F = \{f_1, \ldots, f_s\}$ using the multivariate division algorithm, the resulting polynomial r is written as $\mathrm{reduce}(h, F)$ and is said that h reduces to r.

Algorithm 1: Multivariate Division Algorithm

```
**Input** $h, F = \{f_1, \dots, f_s\}, >$
**Output** $r$
  $p \leftarrow h, r \leftarrow 0$
  **while** $p \neq 0$ **do**
      **if** $\exists i$ s.t. $LT(f_i) \mid LT(p)$ **then**
          $p \leftarrow p - \frac{LT(p)}{LT(f_i)} f_i$
      **else**
          $r \leftarrow r + LT(p), \quad p \leftarrow p - LT(p)$
      **end if**
  **end while** and **return** $r$
```

We denote the result of this process as $r = \text{reduce}(h, F)$. If $\text{reduce}(h, F) = 0$, we have explicitly constructed coefficients a_i such that $h = \sum a_i f_i$, proving that $h \in I$. However, unlike the univariate case, a non-zero remainder does *not* prove that $h \notin I$. The output of the division algorithm depends heavily on the order in which the divisors f_i are listed.

To demonstrate this, let us apply Algorithm 1 to Example 2.2.2, using the Lexicographic order with $x > y$. In this example, $f_1 = x^2y - x$ and $f_2 = x^2 + y^3$, and we want to check if $h = x^2y + y^4$ is in the ideal I. First, we list the divisors as $I = \langle f_1, f_2 \rangle$, so the algorithm checks f_1 first. In this case, the final result is $r = x + y^4 \neq 0$, suggesting that $h \notin I$. If we then list the divisors as $I = \langle f_2, f_1 \rangle$. The algorithm checks f_2 first, in which case the final result is $r = 0$, indicating that $h \in I$. Since we found a remainder of 0 in the second case, we have proven that $h \in I$. This example highlights that the standard multivariate division algorithm is not sufficient for the ideal membership problem because the remainder is not unique. Have in mind, that this is a simple example with only two generators, and in practice, ideals can have many generators and the orderings can be much more complex.

The failure of the division algorithm arises because the leading terms of the generators $\{f_1, \ldots, f_s\}$ may not cover all the leading terms of the ideal I. Cancellation can occur between generators during linear combination, producing a new polynomial in I with a leading term not divisible by any $\mathrm{LT}(f_i)$. To resolve this, we could try to start the division process with a divisor that is an element of I itself. However if the leading term of this element is not divisible by any $\mathrm{LT}(f_i)$, we cannot proceed with the division. Thus, we need a generating set of I such that every element of I has its leading term divisible by some leading term of the generating set. A Gröbner basis is a generating set with this property.

Definition 2.2.5 (Gröbner Basis)**.** Let there be a fixed monomial order. A finite subset of generators $G = \{g_1, \ldots, g_t\} \subset I$ is a Gröbner basis for a non-zero ideal I if

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle = \langle \{\mathrm{LT}(f) \mid f \in I\} \rangle$$

This definition implies two equivalent properties that resolve the issues discussed above:

(i) $f \in I \iff \mathrm{reduce}(f, G) = 0$

(ii) $\mathrm{reduce}(f, G)$ is unique $\forall f \in R$ regardless of the order of g_i.

Property (i) directly answers Example 2.2.2, as if we could find a Gröbner basis G for I, we could simply compute $\mathrm{reduce}(h_3, G)$ to determine if $h_3 \in I$ without taking account of the order of generators. Property (ii) ensures that the multivariate division algorithm behaves consistently when using a Gröbner basis, making it a reliable tool for computations in polynomial ideals. This consistency and reliability makes Gröbner bases a widely used tool in computational algebraic geometry and computer algebra systems.

2.3 Buchberger's Algorithm

While the existence of Gröbner bases is theoretically guaranteed, their practical utility depends on our ability to compute them explicitly. In 1965, Bruno Buchberger introduced the fundamental algorithm for constructing these bases in [4, 3]. The core of his method relies on detecting and repairing "ambiguities" where the multivariate division algorithm fails to be unique.

The primary obstruction to uniqueness in multivariate division is the cancellation of leading terms. If we have two polynomials $f, g \in I$, it is possible that a linear combination $af + bg$ causes the leading terms to cancel, revealing a new leading term of lower order that was previously hidden. Buchberger identified that we do not need to check all random linear combinations. We only need to check the minimal cancellation between the leading terms of pairs of generators.

Definition 2.3.1 (S-polynomial)**.** Let $f, g \in k[x_1, \ldots, x_n]$ be non-zero polynomials. Let $x^\gamma = \text{lcm}(\text{LM}(f), \text{LM}(g))$ be the least common multiple of their leading monomials. The S-polynomial of f and g is defined as

$$S(f, g) = \frac{x^\gamma}{\text{LT}(f)} \cdot f - \frac{x^\gamma}{\text{LT}(g)} \cdot g$$

The S-polynomial is constructed specifically to cancel the leading terms of f and g. It is very likely that this cancellation reveals a new leading term in the ideal that is not divisible by either LT(f) or LT(g). In other words, if this cancellation results in a polynomial that cannot be reduced to zero by the current set of generators [1], we have found a new polynomial that needs to be added to our generating set to ensure it covers all leading terms of the ideal.

Now suppose $f \in I$, then by Definition 2.1.2, f can be expressed as

$$f = \sum_{i=1}^{s} a_i g_i$$

for some $a_i \in R$ and $g_i \in G$. Observe that each term in this sum has a lead term $\mathrm{LT}(a_i g_i) = \mathrm{LT}(a_i)\mathrm{LT}(g_i)$, which is divisible by $\mathrm{LT}(g_i)$. The only way that f has a leading term not divisible by any $\mathrm{LT}(g_i)$ is if there is cancellation among these leading terms. We can trace this cancellation back to pairs of generators g_i, g_j whose leading terms cancel in the sum. The minimal such cancellation is exactly the S-polynomial $S(g_i, g_j)$, and by our assumption that $f \in I$, we must have $S(g_i, g_j) \in I$ as well. Therefore, if G is a Gröbner basis, by property *(i)* from Definition 2.2.5, we must have $\mathrm{reduce}(S(g_i, g_j), G) = 0$ for all pairs i, j. Conversely, if this condition holds for all pairs, then any polynomial $f \in I$ cannot have a leading term that escapes divisibility by some $\mathrm{LT}(g_i)$, because any such escape would trace back to a non-zero remainder from some S-polynomial. This leads us to Buchberger's Criterion.

Theorem 2.3.1 (Buchberger's Criterion)**.** Let $G = \{g_1, \ldots, g_s\}$ be a generating set for an ideal I. Then G is a Gröbner basis for I if and only if:

$$\mathrm{reduce}(S(g_i, g_j), G) = 0$$

for all pairs $1 \leq i < j \leq s$.

[1]It is fascinating that the S-polynomial reveals all potential ambiguities or problems with a generating set.

This theorem transforms the definition of a Gröbner basis from an abstract property about infinite sets (all polynomials in I) into a finite computational check. Buchberger's algorithm is a direct application of the criterion above, which computes a Gröbner basis of I from any starting generating set of I, by iteratively checking all S-polynomials of our current generating set. If any S-polynomial reduces to a non-zero remainder, we force them to reduce to zero, by adding their reduction to our generating set. This process continues to reveal even more S-polynomials to check, until all S-polynomials reduce to zero. At this point our generating set is a Gröbner basis.

An outline of Buchberger's algorithm is provided in Algorithm 2. There are three main functions used in the algorithm. We assume we have a function that returns a pair from the set of pairs P. This selection could be as simple as treating the pair set as a queue data structure and returning the first element at each iteration. We also need a function *reduce* that implements polynomial long division and returns the remainder, as defined in Algorithm 1. Finally, an *update* function is needed to update the set of pairs P when a new generator is added to the basis, defined as

$$\text{update}(P, G, r) = P \cup \{(r, h) \mid h \in G\}$$

where r is the new polynomial added to the basis G, and h iterates over all existing generators in G.

Algorithm 2: Buchberger's Algorithm

```
Input $F = \{f_1, \ldots, f_s\}$
Output Gröbner basis $G$ for $I = \langle F \rangle$
1: $G \leftarrow F$
2: $P \leftarrow \{(f_i, f_j) \mid f_i, f_j \in G, f_i \neq f_j\}$
3: while $P \neq \emptyset$ do
4:     $(f_i, f_j) \leftarrow \text{select}(P)$
5:     $P \leftarrow P \setminus \{(f_i, f_j)\}$
6:     $r \leftarrow \text{reduce}(S(f_i, f_j), G)$
7:     if $r \neq 0$ then
8:         $P \leftarrow P \cup \{(r, h) \mid h \in G\}$
9:         $G \leftarrow G \cup \{r\}$
10:    end if
11: end while
12: return $G$
```

The termination of the Buchberger algorithm is not immediately obvious, as each iteration can potentially add new polynomials to G, leading to more S-polynomials to check, and thus an unbounded loop. However, the algorithm is guaranteed to terminate by considering the monomial ideal generated by the leading terms of the elements in G. Each time a non-zero remainder $r = \text{reduce}(S(f_i, f_j), G)$ is added to G, its leading term $\text{LT}(r)$ is not divisible by any of the leading terms of the current generators. Consequently, the new monomial ideal $\langle \text{LT}(G \cup \{r\}) \rangle$ strictly contains the previous ideal $\langle \text{LT}(G) \rangle$. This process creates a strictly ascending chain of monomial ideals

$$\langle \mathrm{LT}(g_1)\rangle \subsetneq \langle \mathrm{LT}(g_1), \mathrm{LT}(g_2)\rangle \subsetneq \langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), \mathrm{LT}(g_3)\rangle \subsetneq \dots$$

According to the Ascending Chain Condition (ACC), every ascending chain of ideals in the polynomial ring $k[x_1, \dots, x_n]$ over the field k (or any Noetherian ring) must stabilize. In the context of monomial ideals, this is a direct consequence of Dickson's Lemma, as shown in [7], which states that every monomial ideal is finitely generated. Therefore, the chain cannot increase indefinitely, ensuring the algorithm terminates in a finite number of steps.

As proved in [7], Algorithm 2 finds a Gröbner basis for any ideal I. However, in many cases the output of Buchberger's algorithm often contains redundant elements.

Example 2.3.1. Consider the ideal $I = \langle x^2, xy + y, x^2y + z\rangle$ in $\mathbb{Q}[x, y, z]$. Using Grevlex ordering, Buchberger's algorithm produces the Gröbner basis

$$G = \left\{ x^2,\ xy + y,\ x^2y + z,\ y,\ -z \right\}$$

Notice that $\mathrm{LT}(x^2y + z) = x^2y$ is divisible by $\mathrm{LT}(x^2) = x^2$. So $x^2y + z$ is redundant in the sense that removing it from G does not change the ideal generated by the leading terms.

To address this redundancy, we can remove these redundant polynomials from G without losing the Gröbner basis property, to obtain a minimal Gröbner basis.

Definition 2.3.2 (Minimal Gröbner Basis)**.** A Gröbner basis G is minimal if $\mathrm{LT}(g_i)$ does not divide $\mathrm{LT}(g_j)$ for any $i \neq j$.

Example 2.3.2. Continuing from Example 2.3.1, we can rewrite the Gröbner basis G as

$$G_{minimal} = \left\{ x^2,\ y,\ -z \right\}$$

which is a minimal Gröbner basis for the ideal I. Observe that removing the redundant polynomials did not change the ideal generated by the leading terms, as $\langle x^2, y, -z\rangle = \langle x^2, xy + y, x^2y + z\rangle$.

A powerful property in mathematics is uniqueness, which allows us to identify canonical representatives for equivalence classes. In the context of Gröbner bases, we can further refine the concept of minimality to achieve uniqueness among all Gröbner bases for a given ideal with a fixed monomial order.

Definition 2.3.3 (Reduced Gröbner Basis)**.** A Gröbner basis G is reduced if, for all $g \in G$ no monomial appearing in g is divisible by any $\mathrm{LT}(h)$ for $h \in G \setminus \{g\}$, and, $\mathrm{LC}(g) = 1$ (monic polynomials).

Example 2.3.3. Continuing from Example 2.3.2, we can further reduce the minimal Gröbner basis $G_{minimal}$ to obtain

$$G_{reduced} = \left\{ x^2,\ y,\ z \right\}$$

which is a reduced Gröbner basis for the ideal I. Note that we changed $-z$ to z to make it monic.

Similar to many powerful tools in mathematics, like reduced row echelon form in linear algebra, the significance of reduced Gröbner bases lies in their uniqueness. For any ideal I and a fixed monomial order, there exists a unique reduced Gröbner basis. Most implementations of Buchberger's algorithm, including the *gb* function in *Macaulay2*, automatically return this reduced Gröbner basis through the same process shown in Examples 2.3.1, 2.3.2, and 2.3.3.

2.4 Improvements to Buchberger's Algorithm

2.4.1 Pair Elimination

2.4.2 Selection Strategies

2.4.3 Signature-Based Algorithms

2.4.4 Symbolic Pre-processing

2.5 Complexity and Performance Metrics

BIBLIOGRAPHY

[1] Rafał Abłamowicz. *Some Applications of Gröbner Bases in Robotics and Engineering*, pages 495–517. Springer London, London, 2010.

[2] William W. Adams and Philippe Loustaunau. *An Introduction to Gröbner Bases*, volume 3rd. American Mathematical Society, 1994.

[3] Bruno Buchberger. *An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal*. PhD thesis, University of Innsbruck, 1965.

[4] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 1965.

[5] Bruno Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. pages 184–232, 1985.

[6] David Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. Springer, 2nd edition, 2005.

[7] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 5th edition, 2025.

[8] Persi Diaconis and Bernd Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of Statistics*, 26(1):363 – 397, 1998.

[9] Cristian Eder. *Signature-based algorithms to compute standard bases*. PhD thesis, University of Kaiserslautern, 2012.

[10] Rüdiger Gebauer and H. Michael Möller. On an installation of buchberger's algorithm. *Journal of Symbolic Computation*, 6(2):275–286, 1988.

[11] Alessandro Giovini, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. "one sugar cube, please" or selection strategies in the buchberger algorithm. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, ISSAC '91, pages 49–54, New York, NY, USA, 1991. Association for Computing Machinery.

[12] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1*. Springer Verlag, 2000.

[13] Zhiping Lin, Li Xu, and Qinghe Wu. Applications of gröbner bases to signal and image processing: a survey. *Linear Algebra and its Applications*, 391:169–202, 2004. Special Issue on Linear Algebra in Signal and Image Processing.

[14] Bernd Sturmfels. What is a gröbner basis? *Notices of the American Mathematical Society*, 52(10), 2005.

[15] Seth Sullivant. *Algebraic Statistics*, volume 194. Graduate Studies in Mathematics, American Mathematical Society, 2018.

[16] Y. Sun and D. Wang. The f5 algorithm in buchberger's style. *Journal of Systems Science and Complexity*, 24(6):1218–1231, 2011.

APPENDIX A

ALGORITHM IMPLEMENTATION

Macaulay2 code for the multivariate polynomial division algorithm and ideal membership check is provided below.

```
multivariateDivision = (h, F) -> (
R := ring h;

p := h;
r := 0_R;

while p != 0 do (
    divisible := false;
    ltP := leadTerm p;

    for i from 0 to #F-1 do (
        f := F#i;
        ltF := leadTerm f;

        if ltP % ltF == 0 then (
            factor := ltP // ltF;
            p = p - factor * f;
            divisible = true;
            break;
        );
    );
    if not divisible then (
            r = r + ltP;
            p = p - ltP;
        );
    );
    return r;
);

checkMembership = (h, F) -> (
    remainder := multivariateDivision(h, F);
    if remainder == 0 then (
        print("Result: h is in the ideal generated by F.");
        return true;
    ) else (
        print("Result: remainder is non-zero (" | toString(remainder)
        print("Note: If F is a Groebner Basis, h is NOT in the ideal.
        return false;
    );
);

-- Usage for Example 2.2.2
R = QQ[x, y, MonomialOrder => GRevLex];
F = {x^2 * y - x, x^2 + y^3};
h = x^2 * y + y^4;
checkMembership(h, F);
```