New Strategies for Computing Gröbner Bases

Bruno Simões

Department of Mathematics

University Trento

Advisor:

Prof. Lorenzo Robbiano

University of Genova

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

April 12, 2013

Abstract

Gröbner bases are special sets of polynomials, which are useful to solve problems in many fields such as computer vision, geometric modeling, geometric theorem proving, optimization, control theory, statistics, communications, biology, robotics, coding theory, and cryptography.

The major disadvantage of algorithms to compute Gröbner bases is that computations can use a lot of computer power. One of the reasons is the amount of useless critical pairs that the algorithm has to compute. Hence, a lot of effort has been put into developing new criteria to detect such pairs in advance.

This thesis is devoted to describe efficient algorithms for the computation of Gröbner bases, with particular emphasis to those based on polynomial signatures. The idea of associating each polynomial with a signature on which the criteria and reduction steps depend has become extremely popular in part due to its good performance.

Our main result combines the criteria from Gao-Volny-Wang's algorithm with the knowledge of Hilbert Series. A parallel implementation of the algorithm is also investigated to improve the computational efficiency. Our algorithm is implemented in CoCoALib, a C++ free library for computations in commutative algebra.

Acknowledgements

It is a pleasure to thank those who have helped and encouraged me throughout the long and difficult process of completing this thesis. First and foremost I am heartily thankful to my supervisor, Prof. Dr. Lorenzo Robbiano, whose good supervision, encouragement, guidance and support from the initial to the final level allowed me to develop an understanding of the subject. I appreciate all his contributions of time, ideas, and suggestions to make my Ph.D experience productive and stimulating.

I would like to thank Prof. Dr. Patrizia Gianni for her priceless suggestions, Prof. Dr. Ragni Piene and Prof. Dr. Alicia Dickenstein for extending my knowledge about Hilbert Series, and Anna Bigatti and John Abbot for their advise and fruitful discussions during the development of the package for CoCoALib.

The generous support provided by the European Commission through the Marie Curie ITN Network SAGA was greatly appreciated. Without the support, my ambition to study abroad could hardly be realized. Besides, I would like to acknowledge Fondazione Graphitech for hosting me and for providing me the opportunity and the resources required to do my research activity.

Lastly, I owe my deepest gratitude to my whole family, who raise me with love and knowledge and who supported me in all my pursuits. I have no suitable word that can fully describe their everlasting love to me.

Finally, I am thankful to all those who supported me in any respect during the completion of my studies.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1

Introduction

The wisest mind
has something yet to learn.
(George Santayana)

The fields of Geometric and Solid Modeling have received a lot of attention from the academic and industrial communities throughout the past four decades because of their immediate applicability. Yet, many of the issues concerning the representation and manipulation of the geometric objects remain unsolved. The increasing complexity of problems emerging from recent applications, which are now higher-dimensional than before, and the lack of generality of the previous solutions, are some of the obstacles to an effective solution.

In these fields, the geometric formulations are often described in terms of algebraic representations, e.g. curves and surfaces. Although there are other representations like meshes that are more suitable for computer rendering, they lack of geometric compactness. Hence, the understanding and manipulation of the geometrical structure is usually more difficult. Algebraic representations, on the other hand, are also useful to describe their respective geometric constraints, e.g. mechanical assembly planning [Anantha et al., 1996; Cambon et al., 2009], tolerance analysis, constraint-based sketching and design [Michalik et al., 2002], kinematic analysis of robots [Nielsen and Roth, 1999], collision detection [Jia et al., 2011], geometric manipulation [Cambon et al., 2009], manipulation of offsets of curves and surfaces [Hoffmann, 1990] and geometric theorem proving [Kreuzer and Robbiano, 2005; Hoffmann, 1989]. Therefore, these representations can be used simultaneously to represent the geometry and to describe their constraints. The solution to those geometric constraints arises from the solution of the corresponding algebraic equations.

Additionally, algebraic solvers have also their advantages, such as generality, dimension independence, and the natural ability to deal with symbolic constraints. Some of these advantages have long been identified as important research problems in these fields. Unfortunately, most algebraic solvers reduce problems to a search of polynomials roots in the univariate case, which can be ill-conditioned for polynomials of degree greater than 14, as shown by Wilkinson [1959]. As a result, the implementation of these algebraic methods with finite arithmetic precision is considered to be challenging as it constrains the performance of the resulting algorithms. The goal of this thesis is to investigate new strategies to improve their computational performance.

Algebraic methods can be classified into numerical and symbolic methods. Numeric methods can be sub-categorized into homotopy and iterative methods. Iterative methods, essentially, compute a sequence of improving approximate solutions to the problem. In general, they require a good initial guess of the intended solution to guarantee convergence, which is often difficult to provide. The most widely used method is the Newton-Raphson [Ypma, 1995]. Newton-Raphson is a local method and converges much faster than relaxation [Hillyard and Braid, 1978]. Examples of solvers that use these methods are, for example, the solvers described by Light and Gossard [1982]; Lin et al. [1981]. Homotopy or continuation, see for example [Allgower and Georg, 1993], is a family of methods with a growing popularity that rely mostly on path-following techniques in the complex space. These methods are global and guarantee convergence. Moreover, they are exhaustive and allow to determine all solutions of a constraint problem. However, their efficiency is often worse than that of Newton-Raphson, and usually they require a solid theoretical background of the subject.

Symbolic methods based on algorithms for computing Gröbner bases or Resultants can be used for eliminating indeterminates, and thereby reducing problems to a search of polynomials roots in the univariate case. Alasdair and de Pennington [1993], described a solver built on top of the Buchberger's algorithm [Buchberger, 1985], and Kondo [1992] introduced a symbolic algebraic method that works by generating a polynomial that summarizes the changes undergone by the system of equations. In this thesis, we shall focus on algorithms to compute a Gröbner basis. Our choice is justified by the fact that they have many other properties useful to these fields, see Section 3.4.

Although, algebraic methods based on the computation of Gröbner bases are known to be efficient only for polynomial systems of low degree, the prevailing viewpoint is that they usually lead to a better theoretical understanding of the problems, which can uncover new opportunities in the upcoming paradigms of smart geometry. Moreover, they can be extremely useful and very practical when used to pre-process and study

specific constraint systems.

This thesis is devoted towards new criteria to improve the overall efficiency of algorithms to compute a Gröbner basis. To that end, we have developed a new strategy that uses the knowledge about the Hilbert Series. The algorithm is further improved through parallelization techniques.

1.1 What is a Gröbner basis?

Let $\mathcal{J}$ be a set of polynomials. A Gröbner basis for a set of polynomials $\mathcal{J}$ is another set of polynomials, which is equivalent, and has certain computational properties. In fact, many practical problems involving ideals can be solved once we compute such basis. For example, we can solve the ideal membership problem, that is, we can check using conceptually simple algorithms, whether a polynomial f can be written as an algebraic combination of polynomials of $\mathcal{J}$. In other words, whether f is in the ideal $\mathcal{I}$ given by the polynomials $\mathcal{J}$. Technically speaking, if f is in the ideal $\mathcal{I}$, then there must exist a sequence of subtractions from f, by multiples of polynomials in a Gröbner basis of $\mathcal{I}$, that reduces the polynomial to zero. Whether such sequence can be found easily depends on specific properties of the generators.

The invention of the theory of Gröbner bases can be associated with several mathematicians. In our view, the major step was taken by Bruno Buchberger during his PhD thesis [Buchberger, 1965]. First, he formulated the concept of Gröbner bases, extending a suggestion of his advisor Wolfgang Gröbner. Such influence is also acknowledged with his terminology. Lastly, he found an algorithm to compute them, as well as a proof for the fundamental theorem on which the correctness and termination of the algorithm hinges. However, his work was only fully appreciated after the eighties, that is, when researchers in mathematics and computer science started to realize the importance of this new theory, mainly due to its simplicity and power.

1.2 Thesis Overview

In 1965, Buchberger introduced the first algorithm for computing a Gröbner basis, which ended up to be an important tool in computational algebra. The importance of this milestone is related to the fact that this scholar provided us, for the first time, with a systematic way of calculating sums of ideals, products of ideals, intersections of ideals, etc. As mentioned before, his algorithm has the advantage of being easy to understand and relatively simple to implement, mostly due to the limited theoretical background

required to use it, and that makes it accessible to wider audiences. However, it suffers from poor performance on large problems and it is sensitive to several factors such as the choice of a term ordering, the number of indeterminates, etc.

All algorithms that aim to compute a Gröbner basis follow the basic blueprint of Buchberger's algorithm, but recent algorithms introduce a new point of view. We call these new approaches "signature-based strategies", inasmuch as their computations take "polynomial signatures" into account. The idea behind our main results is to combine, without harming the performance or causing wrong results, the criteria from signature-based strategies with classical improvements based upon the Hilbert Series. A parallel implementation of the algorithm is also investigated.

This thesis is structured into six chapters. In Chapter 2, we recall some of the needed vocabulary and concepts, for the benefit of the non-specialist. Once presented with the roadmap, the reader can refer to books on algebra or algebraic geometry for further details. The central data structure in the chapter is the one of polynomial, which is used to construct polynomial rings.

In Chapter 3, we bring the reader into the realms of Gröbner bases. First, we recall their formal definition and all the details on how to construct them. Then, we describe a few practical applications, followed by a brief introduction to their computational complexity. Lastly, we present a survey about existent criteria used to improve their computational performance.

Chapter 4 begins with an introduction to the concept of syzygies and free resolutions, which are required to fully understand the proposed algorithms. Then, we introduce a new strategy to compute the Gröbner bases, based on polynomial signatures, and we show that an extension of the current state-of-art is still possible by taking advantage of the knowledge of Hilbert Series. Parallelization attempts are also investigated in this chapter.

In Chapter 5, we compare the performance of some of the fastest algorithms to compute a Gröbner basis. All algorithms are implemented in CoCoALib [2013], and all examples are commonly used to benchmark Gröbner basis algorithms. We shall see from the results that proposed algorithms provide a good alternative.

We conclude this thesis with a shortlist of remarks and some future directions regarding the current results.

Chapter 2

Theoretical Foundations

Even the longest journey
begins with the first step.
(Chinese Proverb)

The previous chapter provides us with a glimpse of the subject that we are going to study, that is, efficient algorithms for computing Gröbner bases. In this chapter, we craft a roadmap that leads to such a goal. No thesis can be totally self-contained, and we do not expect this one to be an exception. In particular, we assume that the reader has some basic knowledge of basic algebra, but we do not think it is harmful if we recall some fundamental definitions. Also, readers familiar with this topic may want to skim this chapter for notation and terminology.

We begin this chapter with a short introduction to polynomial rings and their ideals. Then we bring the reader into the realm of the term orderings. Term orderings are essential to write polynomials in a well-defined way. Therefore, they are required to the implementation of polynomials on a computer. Once fixed a term ordering, their leading terms can be singled out, and used to construct leading term ideals and modules, which are conceptually simpler to handle. This chapter ends with a brief introduction to Hilbert functions, which is one of the concepts that is required to fully understand the proposed criterion. A Hilbert function is said to be a map from the nonnegative integers to themselves, which stores the dimensions corresponding to the homogeneous components of a graded module.

For a more detailed introduction to commutative algebra we refer the reader to Atiyah and MacDonald [1969]; Eisenbud [2008]. Books providing extra emphasis to the theory of Gröbner bases and their computational aspects are, for example, Kreuzer and Robbiano [2000, 2005]. The majority of the proofs given in this chapter are ei-

ther easy to deduce or can be found in any introductory book about commutative or computational algebra (including the aforementioned ones). We focus ourselves on the theory of Gröbner bases, thus proofs are only provided if they are short, beautiful, and give some deeper insight on the topics covered, otherwise, we provide only references.

The main notion to retain from this chapter is the one of a polynomial, which plays a fundamental role in this thesis. None of the statements presented in this chapter are original. The same can be said also about their proofs.

2.1 Polynomial Rings

In this section, we shall recall the notion of polynomials and polynomial rings. These are some of the fundamental objects of study in this thesis. Since polynomial rings are also rings, we will start from that definition.

Let $\mathrm{P} = R[x_1, \ldots, x_n]$ be a polynomial ring over a ring R that consists of the multivariate polynomials of the form:

$$\sum c_{\alpha_1,\ldots,\alpha_n} x_1^{\alpha_1} \cdots x_n^{\alpha_n}$$

where $c_{\alpha_1,\ldots,\alpha_n} \in R$ and α_i are non-negative integers. The ring elements $c_{\alpha_1,\ldots,\alpha_n}$ are the *coefficients* of f. All powers products $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ are assumed to be commutative with other elements of the ring: $a_i x^i = x^i a_i$. Since elements of the type $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ are used frequently throughout this thesis then it is worth giving them a proper name.

Definition 2.1. A *term* (or a *power product*) is a product of n indeterminates, each raised to the power of a non-negative integer. It can be written concisely as

$$x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \tag{2.1}$$

where α is the coordinate vector $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}_0^n$.

Definition 2.2. The *total degree* of a term x^α is simply defined as $\deg(x^\alpha) = |\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n$. The degree of x^α in any indeterminate x_i is $\deg_{x_i}(x^\alpha) = \alpha_i$.

We label the set of all terms involving the indeterminates $x_1, \ldots, x_n$ by the expression $\mathbb{T}^n$ or $\mathbb{T}^n(x_1, x_2, \ldots, x_n)$. Also, we say that a term $p = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \in \mathbb{T}^n$ is *divisible* by $q = x_1^{\beta_1} \cdots x_n^{\beta_n} \in \mathbb{T}^n$, if

$$\Big(\forall\, 1 \leq i \leq n\Big)\ \Big[\beta_i \leq \alpha_i\Big].$$

Similarly to the integer case, it is possible to define least common multiples and greatest common divisors in a factorial domain.

Definition 2.3. Let $\mathrm{P} = K[x_1, \ldots, x_n]$ be a polynomial ring over a field K. Let $t_1 = \prod_{i=1}^{n} x_i^{\alpha_i} \in \mathbb{T}^n$ and $t_2 = \prod_{i=1}^{n} x_i^{\beta_i} \in \mathbb{T}^n$ be two terms. Then we define

$$\mathsf{lcm}(t_1, t_2) = \prod_{i=1}^{n} x_i^{\max\{\alpha_i, \beta_i\}}$$

as the *least common multiple* of t_1 and t_2, and

$$\gcd(t_1, t_2) = \prod_{i=1}^{n} x_i^{\min\{\alpha_i, \beta_i\}}$$

as the *greatest common divisor* of t_1 and t_2.

We say that t_1 and t_2 are *co-prime* or relatively prime if $\gcd(t_1, t_2) = 1$.

Definition 2.4. The *support* of a polynomial f is defined as the set $\mathrm{Supp}(f) = \{x^\alpha \in \mathbb{T}^n \mid \mathrm{coef}(c_\alpha x^\alpha) \neq 0\}$. A polynomial f is said to be *homogeneous of degree* d if all terms have the same total degree d.

Example. *Let $f = 3xy^2 + x^2 + 1$ be a polynomial in P. Then f has degree* $\deg(f) = 3$ *and a support set Supp$(f) = \{xy^2, x^2, 1\}$.*

Another important definition that is worth of our attention is the one called *monomial*.

Definition 2.5. A *monomial* is an element of P of the form $m = ct$, where $c \in R$ is its *coefficient* and $t \in \mathbb{T}^n$ is its *term*. The *total degree* of the monomial m is simply the total degree of t.

At the moment, we can rephrased the definition of a polynomial as a sum of a finite set of monomials, and its total degree is the maximum of the total degrees of the monomials in it; by convention, $\deg(0) = -\infty$. Moreover, two polynomials are equal only if their set of the non-zero monomials is the same.

As monomials are used to defined polynomials, these last ones can be used to generate polynomial ideals. The ideal that is generated by a system of polynomials $f_1, f_2, \ldots, f_n \in \mathrm{P}$ is simply the set of all combinations $\langle f_1, f_2, \ldots, f_n \rangle = \{h_1 f_1 + h_2 f_2 + \cdots + h_n f_n\}$, where $h_1, h_2, \ldots h_n \in \mathrm{P}$ are arbitrary polynomials.

2.2 Term Orderings

One of the questions that remains unanswered is: in how many different ways can we write a polynomial $p \in \mathrm{P}$? This question might not be relevant to many mathematicians but it fundamental to those who wish to implement and use polynomials in a computer.

Clearly, the terms in the support of a polynomial can be ordered by decreasing or increasing degree. However, that is not enough since different terms can have the same degree. Our next task is to equip polynomials with a certain additional property called a term ordering, which allow us to define them uniquely. This fundamental property is the key for the finiteness of most algorithms we shall encounter later.

Term Orderings on Polynomial Rings

In the univariate case the operation is straightforward because we can define $x^a \succ x^b$ as being true if and only if $a > b$. But for the multivariate case we need to find a new relation that is capable of providing such uniqueness and that is consistent with polynomial multiplication.

Definition 2.6. A *total ordering* is a set S and a relation on that set that satisfies the conditions for a partial ordering plus an additional condition known as the comparability condition. A relation $\leq$ is said to be a total ordering on a set S if the following properties hold.

- Reflexivity: $a \leq a$ for all $a \in S$.
- Antisymmetry: $a \leq b$ and $b \leq a$ implies $a = b$.
- Transitivity: $a \leq b$ and $b \leq c$ implies $a \leq c$.
- Comparability (trichotomy law): For any $a, b \in S$, either $a < b, a > b$ or $b = a$.

The first three axioms define only a *partial ordering*, whereas the addition of the trichotomy law defines a total ordering. Every totally ordered set that is finite is well ordered.

Definition 2.7. A *well-ordering* on a given set is a total ordering such that any nonempty subset has a smallest element w.r.t $\succ$.

Example. *The natural numbers are a well-ordered set.*

Definition 2.8. An *ordering* $\succ$ is an *admissible term ordering* if it satisfies three conditions:

1. An ordering $\succ$ is a total ordering on the set of all terms $\mathbb{T}^n$.
2. It is multiplicative; i.e., $x^a \succ x^b$ implies $x^{a+c} \succ x^{b+c}$ for all $a, b, c \in \mathbb{N}^n$.
3. An ordering $\succ$ is a well-ordering (every nonempty subset has a smallest element), namely, the constant term is the smallest.

Previously, we saw that a polynomial can be defined as a linear combination of terms. Now we learned that we can rearrange them unambiguously in ascending (or descending) order. We only have to compare the terms to establish their proper relative positions. That is, given any two terms x^α and x^β we have that only one of the three statements $x^\alpha > x^\beta, x^\alpha = x^\beta, x^\alpha < x^\beta$ hold. Thereby, the first condition in the above definition is satisfied. The second condition ensures that the relative ordering of terms in a polynomial remains unchanged after multiplied by a term. Lastly, the third condition states that every strictly decreasing sequence of terms eventually terminates. This property is often used to prove that algorithms must terminate. It exploits the fact that terms strictly decrease at each step of the algorithm. Nevertheless, the reader should bear in mind that different term orderings exist. Below we define the most relevant ones, and we leave their proof to the reader.

Definition 2.9. Let $\succ$ be a term ordering on P. Then $\succ$ is

1. A *global term ordering* if and only if $x_i \succ 1$ for $i = 1, \ldots, n$
2. A *local term ordering* if and only if $x_i \prec 1$ for $i = 1, \ldots, n$, and
3. A *mixed term ordering* if neither 1) nor 2) hold.

One of the simplest examples of a term ordering is the lexicographic ordering, which is frequently used to order words in the dictionary. There are several other term orderings of interest in computational algebra. Let us introduce a few among them. Let $\alpha, \beta \in \mathbb{N}^n$.

- The *lexicographic ordering* $\succ_{\text{Lex}}$ on $\mathbb{N}^n$ s.t.
  $\alpha \succ_{\text{Lex}} \beta$ if and only if the leftmost nonzero in $\alpha - \beta \in \mathbb{Z}^n$ is positive.
- The *degree lexicographic ordering*[1] $\succ_{\text{DegLex}}$ on $\mathbb{N}^n$ s.t.
  $\alpha \succ_{\text{DegLex}} \beta$ if and only if $|\alpha| > |\beta|$ or $|\alpha| = |\beta|$ and $\alpha \succ_{\text{Lex}} \beta$.
- The *degree reverse lexicographic ordering*[2] $\succ_{\text{DegRevLex}}$ on $\mathbb{N}^n$ s.t.
  $\alpha \succ_{\text{DegRevLex}} \beta \Leftrightarrow |\alpha| > |\beta|$ or $|\alpha| = |\beta|$ and the rightmost nonzero entry in $\alpha - \beta \in \mathbb{Z}^n$ is negative.

[1] Sort first by total degree, then lexicographic

[2] This ordering, somewhat non-intuitive, has some desirable computational properties

Example. *Consider the terms* $\alpha = x^3y^2z^8$ *and* $\beta = x^2y^9z^2$. *If the indeterminates are ordered as* $x \succ y \succ z$, *we have*

$$\alpha \succ_{\text{Lex}} \beta, \quad \alpha \succ_{\text{DevLex}} \beta, \quad \alpha \succ_{DegRevLex} \beta$$

An example of a local ordering is the *negative lexicographical ordering*. For such term ordering, we have that $\alpha \succ \beta$ if and only if the nonzero entry of lowest index in $\alpha - \beta$ is negative.

Term orderings have other useful properties. Once we fix an ordering, every polynomial f has a unique leading term. Moreover, that unique leading term is said to be the largest term in the polynomial (with respect to that term ordering) with a nonzero coefficient in the expansion of f.

Definition 2.10. Let $f = c_\alpha x^\alpha + \sum c_k x^{\beta_k} \in \mathrm{P}$ such that $x^\alpha \succ x^{\beta_k}$ and $c_\alpha \neq 0$. Then:

- $\mathsf{LC}_\succ(f) = c_\alpha$ is the *leading coefficient* of f. The polynomial f is said to be *monic* if $c_\alpha = 1$.
- $\mathsf{LT}_\succ(f) = x^\alpha$ is the *leading term* of f.
- $\mathsf{LM}_\succ(f) = \mathsf{LC}_\succ(f) \cdot \mathsf{LT}_\succ(f) = c_\alpha x^\alpha$ is the *leading monomial* of f.

We shall assume our polynomial rings to be always equipped with a well-ordering.

Term Orderings on Free P-modules

Up to now, we saw how to define polynomials uniquely in P. In this section, we extend the definition of term orderings on polynomial rings to the one on monomial modules. There are two reasons that can justify our interest in such orderings. First, we are interested in the computation of Gröbner bases of modules. Lastly, our algorithm work with elements of such modules. Hence, we start by revising Definition 2.1.

Definition 2.11. Let M be a submodule of the free P-module $F = \oplus_{j=1}^{r} P e_j$, with canonical basis elements e_i. We say that

1. $t \in F$ is a *term* in F, if for some i, the element t is of the form ue_i, such that u is a term in P.
2. M is a *monomial module*, if it is generated by terms t.
3. $m = ct$ is a *monomial* with coefficient in $c \in R$.
4. $index(t) = i$ is the *index of a term* $t = x^\alpha e_i$.

5. Computing the *degree of a term* $m = x^\alpha e_i$ can be reduced to the one of the term $x^\alpha \in \mathrm{P}$, as defined in 2.4:

$$\deg(m) := \deg(x^\alpha) = \sum_{i=1}^{n} \alpha_i.$$

   Clearly, $\deg(f) := \max\{\deg(t) \mid t \text{ is a term of } f\}$.

The set of all terms $\mathbb{T}^n\langle e_1, \ldots, e_r\rangle$ is obviously a disjoint union of r copies of $\mathbb{T}^n$, where $e_1, \ldots, e_r$ merely indicate the copy we are considering.

Definition 2.12. Let $x^\alpha e_i$ and $x^\beta e_j$ be two terms in $\mathbb{T}^n\langle e_1, \ldots, e_r\rangle$. We say that $x^\alpha e_i$ divides $x^\beta e_j$ if and only if

$$i = j \text{ and } x^\alpha \mid x^\beta.$$

for all $x^\alpha, x^\beta \in \mathbb{T}^n$. We use $x^\alpha e_i \mid x^\beta e_j$ as a shorthand notation for this operation.

We can write a element $f \in M$ as a sum of monomials

$$f = \sum_{i=1}^{r} \left(\sum_{\alpha \in \mathbb{N}^n}^{\text{finite}} c_\alpha x^\alpha \right) e_i$$

such that $c_\alpha \in R$ and $x^\alpha \in \mathbb{T}^n$. However, this representation is again unique only up to the order of the monomials. Thus, a term ordering on M is also required. Naturally, this is just a generalization of a term ordering on P that takes into account the canonical basis elements e_i.

Definition 2.13. Let $\succ_\sigma$ be a term ordering on P and $M = \oplus_{i=1}^r = \mathrm{P}e_i$ a free P-module of rank r with canonical basis elements e_i. A *module term ordering* $\succ$ is a total ordering on the set of all terms of M such that

- $x^\alpha e_j \succ x^\beta e_j \Rightarrow x^\gamma x^\alpha e_i \succ x^\gamma x^\beta e_j$ and
- $x^\alpha \succ_\sigma x^\beta \Rightarrow x^\alpha e_i \succ x^\beta e_i$.

  for all $\alpha, \beta, \gamma \in \mathbb{N}^n$ and $i, j \in \{1, \ldots, r\}$.

Clearly, a term ordering on the polynomial ring P can also be understood as a module term ordering on the module $\mathrm{P} \cong \mathrm{P}e_1$. Therefore, Definition 2.13 is just a generalization of the former definition of term ordering.

Example. *Suppose we fix a term ordering $\succ_\sigma$ on P that induces the (module) term ordering $\succ$ on M.*

- $\succ_{\text{POT}}$ *denotes an ordering which emphasizes the index of the canonical basis element:*

$$x^\alpha e_i \succ_{\text{POT}} x^\beta e_j \text{ implies that } i > j \text{ or } i = j \text{ and } x^\alpha \succ_\sigma x^\beta$$

- $\succ_{\text{TOP}}$ *denotes an ordering that gives priority to terms:*

$$x^\alpha e_i \succ_{\text{TOP}} x^\beta e_j \text{ implies that } x^\alpha \succ_\sigma x^\beta \text{ or } x^\alpha = x^\beta \text{ and } i > j.$$

Later we will see more examples of module orderings but for the moment we will concentrate our attention to the benefits of having module orderings. Similar to the polynomial case we can now identify and define special elements of $f \in M$.

Definition 2.14. Given a module ordering $\succ$ on M, every element $f \in M$ can be uniquely represented by $f = c_\alpha x^\alpha e_i + f'$, and for all nonzero terms $c_\beta x^\beta e_j$ of f' it holds that $x^\alpha e_i \succ x^\beta e_j$.

In analogy to Definition 2.10 we have the following definitions for modules.

Definition 2.15. Let f be a polynomial in a module M.

- the leading term of f is $\mathsf{LT}_\succ(f) = x^\alpha e_i$,
- the leading coefficient of f is $\mathsf{LC}_\succ(f) = c_\alpha$,
- the leading monomial of f is $\mathsf{LM}_\succ(f) = c_\alpha x^\alpha e_i$,

Likewise in the polynomial case, we shall always equip a module M with a module term ordering $\succ$. Hence, the use of the notation M implies also the use of a module term ordering on M.

Next we analyze an example that should help us to understand better the relationship between two module terms.

Example. *Let $P = R[x, y, z]$ and $M = P^2$. Then, the representation of f as the sum of $m_1 = -2x^2ye_1$, $m_2 = 4x^3yz^2e_1$, and $m_3 = z^4e_2$ is given by:*

- *If we pick $\succ_{\text{POT}}$ induced by $\succ_{\text{DegRevLex}}$, we get the following:*

$$f = (4x^3yz^2 - 2x^2y)e_1 + z^4e_2.$$

- *If instead we pick $\succ_{\text{TOP}}$ induced by $\succ_{\text{DegRevLex}}$ then the sequence of the monomials obtained is:*

$$f = 4x^3yz^2e_1 + z^4e_2 - 2x^2ye_1,$$

With this, we conclude our introduction to term orderings on polynomial rings and free modules. For more details see [Kreuzer and Robbiano, 2000].

2.3 Monomial Ideals and Dickson's Lemma

The aim of this section is to show that monomial ideals are finitely generated, and that follows easily from the Dickson's Lemma. Then, we provide a generalization of that theorem to the monomial modules. The importance of such result is self-evident in the next chapters. However, it should be already clear that statements about "finiteness" are extremely valuable from a computational point of view. Let $\mathcal{I}$ be an ideal in the polynomial ring $\mathrm{P} = R[x_1, \ldots, x_n]$.

Definition 2.16. An ideal of P is called a *monomial ideal* if it can be generated by a set of terms.

The following lemma allows us to characterize all the terms that lie in a given monomial ideal.

Lemma 2.17. *Let $\mathcal{I} = \langle x^\alpha \mid \alpha \in X \rangle \subseteq \mathbb{T}^n$ be a monomial ideal. Then a term $x^\beta \in \mathbb{T}^n$ is in $\mathcal{I}$ if and only if there is some $\alpha \in X$ such that $x^\alpha \mid x^\beta$.*

Proof. See, for example, Cox et al. [2007]. □

Observe that the set $\alpha + \mathbb{N}^n = \{\alpha + \beta \mid \beta \in \mathbb{N}^n\}$ consists of the exponents of all terms divisible by x^α. Hence, the terms in a given monomial ideal can be visualized as a union of positive and integer coordinate points. For example, in Figure 2.1, the integer coordinate points inside filled area represent all the terms in the ideal $\mathcal{I} = (x_1^5, x_1^3 x_2, x_1 x_2^2, x_2^4) \subseteq R[x_1, x_2]$.

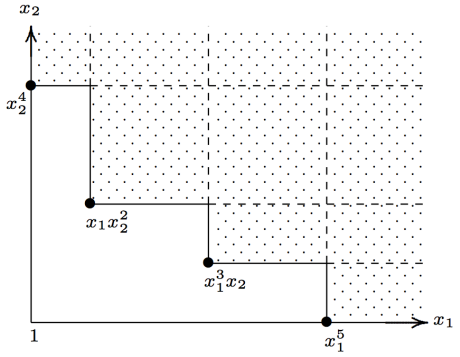

Figure 2.1: Representation of a monomial ideal with two indeterminates

A polynomial f is in a monomial ideal $\mathcal{I} = \langle x^\alpha \mid \alpha \in X \rangle$ if and only if each term of f is divisible by one of the given generators x^α. It follows that a monomial ideal is uniquely determined by its terms; that is,

Theorem 2.18. *Two monomial ideals are the same if and only if they contain the same collection of terms.*

Proof. For more details and proofs, see for example, [Cox et al., 2007]. □

The main result in this section is given by the Dickson's lemma, which states that each monomial ideal is finitely generated.

Theorem 2.19 (Dickson's Lemma)**.** *If $\mathcal{I} = \langle x^\alpha \mid \alpha \in X \subseteq \mathbb{N}^n \rangle$, then there is a finite subset $Y \subset X$ for which $\mathcal{I} = \langle x^\alpha \mid \alpha \in Y \rangle$. In particular, for every polynomial ring P, the monomial ideal $\mathcal{I}$ has a finite basis.*

Proof. Let $\mathcal{I} = (t_1, t_2, \dots) \subseteq \mathbb{T}^n$. Let log:$\mathbb{T}^n \to \mathbb{N}^n$ be the map given by $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \to (\alpha_1, \dots, \alpha_n)$. Such map is clearly an isomorphism of monoids. Hence, the monomial ideal $(\log(t_1), \log(t_2), \dots) \subseteq \mathbb{N}^n$ is finitely generated by the Noetherianity condition: there is no infinite properly ascending chain of ideals of $\mathbb{N}^n$. Thus there exists a number $i > 0$ such that this monomial ideal is generated by $\log(t_1), \dots, \log(t_i)$ where $t_i \in \mathbb{T}^n$. Therefore, the set of terms $\{t_1, \dots, t_i\}$ generates the monomial ideal $(t_1, t_2, \dots) \subseteq \mathbb{T}^n$. □

The next theorem give us a generalization of the Dickson's Lemma for monomial modules.

Theorem 2.20 (Structure Theorem for Monomial Modules)**.** *Let $M \subseteq P^r$ be a monomial module.*

1. *The module M is finitely generated. That is, $M = \langle t_1 e_{\alpha_1}, \dots, t_s e_{\alpha_s} \rangle$, where s is finite, $t_1, \dots, t_s \in \mathbb{T}^n$ and $1 \leq \alpha_i \leq r$.*
2. *There are monomial ideals $\mathcal{I}_1, \dots, \mathcal{I}_r \subseteq P$ such that M is of the form $M \cong \oplus_{i=1}^r \mathcal{I}_i e_i$.*

Proof. Assume $B \subseteq \mathbb{T}^n \langle e_1, \dots, e_r \rangle$ to be a system of generators of the monomial module M. Let $B_i = \{t \mid te_i \in B\} \subseteq \mathbb{T}^n$ be a set of terms with $1 \leq i \leq r$. By Dickson's Lemma, the monomial ideal $\mathcal{I}_i = (B_i)$ has a finite systems of generators $S_i \subseteq B_i$. Clearly, the P-module M is then generated by $S_1 e_1 \cup \cdots \cup S_r e_r \subseteq \mathbb{T}^n \langle e_1, \dots, e_r \rangle$. This proves the finiteness and also the claim that $M = \sum_{i=1}^r \mathcal{I}_i e_i$. The fact that this sum is direct follows from $M \subseteq \oplus_{i=1}^r P e_i$. □

2.4 Leading Term Ideals and Modules

In the previous section, we have seen that once a term ordering is chosen, the terms in the support of a polynomial can be uniquely sorted. Therefore, the polynomial can be uniquely represented as a sum of monomials.

The aim of this section is to recall a few more properties concerning ideals and modules, such as the relation they have with their leading term ideals and modules. These properties are useful to answer questions like: let $\mathcal{I}$ be an ideal in a polynomial ring $\mathrm{P} = K[x_1, \ldots, x_n]$ over a field K. Obviously, the residue class ring $\mathrm{P}/\mathcal{I}$ can be viewed as a K-vector space. So, is it possible to exhibit an explicit basis?

We shall see that with the help of leading terms, the Macaulay's Basis Theorem yields a noteworthy answer to this question. However, this theorem is based on the assumption that the module ordering is a term ordering and the base ring is a field. The assumption underlines the theoretical importance of term orderings.

Definition 2.21. Let $M \subseteq \mathrm{P}^r$ be a P-submodule and $\succ$ fixed term ordering.

1. The module $\mathsf{LT}_\succ(M) = \langle \mathsf{LT}_\succ(m) \mid m \in M \backslash \{0\} \rangle$ is said to be the *leading term module* of M with respect to $\succ$.
2. If $M \subseteq \mathrm{P}$ then we say that $\mathsf{LT}_\succ(M) \subseteq \mathrm{P}$ is the *leading term ideal* generated by the leading terms of all the elements in M with respect to $\succ$. A term x^α is called *standard* if it does not belong to $\mathsf{LT}_\succ(M)$.
3. The set $\mathsf{LT}_\succ\{M\} = \{\mathsf{LT}_\succ(m) \mid m \in M \backslash \{0\}\}$ is called *monomodule*.

Example. *Assume that $M = \langle 0 \rangle$ is a module generated by the element zero. Then we have $\mathsf{LT}_\succ(M) = \langle 0 \rangle$ and $\mathsf{LT}_\succ\{M\} = \emptyset$.*

Assume that $m_1, \ldots, m_s \in \mathrm{P}^r \backslash \{0\}$ are monomials and that $M = \langle m_1, \ldots, m_s \rangle \subseteq \mathrm{P}^r$ is the submodule generated by them. Then, we have that the inclusion given by $\langle \mathsf{LT}_\succ(m_1), \ldots, \mathsf{LT}_\succ(m_s) \rangle \subseteq \mathsf{LT}_\succ(M)$ holds. Now we introduce the most important result of this section.

Theorem 2.22 (Macaulay's Basis Theorem)**.** *Assume $P = K[x_1, \ldots, x_n]$ to be a polynomial ring over K. Let $M \subseteq P^r$ be a P-submodule, $\succ$ a term ordering on $\mathbb{T}^n\langle e_1, \ldots, e_r \rangle$, and B the set of all terms in $\mathbb{T}^n\langle e_1, \ldots, e_r \rangle \backslash \mathsf{LT}_\succ\{M\}$. Then the residue classes of the elements of B form a basis of the K-vector space P^r/M.*

Proof. See, for example, Section 1.5 in Kreuzer and Robbiano [2000]. □

This theorem shows how to compute effectively P^r/M. First we need to compute $\mathsf{LT}_\succ(M)$ for a given term ordering $\succ$, and then we represent each element as a unique and finite linear combination of the residue classes of the elements of $B = \mathbb{T}^n\langle e_1, \ldots, e_r\rangle \setminus \mathsf{LT}_\succ\{M\}$. Unfortunately, we have not seen yet how to compute efficiently $\mathsf{LT}_\succ(M)$. That is one of the subjects of the next chapter.

2.5 Gradings

The main purpose of this section is to recall the definition of gradings. Then, we shall see that homogeneous ideals and graded submodules can be characterized by the property that they have homogeneous sets of generators. Last, but not least, we show that it is also possible to represent homogeneous elements in terms of those generators using homogeneous coefficients of complementary degree.

We recall again that in this thesis monoids and rings are expected to be commutative. Let $(\Gamma, +)$ be a monoid and M a R-module.

Definition 2.23. The ring R is said to be a Γ-*graded ring* if there exist abelian subgroups R_γ, and

1. $R = \oplus_{\gamma\in\Gamma} R_\gamma$, and
2. for all $\gamma, \gamma' \in \Gamma$ it holds that $R_\gamma R_{\gamma'} \subseteq R_{\gamma+\gamma'}$.

We say that elements $r \in R_\gamma$ are *homogeneous of degree* γ, that is, $\deg(r) = \gamma$. Additionally, the element 0 is a homogeneous element of R of every degree. Lastly, the decomposition of every element into its homogeneous components is unique. That property is a consequence of the direct sum in Definition 2.23.

Example. *Let $P = R[x_1, \ldots, x_n]$ be a polynomial ring in n indeterminates over R. Let*

$$P_d = \{f \in P \mid \deg(t) = d \text{ for all } t \in \mathit{Supp}(f)\} \text{ for } \mathrm{d} \geq 0$$

The set P_d establishes a relation between P and a $\mathbb{N}$-graded ring. If $\deg(x_1) = \cdots = \deg(x_n) = 1$ then this grading is called the standard grading of P. Again, the elements of P_d are denoted as homogeneous polynomials of degree d.

A natural way to extend Definition 2.23 to R-modules is to use again the monoid Γ as the set of possible degrees. However doing so is not sufficiently general. The following definition underlines for now how to proceed.

Definition 2.24. Let R be a Γ-graded ring and $(\Omega, *)$ a Γ-monomodule. We say that M is a Ω-*graded R-module* if there exist abelian subgroups M_ω such that

1. $M = \oplus_{\omega\in\Omega} M_\omega$, and
2. for all $\gamma \in \Gamma, \omega \in \Omega$ it holds that $R_\gamma M_\omega \subseteq M_{\gamma*\omega}$.

Additionally, we say that a R-submodule N of M is a Ω-*graded* R-*submodule* of M if $N = \oplus_{\omega\in\Omega}(N \cap M_\omega)$, and that a Γ-graded submodule of R is a Γ-homogeneous ideal of R. For the sake of simplicity we call it *homogeneous ideal* of R if Γ is obvious from the context.

There are other techniques that can be used to define Ω-graded R-modules, called shifting degrees.

Example. *Let $\gamma \in \Gamma$ be a fixed element such that the multiplication map $\mu_\gamma : \Omega \longrightarrow \Omega$ defined by $s \longrightarrow \gamma * s$ is injective. For instance, if the left cancellation law holds in Ω, this assumption is satisfied for all $\gamma \in \Gamma$. Then we define $M(\gamma)_s = M_{\gamma*s}$ for every $s \in \Omega$, and we let $M(\gamma) = \oplus_{s\in\Omega} M(\gamma)_s$. In this way, we get a Ω-graded R-module $M(\gamma)$. We call it the module obtained by shifting degrees by γ. If the map $\mu_\gamma : \Omega \longrightarrow \Omega$ is bijective, the set underlying $M(\gamma)$ agrees with M. For more information concerning homomorphisms between these objects, see Kreuzer and Robbiano [2000].*

The following proposition and its corollary are very handy, especially for practical purposes. They allow us to quickly prove that some submodules are Ω-graded by exhibiting a homogeneous system of generators, and at the same time it give us a convenient representation of arbitrary homogeneous elements in terms of those homogeneous generators.

Proposition 2.25. *Let $N \subseteq M$ be a R-submodule and $N_\omega = N \cap M_\omega$ for all $\omega \in \Omega$. Then the following three conditions are equivalent.*

1. *$N = \oplus_{\omega\in\Omega} N_\omega$*
2. *Let $n = \sum_{\omega\in\Omega} n_\omega$ be the decomposition of $n \in N$ into its homogeneous components, then $n_\omega \in N$ for all $\omega \in \Omega$.*
3. *There is a system of generators of N which consists only of homogeneous elements.*

Proof. See, for example, Section 1.7 in Kreuzer and Robbiano [2000]. □

Corollary 2.26. *Let $N \subseteq M$ be a Ω-graded R-submodule and $\{n_\gamma \mid \gamma \in \Gamma\}$ be a set of homogeneous generators of N. Suppose that the right-cancellation law holds in Ω and that $\omega \in \Omega$. Then we can say that every element $n \in N_\omega$ has a representation $n = \sum_{\gamma\in\Gamma} r_\gamma n_\gamma$ such that $r_\gamma \in R$ are homogeneous elements and $\deg(r_\gamma) * \deg(n_\gamma) = \omega$ for every $\gamma \in \Gamma$.*

Proof. See, for example, Section 1.7 in Kreuzer and Robbiano [2000]. □

We conclude this section with a definition of homogenization and dehomogenization of polynomials.

Definition 2.27. Given any polynomial $p \in R[x_1, \ldots, x_n]$ and an extra indeterminate x_0, we say that,

$$p^h = x_0^{\deg(p)} p\left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\right)$$

denotes the homogenization of p with respect to x_0. Hence, p^h is homogeneous polynomial of degree $\deg(p)$.

Definition 2.28. Conversely, for every homogeneous polynomial $q \in R[x_0, \ldots, x_n]$ there exists a dehomogenization with respect to x_0 defined by

$$q^{deh} = q(1, x_1, \ldots, x_n) \in R[x_1, \ldots, x_n].$$

2.6 Hilbert Functions and Hilbert Series

In this section we recall the definition of Hilbert functions and Hilbert Series. These are two concepts that have an essential role throughout the next chapters. A Hilbert function is said to be a map from the nonnegative integers to themselves, which stores the dimensions corresponding to the homogeneous components of a graded module. In many situations of interest, the Hilbert function attached to a module agrees for sufficiently large inputs with a polynomial, called a Hilbert polynomial. Thus an infinite amount of information can be encoded and stored in a finite object. Without a doubt, the coefficients and the degree of a Hilbert polynomial are important invariants that can reveal some of the properties of the module.

Let $P = K[x_0, \ldots, x_n]$ be a graded polynomial ring, where each x_i is a homogeneous component of degree one.

Definition 2.29. The *Hilbert function* of the finitely generated P-module M is defined by the following map

$$\begin{array}{rccc} \mathsf{HF}(M,d) & : & \mathbb{Z} & \longrightarrow & \mathbb{Z} \\ & & v & \mapsto & \dim(M_d) \end{array}$$

where M_d is the degree d part of M.

The next proposition shows that the Hilbert function becomes a polynomial in large enough degree, called the Hilbert polynomial.

Proposition 2.30. *The Hilbert polynomial $q(d)$ is the unique polynomial in d such that the equivalence $q(d) = \mathsf{HF}(M, d)$ holds for all $d >> 0$. Let S be the support of M. Then, the Hilbert polynomial $q(d)$ has the following form*

$$q(d) = \frac{i}{r!}d^r + \cdots$$

where i is the degree of S and r is its dimension.

Proof. See, for example, Section 5.1 in Kreuzer and Robbiano [2009]. □

A compact representation of the dimensions corresponding to the homogeneous components of M is given by introducing the Hilbert Series. This representation of the Hilbert function is desirable if we want to manipulate it on a computer. From now we shall use "computing the Hilbert function" and "computing the Hilbert Series" interchangeably.

Definition 2.31. Let $M = \oplus_{n\geq 0} M_n$ be a finitely generated P-module M. The *Hilbert Series* of M, is defined by the generating function

$$\mathsf{HS}(M, t) = \sum_{d=-\infty}^{\infty} \mathsf{HF}(M, d)t^d = \frac{g(t)}{(1-t)^{n+1}}$$

where $g(t) \in \mathbb{Z}[t, t^{-1}]$ is a Laurent polynomial over $\mathbb{Z}$.

At this point, it is also our greatest interest to analyze how the Hilbert Series of $\mathcal{I}$ relates to the Hilbert Series of quotient rings $\mathrm{P}/\mathcal{I}$, since we also want to compute them. The next theorem claims that, for a polynomial ring P and a homogeneous ideal $\mathcal{I}$, the Hilbert Series of $\mathrm{P}/\mathcal{I}$ has always the form of a rational function.

Theorem 2.32 (Hilbert, Serre)**.** *Let $P = K[x_0, \ldots, x_n]$ be a graded standard polynomial ring and $\mathcal{I}$ a graded ideal in P. Then, the Hilbert Series of $P/\mathcal{I}$ can be expressed as*

$$\mathsf{HS}(P/\mathcal{I}, t) = \frac{g(t)}{(1-t)^{n+1}}$$

where $g(t)$ is a polynomial in t with integer coefficients. Moreover, the following relation holds,

$$\mathsf{HS}(P/\mathcal{I}, t) = \mathsf{HS}(P, t) - \mathsf{HS}(\mathcal{I}, t)$$

Proof. See, for example, Section 5.2 in Kreuzer and Robbiano [2005]. □

A practical approach to compute the Hilbert Series of $\mathrm{P}/\mathcal{I}$ is to compute the Gröbner basis of $\mathcal{I}$, and then compute the Hilbert Series of $\mathrm{P}/\mathsf{LT}_{\succ}(\mathcal{I})$ using elimination of variables. In Section 4.4, we shall see that $\mathrm{P}/\mathcal{I}$ and $\mathrm{P}/\mathsf{LT}_{\succ}(\mathcal{I})$ have the same Hilbert function. But first, we shall introduce the concept of Gröbner basis.

Chapter 3

Gröbner Bases Theory

Any fool can know.
The point is to understand.
(Albert Einstein)

This aim of this chapter is to introduce the reader to the theory of Gröbner bases. First, we recall the definition of a Gröbner basis as well as all the details on how to construct them. Then, in Section 3.5, we provide some remarks concerning the computational complexity of the algorithm. We conclude with a brief review of existent optimizations to the computation of a Gröbner basis, and with an introduction to signature-driven strategies.

3.1 Introduction to the Theory of Gröbner Bases

A Gröbner basis is a special basis of a polynomial ideal over a field, which has certain attractive computational properties. As it turns out, many computational problems involving ideals can be easily solved once we compute them. For example, let us assume that we want to compute $\mathsf{LT}_{\succ}(\mathcal{I})$ for a given ideal $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$. It is worth mentioning that a direct computation of $\mathsf{LT}_{\succ}(\mathcal{I})$ is impossible because it requires the computation of the leading term of each polynomial in the ideal $\mathcal{I}$. As an alternative, we might want to consider the monomial ideal generated by the leading terms of the generators,

$$\langle \mathsf{LT}_{\succ}(f_1), \ldots, \mathsf{LT}_{\succ}(f_s) \rangle \subseteq \mathsf{LT}_{\succ}(\mathcal{I})$$

In general, these two monomial ideals are distinct yet the equality can sometimes occur. The example below illustrates a case where the two monomial ideals do not coincide.

Example. *Consider the ideal* $\mathcal{I} = \langle f_1, f_2 \rangle = \langle 3x^2+7xy^2, 3xy+7y^3-1 \rangle$ *and a fixed term ordering* $\succ$. *On the one on hand, we have that* $yf_1 - xf_2 = x \in \langle f_1, f_2 \rangle$ *and* $\mathsf{LT}_\succ(x) = x \in \langle \mathsf{LT}_\succ(\mathcal{I}) \rangle$. *However, on the other hand, we have that* $x \notin \langle \mathsf{LT}_\succ(f_1), \mathsf{LT}_\succ(f_2) \rangle = \langle 3x^2, 3xy \rangle$.

At this point, one might ask if it is possible to produce a set of generators for which these two ideals are the same. The answer to this question is exactly the notion of a Gröbner basis.

Definition 3.1. Given an ideal $\mathcal{I} \subseteq \mathrm{P}$ and a fixed term ordering $\succ$. A finite set of polynomials $\mathcal{G} \subseteq \mathcal{I} \backslash \{0\}$ is a *Gröbner basis* of $\mathcal{I} \backslash \{0\}$ if and only if the leading term of every element of $\mathcal{I}$ is divisible by one term in the set $\mathcal{G}$,

$$\langle \mathsf{LT}_\succ(g) \mid g \in \mathcal{G} \rangle = \mathsf{LT}_\succ(\mathcal{I})$$

Unfortunately, this definition does not tell us how to construct a Gröbner basis for the ideal generated by a given $\mathcal{G}$. But, in the next section, we shall see other characterizations that give a systematic way of doing it.

3.2 Buchberger's Algorithm

Buchberger introduced in his PhD dissertation [Buchberger, 1965] an algorithm to systematically compute a Gröbner basis $\mathcal{G}$ for any ideal $\mathcal{I}$. From his work, it turns out that the concept of a Gröbner bases is intimately related to the idea of *reduction* and *normal form*.

Definition 3.2. Let $\mathcal{G} = (g_1, \ldots, g_m) \in P^m$ be an ordered tuple of non-zero polynomials, and f an additional polynomial in P. The algorithm for dividing f by $(g_1, \ldots, g_m)$, denoted as Division Algorithm, is an algorithm that writes f as $f = e_1 g_1 + \cdots + e_m g_m + r$, where r has the property that none of its monomials is divisible by the leading term of any of the g_i.

The implementation of the division algorithm is rather simple and it always terminates because there is a term ordering. Hence, we know that the degree of f shall drop after each reduction step. The algorithm is the following.

Input:
An ordered tuple of non-zero polynomials $(g_1, \ldots, g_m) \in P^m$.
A polynomial $f \in P$ and a term ordering $\succ$.
Output:
A polynomial r not divisible by the leading term of any of the g_i.

```
1  Set e_1 = e_2 = ··· = e_m = r = 0;
2  while f ≠ 0 do
3      foreach i = 1, ..., m do
4          if LT_≻(g_i) divides LT_≻(f) then
5              f = f − LT_≻(f) / LT_≻(g_i) ;
6              e_i = e_i + LT_≻(f) / LT_≻(g_i);
7              Go to step 2;
8          end
9      end
10     r = r + LT_≻(f);
11     f = f − LT_≻(f);
12 end
13 return r
```

Algorithm 1: Division Algorithm

Definition 3.3. Let r be the polynomial, as defined in Definition 3.2, and obtained by reducing f repeatedly by polynomials of $\mathcal{G}$, until it is *irreducible*. Then f is called the *normal form* of f with respect to $\mathcal{G}$, and it is denoted as $r = \mathsf{NR}_{\succ,\mathcal{G}}(f)$.

The next theorem establishes a connection between the previous definitions and the definition of a Gröbner basis.

Theorem 3.4 (Buchberger)**.** *$\mathcal{G}$ is a Gröbner basis of an ideal $\mathcal{I}$ if and only if the following holds*

$$f \in \mathcal{I} \iff \mathsf{NR}_{\succ,\mathcal{G}}(f) = 0$$

Proof. Let f be an arbitrary polynomial in the ideal $\mathcal{I}$. Then the division of f by $\mathcal{G}$ yields the following form

$$f = e_1 g_1 + \cdots + e_m g_m + r$$

Hence $r \in \mathcal{I}$ because $f - r \in \mathcal{I}$. Assume that $r \neq 0$. Then we can find a k such that $\mathsf{LT}_\succ(g_k)$ divides $\mathsf{LT}_\succ(r)$, since $\mathcal{G}$ is a Gröbner basis. This is a contradiction to the fact that r is reduced with respect to $\mathcal{G}$. Thus, r must be equal to zero. □

Another characterization of a Gröbner bases is intimately related to the concept of S-polynomials. The computation of the S-polynomial of two polynomials g_1 and g_2 consists in the multiplication of the two polynomials by some monomial factors, such that the leading term of both polynomials becomes equal. Then, by subsequent subtraction, we have that this least common multiple term is canceled.

Definition 3.5. Let $g_1, g_2 \in \mathrm{P}$. The *S-polynomial* of g_1 and g_2 is defined to be

$$S_{1,2} = \frac{p}{\mathsf{LM}_{\succ}(g_1)} g_1 - \frac{p}{\mathsf{LM}_{\succ}(g_2)} g_2$$

where p is the least common multiple

$$p = \mathsf{lcm}(\mathsf{LT}_{\succ}(g_1), \mathsf{LT}_{\succ}(g_2))$$

Example. *Consider the ideal* $\mathcal{I} = \langle g_1, g_2 \rangle = \langle xy - z^2, y^2 - z^2 \rangle \subseteq \mathbb{Q}[x, y, z]$ *and the term ordering* $\succ_{\mathrm{DegRevLex}}$

$$S_{2,1} = xg_2 - yg_1 = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2$$

The result of this computation is a new element $\mathbf{g}_3 = xz^2 - yz^2$.

The intuition behind the notion of S-polynomials is the following: the least common multiple of the "leading terms" of g_1 and g_2 is the first possible polynomial that allows essentially two different reductions modulo g_1, g_2. Then the Theorem 3.6 gives shape to a necessary condition that is sufficient to guarantee that a set of polynomials is a Gröbner basis of some ideal $\mathcal{I}$. That is, we only need compute the finitely many S-polynomials of a given finite set of polynomials, to master the infinitely many polynomials that allow two or more essentially different reductions.

Theorem 3.6 (Buchberger's Criterion)**.** *Let* $\mathcal{I} = \langle g_1, \ldots, g_m \rangle \subseteq P$, $\mathcal{G}$ *a set of polynomials* $\{g_1, \ldots, g_m\}$, *and* $\mathbb{B} = \{(i, j) \mid 1 \leq i < j \leq m, e_i = e_j\}$, *where* $\mathsf{LM}_{\succ}(g_i) = c_i t_i e_i$. *Then the following conditions are equivalent:*

1. *The set* $\mathcal{G}$ *is a Gröbner basis of* $\mathcal{I}$ *with respect to* $\prec$.
2. *For all pairs* $(i, j) \in \mathbb{B}$, *we have* $\mathrm{NR}_{\prec,\mathcal{G}}(\mathrm{S_{i,j}}) = 0$.

Proof. See, for example, Section 2.5 in Kreuzer and Robbiano [2009]. □

Theorem 3.4 and 3.6 are clearly better characterizations of a Gröbner basis, if compared to our first definition. In the first definition, we have that every polynomial must have a unique normal form. Although, the Theorem 3.4 has similar requirements, they are restricted to polynomials in the ideal. A further improvement is given in Theorem 3.6, where we only have to consider a finite set of polynomials. Hence, it gives an effective criterion to test if a given $\mathcal{G}$ is indeed a Gröbner basis: given $\mathcal{G}$, for each pair $f, g \in \mathcal{G}$, we compute $\mathsf{NR}(S_{f,g})$ and see if it is equal to zero. If this is zero for all f, g then we know $\mathcal{G}$ is a Gröbner basis. In other words, it tell us how to recreate Buchberger's algorithm.

Input:
A set F of generators $\{f_1, \ldots, f_m\}$ of the ideal $\mathcal{I}$.
A term ordering $\succ$.
Output:
A Gröbner basis of $\mathcal{I}$.

1 Set $\mathcal{G}$ equal to F;
2 Let $\mathbb{B}$ be the set of pairs $\{(i,j) \mid g_i, g_j \in \mathcal{G}, i > j\}$;
3 **while** $\mathbb{B} \neq \emptyset$ **do**
4 Choose a pair $(i,j) \in \mathbb{B}$;
5 Set $\mathbb{B} := \mathbb{B} \setminus \{(i,j)\}$;
6 Compute S_{ij} and let $h = \mathrm{NR}_{\succ,\mathcal{G}}(\mathrm{S_{ij}})$;
7 **if** $h = 0$ **then**
8 Go to line 3;
9 **end**
10 **if** $h \neq 0$ **then**
11 Add h to $\mathcal{G}$;
12 Build new all critical pairs containing h and add them to $\mathbb{B}$;
13 Go to step 3;
14 **end**
15 **end**
16 **return** $\mathcal{G}$

Algorithm 2: Buchberger's Algorithm

Example. *Consider* $\mathcal{I} = \langle f_1, f_2 \rangle$ *such that* $f_1 = x^2 + 2xy^2$ *and* $f_2 = xy + 2y^3 - 1$ *are two polynomials in* $P = K[x, y]$. *Let* $\succ$ *be the Lexicographic ordering in* P.

1. *Set* $\mathcal{G}$ *equal to* $\{f_1, f_2\}$.
2. *Let* $\mathbb{B}$ *be the set of pairs* $\{(1, 2)\}$.
3. *Choose* $(1, 2) \in \mathbb{B}$ *and set* $\mathbb{B} = \emptyset$.
4. *Compute the S-polynomial for the critical pair (1, 2):* $S_{1,2} = x$.
5. *Reduce the S-polynomial* $S_{1,2}$ *using the set of polynomials* $\mathcal{G}$. *In this particular case,* $S_{1,2}$ *is already irreducible:* $\mathsf{NR}_{\succ,\mathcal{G}}(x) = x$.
6. *It is true that* $x \neq 0$, *so we must add* $f_3 = x$ *to our generating set* $\mathcal{G}$ *and update* $\mathbb{B}$; *we get* $\mathcal{G} = \{f_1, f_2, f_3\}$ *and* $\mathbb{B} = \{(1, 3), (2, 3)\}$.
7. *Pick another pair of* $\mathbb{B}$, *say* $(2, 3)$, *and compute* $S_{2,3} = 2y^3 - 1$. *Insert* $S_{2,3}$ *into the* $\mathcal{G}$ *because* $S_{2,3} = f_4 = 2y^3 + 1$ *is already completely reduced modulo* $\mathcal{G}$. *Generate new critical pairs. As a result we obtain* $\mathcal{G} = \{f_1, f_2, f_3, f_4\}$ *and* $\mathbb{B} = \{(1, 3), (1, 4), (2, 4), (3, 4)\}$. *The computation of the remaining pairs in* $\mathbb{B}$ *re-*

sults in all of them reducing to 0 *modulo* $\mathcal{G} = \{f_1, f_2, f_3, f_4\}$:

$$\begin{aligned}
S_{1,3} &= 2xy^2 = 2y^2 f_3 \text{ so } \mathsf{NR}_{\succ,\mathcal{G}}(\mathrm{S}_{1,3}) = 0;\\
S_{1,4} &= \frac{1}{2}x^2 + 2xy^5 = \frac{1}{2}f_1 + xy^2 f_4 \text{ so } \mathsf{NR}_{\succ,\mathcal{G}}(\mathrm{S}_{1,4}) = 0;\\
S_{2,4} &= \frac{1}{2}x + 2y^5 - y^2 = \frac{1}{2}f_3 + y^2 f_4 \text{ so } \mathsf{NR}_{\succ,\mathcal{G}}(\mathrm{S}_{2,4}) = 0;\\
S_{3,4} &= \frac{1}{2}x = \frac{1}{2}f_3 \text{ so } \mathsf{NR}_{\succ,\mathcal{G}}(\mathrm{S}_{3,4}) = 0.
\end{aligned}$$

Hence, the computed Gröbner basis of $\mathcal{I}$ *is given by:*

$$\mathcal{G} = \{x^2 + 2xy^2, xy + 2y^3 - 1, x, 2y^3 - 1\}.$$

The above algorithm computes Gröbner bases that are usually bigger than necessary. In the above example we could for example eliminate f_1 and f_2, since their leading terms are both multiples of the leading term of f_3. Hence, we would get

$$\mathcal{G} = \{f_3, f_4\} = \{x, 2y^3 - 1\}.$$

Although, Buchberger's algorithm is deceptively simple, its theoretical cost is extremely high. In fact, no general upper bound for the running time is known. However, Dubé [1990] proved an upper bound for the maximum degree of the Gröbner basis elements which depends doubly exponentially on the number of variables. Moreover, he also proved that this doubly exponential behavior cannot be improved. For more details about upper and lower bounds, see for example, Section 3.4.

In spite of all the bad news, practical experience shows that the algorithm often terminates after a reasonable time, and in theory it shall always terminate.

Lemma 3.7. *The basic algorithm terminates*

Proof. Let $\mathcal{G}_n$ denote the trial Gröbner basis that is produced by the algorithm at the n-th iteration step. We know that $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \mathcal{G}_3 \subset \cdots$, however, the most important part is that the sets of leading terms increase as well: $\mathsf{LT}_{\succ}(\mathcal{G}_1) \subset \mathsf{LT}_{\succ}(\mathcal{G}_2) \subset \mathsf{LT}_{\succ}(\mathcal{G}_3) \subset \cdots$. By Dickson's Lemma [1913], this process ends in a finite number of steps, once the sequence of leading terms stabilizes. See Buchberger [1965] for a complete proof. □

Theorem 3.8. *The basic algorithm is correct*

Proof. Since the algorithm terminates, we only have to show that at termination the set $\mathcal{G}$ is a Gröbner basis. By the previous characterization, we must show that for all $f, g \in G$, $\mathsf{NR}(S_{f,g}) = 0$. The loop invariant that we may observe is this: for all $f, g \in \mathcal{G}$,

if $\{f, g\} \notin B$ then $\mathsf{NR}(S_{f,g}) = 0$. This invariant holds at the beginning of the iterations. Each iteration preserves the invariant. Since the set $\mathbb{B}$ is empty on termination, the set $\mathcal{G}$ is indeed a Gröbner basis. □

3.3 Uniqueness of Reduced Gröbner Bases

The previous characterizations of Gröbner bases are not uniquely defined, e.g. they do not require a Gröbner basis to have a minimal number of generators, relative to the choice of an admissible ordering. This means that any finite subset of $\mathcal{I}$ that contains a Gröbner basis of $\mathcal{I}$ is also a Gröbner basis. Nevertheless, this lack of uniqueness can be solved by readjusting the definition of Gröbner basis into something called reduced Gröbner bases.

A *reduced Gröbner basis* is obtained by iteratively substituting all elements from $\mathcal{G}$ by their normal form with respect to the other elements. After deleting all zero elements from the resulting set and making all leading coefficients equal to 1, the resulting monic reduced Gröbner basis is unique, i.e. it only depends on $\mathcal{I}$ and the chosen term ordering.

Definition 3.9. $\mathcal{G}$ is a *minimal Gröbner basis* if for all $f \in \mathcal{G}$, $\langle \mathcal{G} \rangle \neq \langle \mathcal{G} - \{f\} \rangle$. A Gröbner basis $\mathcal{G}$ is *self-reduced* if for all $f \in \mathcal{G}$, f is a $\langle \mathcal{G} - \{f\} \rangle$-normal form. A Gröbner basis $\mathcal{G}$ is a *reduced Gröbner basis* if

1. for each $g \in \mathcal{G}$, the coefficient $\mathsf{LC}_{\succ}(g)$ is 1,
2. $\mathcal{G}$ is self-reduced, and
3. no trailing term of any $g \in \mathcal{G}$ lies in $\mathsf{LT}_{\succ}(\mathcal{I})$.

Lemma 3.10. *If $\mathcal{G}, \mathcal{G}'$ are minimal Gröbner basis of the same ideal, $\langle \mathcal{G} \rangle = \langle \mathcal{G}' \rangle$, then the set of leading terms in $\mathcal{G}$ is equal to the set of leading terms in $\mathcal{G}'$.*

Proof. Suppose for some $g \in \mathcal{G}$, $\mathsf{LT}_{\succ}(g)$ does not occur among the leading terms of polynomials in $\mathcal{G}'$. Since $\mathsf{NR}_{\succ,\mathcal{G}'}(g) = 0$ and $g \neq 0$, there is some $g' \in \mathcal{G}'$ such that $\mathsf{LT}_{\succ}(g')$ properly divides $\mathsf{LT}_{\succ}(g)$. Again, since $\mathsf{NR}_{\succ,\mathcal{G}}(g') = 0$, there is some $g'' \in \mathcal{G}$ such that $\mathsf{LT}_{\succ}(g'')$ divides (not necessarily properly) $\mathsf{LT}_{\succ}(g)$. This means that $\mathsf{LT}_{\succ}(g'')$ properly divides $\mathsf{LT}_{\succ}(g)$, contradicting the preceding lemma. □

Theorem 3.11. *The reduced Gröbner basis of an ideal is unique (relative to the choice of an admissible ordering).*

Proof. Let $\mathcal{G}, \mathcal{G}'$ be two reduced minimal Gröbner bases of the same ideal. We obtain a contradiction by supposing that there is some polynomial g in $\mathcal{G} - \mathcal{G}'$. By the previous

lemma, if they are different then there is some other polynomial g' in $\mathcal{G}' - \mathcal{G}$ such that $\mathsf{LT}_{\succ}(g) = \mathsf{LT}_{\succ}(g')$ (recall that $\mathsf{LC}_{\succ}(g) = 1 = \mathsf{LC}_{\succ}(g')$). Let $h = g - g'$. Then $h \neq 0$ and $\mathsf{NR}_{\succ,\mathcal{G}}(h) = 0$ since $\mathcal{G}$ is a Gröbner basis. Hence, some term t occuring in h can be eliminated by application of some $f \in \mathcal{G}$. Now t must occur in g or g'. If t occurs in g then g is reducible by f, contradicting the assumption that $\mathcal{G}$ is reduced. If t occurs in g" then let $g' \in \mathcal{G}'$ such that $\mathsf{LT}_{\succ}(g') = \mathsf{LT}_{\succ}(g)$. Again g' is reducible by f', contradicting the assumption that $\mathcal{G}'$ is reduced. □

Theorem 3.12. *Every nonzero polynomial ideal has a unique reduced Gröbner basis.*

Proof. See, for example, the proof of Theorem 2.4.13 of Section 2.4 in Kreuzer and Robbiano [2000]. □

In the next section we shall provide some applications of the Gröbner bases theory, followed by some glimpses on its computational complexity.

3.4 Applications of Gröbner Bases

The range of applicability of the Gröbner bases is enormous, e.g. mechanical assembly planning [Anantha et al., 1996; Cambon et al., 2009], constraint-based sketching and design [Michalik et al., 2002], kinematic analysis of robots [Nielsen and Roth, 1999], collision detection [Jia et al., 2011], geometric manipulation [Cambon et al., 2009], manipulation of offsets of curves and surfaces [Hoffmann, 1990], implicitization of parametric representations [Gao and Chou, 1992], algebraic cryptanalysis of cryptosystems [Buchmann et al., 2006; Ackermann and Kreuzer, 2006; Faugère and Joux, 2003], and geometric theorem proving [Kreuzer and Robbiano, 2005; Hoffmann, 1989; Chen et al., 2006; Recio and Vélez, 1999; Montes and Recio, 2007]. In this section, we attempt to describe a few of them. For additional examples, see for example Winkler [1996]; Cox et al. [2007].

Performing Calculations in Quotient Rings

Let $\mathcal{I}$ be an ideal in $\mathrm{P} = R[x_1, \ldots, x_n]$. Let $P/\mathcal{I}$ be the quotient ring defined as the set of equivalence classes modulo the ideal. Then, the theory of Gröbner bases can be conveniently used to carry out computations in such ring, as it provides a basis for its vector space. See Macaulay's Basis Theorem 2.22 and its proof. In fact, this is one of the main motivations behind their invention.

Solving the Ideal Membership Problem

The theory of Gröbner bases can also be used to solve the ideal membership problem for polynomial ideals. That is, given an ideal $\mathcal{I} \subset \mathrm{P}$ and a polynomial f in the polynomial ring P, we want to verify whenever $f \in \mathcal{I}$.

Theorem 3.13. *Let $\mathcal{I} \subset P$ be an ideal and f a polynomial in the polynomial ring P. If $\mathcal{G}$ is a Gröbner basis of $\mathcal{I}$, then the normal form of f with respect to $\mathcal{G}$ is zero if and only if $f \in \mathcal{I}$.*

Proof. See, for example, the proof of the Theorem 3.4. □

Example. *Suppose that we have the following polynomial relations (our axioms)*

$$\begin{aligned} 4z - 4xy^2 - 16x^2 - 1 &= 0, \\ 2y^2z + 4x + 1 &= 0, \\ 2x^2z + 2y^2 + x &= 0 \end{aligned}$$

and we want to check if the relation $g(x, y) = 4xy^4 + 16x^2y^2 + y^2 + 8x + 2 = 0$*, which represents our hypothesis, follows from the axioms. In other words, whether* g *is in the ideal* $\mathcal{I}$ *generated by the axioms.*

If we attempt to perform the normal reduction of g *by the given axioms, we get a normal form different from zero. However, if we compute a Gröbner basis for* $\mathcal{I}$ *with respect to the lexicographic ordering, then we have that* $g(x, y)$ *reduces to zero.*

Solving the Radical Membership Problem

In some domains like geometry, we are more interested in the radical membership problem than in the ideal membership problem.

Definition 3.14. Let $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$ be an ideal of $\mathrm{P} = K[x_1, \ldots, x_n]$. The radical of $\mathcal{I}$, denoted $\sqrt{\mathcal{I}}$, is defined as $\sqrt{\mathcal{I}} = \{f \in \mathrm{P} \mid \text{there exist } \alpha \in \mathbb{N} \text{ such that } f^\alpha \in \mathcal{I}\}$. So, $f \in \sqrt{\mathcal{I}} \iff f^n \in \mathcal{I}$ for some $n \in \mathbb{N}$.

Geometrically $f \in \sqrt{\mathcal{I}}$ means that the hypersurface defined by f contains all the points in the algebraic set defined by $f_1, \ldots, f_s$. The next theorem relates the radical of an ideal $\mathcal{I}$ to the set of common zeros $V(\mathcal{I})$ of the polynomials contained in $\mathcal{I}$.

Theorem 3.15. *(Hilbert's Nullstellensatz) Let* $\mathcal{I}$ *be an ideal in* $P = K[x_1, \ldots, x_n]$*, where* K *is an algebraically closed field. Then* $\sqrt{\mathcal{I}}$ *consists of exactly those polynomials in* P *which vanish on all the common zeros of* $\mathcal{I}$*.*

Proof. See, for example, Cox et al. [2007]. □

It turn out, from the Hilbert's Nullstellensatz that $f \in \sqrt{\mathcal{I}}$ if and only if f vanishes at every common zero of $f_1, \ldots, f_s$, in other words,

$$f \in \sqrt{\mathcal{I}} \iff 1 \in \langle f_1, \ldots, f_s, zf - 1 \rangle \in K[x_1, \ldots, x_n, z]$$

where z is a new additional indeterminate. Consequently, the radical membership problem can be reduced to the ideal membership problem.

Solution of Algebraic Equations Using Gröbner Bases

The theory of Gröbner bases can answer questions about solvability and number of zeros of a system of polynomials. Moreover, it can be used to find the solutions of a system of polynomials. Let $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ be a system of polynomial equations. Then, the first step is to check is if the system of polynomial equations has solutions in $\overline{K}^n$, $\overline{K}$ being the algebraic closure of K.

Theorem 3.16. *Let $\mathcal{I} = \langle f_1, \dots f_s \rangle$. Let $\mathcal{G}$ be a reduced Gröbner basis of $\mathcal{I}$. Then the system of polynomial equations $f_1(x_1, \dots, x_n) = \cdots = f_s(x_1, \dots, x_n) = 0$ is unsolvable in $\overline{K}^n$ if and only if $1 \in \mathcal{G}$.*

Proof. See, for example, Winkler [1996], Theorem 8.4.3. □

Suppose that the system of polynomial equations is solvable. Then, we have to determine whether the number of solutions is finite or not.

Theorem 3.17. *Let $\mathcal{G}$ be a Gröbner basis of $\mathcal{I}$. Then the system of polynomial equations $f_1(x_1, \dots, x_n) = \cdots = f_s(x_1, \dots, x_n) = 0$ has finitely many solutions if and only if for every $i, 1 \leq i \leq n$, there is a polynomial $g_i \in \mathcal{G}$ such that $\mathsf{LT}_{\succ}(g_i)$ is a pure power of x_i. Moreover, if $\mathcal{I}$ is 0–dimensional then the number of zeros of $\mathcal{I}$, counted with multiplicity, is equal to $dim(K[x_1, \dots, x_n]/\mathcal{I})$.*

Proof. See, for example, Winkler [1996], Theorem 8.4.4. □

Gröbner bases are also useful for solving systems of algebraic equations. A crucial observation is their elimination property, that is, if $\mathcal{G}$ is a Gröbner basis of $\mathcal{I}$ with respect to the lexicographic ordering, then the i-th elimination ideal of $\mathcal{I}$, i.e. $\mathcal{I} \cap K[x_1 \dots, x_i]$, is generated by those polynomials in $\mathcal{G}$ that depend only on the indeterminates $x_1, \dots, x_i$.

Theorem 3.18. *(Elimination Property of Gröbner Bases) Let $\mathcal{G}$ be a Gröbner basis of $\mathcal{I}$ with respect to the lexicographic ordering. Then $\mathcal{I} \cap K[x_1, \dots, x_i] = \langle \mathcal{G} \cap K[x_1, \dots, x_i] \rangle$, where the ideal on the right hand side is generated over the ring $K[x_1, \dots, x_i]$.*

Proof. See, for example, Winkler [1996], Theorem 8.4.5. □

Another characterization of the elimination property of the Gröbner bases is given by means of the Shape lemma.

Theorem 3.19 (Shape Lemma)**.** *If a system of polynomial equations $f_1(x_1, \dots, x_n) = 0, \dots, f_m(x_1, \dots, x_n) = 0$ has a finite number of solutions, then the Gröbner basis of the ideal generated by $\{f_1, \dots, f_n\}$, with respect to the pure lexicographic ordering and under some suitable assumptions, has a upper triangular structure.*

Proof. See, for example, [Becker et al., 1994; Kreuzer and Robbiano, 2000]. □

Equality of Ideals

The theory Gröbner bases provides also a systematic way to decide whether two given ideals are equal. The answer to this problem tell us, for example, if two different algebraic representations correspond to a certain geometric object. Hence, they have a wide applicability in fields like Computer Vision.

Let $\mathcal{I} = \langle f_1, \ldots, f_m \rangle \in K[x_1, \ldots, x_n]$ and $\mathcal{J} = \langle g_1, \ldots, g_k \rangle \in P$. Choose any admissible ordering. We know, from Theorem 3.12, that there exist for every nonzero polynomial ideal a unique reduced Gröbner basis. Hence, let $\mathcal{G}_\mathcal{I}, \mathcal{G}_\mathcal{J}$ be the unique reduced Gröbner bases of $\mathcal{I}$ and $\mathcal{J}$, respectively. Then by the Lemma 3.10, $\mathcal{I}$ is equal to $\mathcal{J}$ if and only if $\mathcal{G}_\mathcal{I} = \mathcal{G}_\mathcal{J}$, that is, they are both generated by the same reduced Gröbner basis.

Invertibility of Polynomial Mappings

The Jacobian conjecture states that a polynomial mapping has an inverse which is itself a polynomial mapping if and only if the determinant of the Jacobian of the mapping is different from zero. Essen, in an attempt of proving the conjecture, found the following criterion of invertibility:

Corollary 3.20 (Essen, 1991)**.** *Let $f_1, \ldots, f_n$ be the coordinate functions of a polynomial mapping in the indeterminates $x_1, \ldots, x_n$. Let $y_1, \ldots, y_n$ be new indeterminates and let $\prec$ be an admissible ordering such that $y \prec x$. Then the mapping is invertible if and only if the Gröbner basis of $\{y_1 - f_1, y_2 - f_2, \ldots, y_n - f_n\}$ has the form $\{x_1 - g_1, x_2 - g_2, \ldots, x_n - g_n\}$, where $g_1, g_2, \ldots, g_n$ are the coordinate functions of the inverse mapping.*

Proof. See, for example, [Essen, 1991]. □

It turns out that verification of the result by composition of mappings usually takes more time to compute than the Gröbner basis computation of the inverse mapping.

Integer Programming Problems

Integer programming problems are much harder to solve than linear programming problems. Nevertheless, they can be solved by means of Gröbner bases techniques.

Let $a_{ij}, b_i \in \mathbb{Z}$, and $c_j \in \mathbb{R}$, where $1 \leq i \leq n$ and $j = 1 \leq j \leq m$. Our goal is to find a solution $(\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{N}^m$ of the system bellow, that minimizes the cost function

$c(\sigma_1, \sigma_2, \ldots, \sigma_m) = \sum_{j=1}^{m} c_j \sigma_j$.

$$\begin{aligned} a_{11}\sigma_1 + a_{12}\sigma_2 + \cdots + a_{1m}\sigma_m &= b_1 \\ a_{21}\sigma_1 + a_{22}\sigma_2 + \cdots + a_{2m}\sigma_m &= b_2 \\ \vdots &= \vdots \\ a_{n1}\sigma_1 + a_{n2}\sigma_2 + \cdots + a_{nm}\sigma_m &= b_n \end{aligned}$$

Suppose we consider only the case when the a_{ij} and b_i are non-negative integers. Then we can write the previous system as a system of equations

$$x_i^{a_{11}\sigma_1 + a_{12}\sigma_2 + \cdots + a_{1m}\sigma_m} = x_i^{b_1}, \text{ for } i = 1 \ldots n$$

which can be reduced to a single equation

$$(x_1^{a_{11}} x_2^{a_{21}} \cdots x_n^{a_{n1}})^{\sigma_1} \cdots (x_1^{a_{1m}} x_2^{a_{2m}} \cdots x_n^{a_{nm}})^{\sigma_m} = x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$$

Now consider the polynomial map

$$K[y_1, \ldots, y_m] \longrightarrow K[x_1, \ldots, x_n]$$

defined by

$$\phi(y_1^{\sigma_1} \cdots y_m^{\sigma_m}) = (x_1^{a_{11}} x_2^{a_{21}} \cdots x_n^{a_{n1}})^{\sigma_1} \cdots (x_1^{a_{1m}} x_2^{a_{2m}} \cdots x_n^{a_{nm}})^{\sigma_m}$$

The next statements provide an algorithm for determining solutions for these polynomial systems.

Lemma 3.21. *There exists an integer solution $(\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{N}^m$ if and only if the term $x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$ is the image ϕ of a term $y_1^{\sigma_1} y_2^{\sigma_2} \cdots y_m^{\sigma_m} \in K[y_1, \ldots, y_m]$.*

Proof. See, for example, Adams and Loustaunau [1994] p.107-108. □

The proof given by Adams and Loustaunau introduces all the necessary tools required to describe an algorithm capable of addressing this problem, which is as follows:

- Compute a Gröbner basis $\mathcal{G}$ of $\mathcal{I} = \langle y_i - x_1^{a_{11}} x_2^{a_{21}} \cdots x_n^{a_{n1}} \mid j = 1, \ldots, m \rangle$ with respect to an elimination ordering, such that $y \prec x$.
- Compute the remainder $r = y_1^{\sigma_1} y_2^{\sigma_2} \cdots y_m^{\sigma_m}$ of the division of $x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$ by $\mathcal{G}$.
- If $r \notin K[y_1, \ldots, y_m]$ then the system does not have non-negative integer solutions. Otherwise, $(\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{N}^m$ is the solution.

For a more detailed discussion on the use of Gröbner bases techniques to solve integer programming problems, see for example Adams and Loustaunau [1994], as well as the expository papers [Thomas, 1995, 1998].

Classical Graph Vertex Colorability Problem

Vertex coloring is described as the following optimization problem. Let G be a simple, undirected graph with vertices $V = \{1, \ldots, n\}$. Let $E \in V^2$ be the set of edges of G, and

$$f_G = \prod_{(i,j)\in E,\, i<j} (x_i - x_j)$$

The polynomial representation of a given graph G. Let $C_k = \{c_1, \ldots, c_k\}$ be a set of colors for some natural number $k < n$. The graph G is said to be k-colorable if there exists a map $\gamma : V \longrightarrow C_k$ such that all adjacent vertices receive different colors. In that case, we say that a k-coloring γ is proper. Otherwise, it is called improper. Now, consider the following ideals:

$$\begin{aligned} \mathfrak{I}_{n,k} &= \langle x_i^k - 1 \mid i \in V \rangle, \\ \mathfrak{I}_{G,k} &= \mathfrak{I}_{n,k} + \langle x_i^{k-1} + x_i^{k-2}x_j + \cdots + x_i x_j^{k-2} + x_j^{k-1} \mid (i,j) \in E \rangle. \end{aligned}$$

In order to solve the colorability problem, one should think of the zeros of $\mathfrak{I}_{n,k}$ and $\mathfrak{I}_{G,k}$. Suppose that we want to color the vertices of a given graph G. Then how many different colors do we need if adjacent vertices cannot receive the same color? The next theorem gives an answer to the problem.

Theorem 3.22. *The k-colorability of a graph $\mathcal{G}$ is equivalent to have $1 \in \mathfrak{I}_{\mathcal{G},k}$ for a certain ideal $\mathfrak{I}_{\mathcal{G},k} \subseteq P$.*

Proof. See, for example, Mnuk [2002]. □

Let $l \leq k$ be the number of distinct colors in a proper k-coloring $\gamma(V)$. Let $cl(i)$ be the color class of a vertex $i \in V$, that is, the set of vertices having the same color as i. Let $\max\{cl(i)\}$ be the maximum of a color class. In other words, the largest vertex contained in it. Let $m_1 < m_2 < \cdots < m_l = n$ be the maximum of each l color class, and let $U \subseteq V$ be a subset of vertices. Then, the next theorem gives a result for graphs that have a unique proper k-coloring up to permutation of the colors in C_k.

Theorem 3.23. *A graph G with n vertices is uniquely k-colorable if and only if the reduced Gröbner basis of $\mathfrak{I}_{G,k}$ with respect to any term order with $x_n \prec \cdots \prec x_1$ has the form $\{g_1, \ldots, g_n\}$, where*

$$g = \begin{cases} x_i^k - 1 & \text{if } i = m_l \\ h^j_{\{m_j,\ldots,m_l\}} & \text{if } i = m_j \text{ for some } j \neq l, \\ h^1_{\{i,m_2,\ldots,m_l\}} & \text{if } i \in cl(m_1), \\ x_i - x_{\max\{cl(i)\}} & \text{otherwise.} \end{cases}$$

and where h_U^d is the sum of all terms of degree d in the indeterminates $\{x_i \mid i \in U\}$.

Proof. See, for example, Mnuk [2002]. □

The next figure shows a uniquely 3-colourable graph.

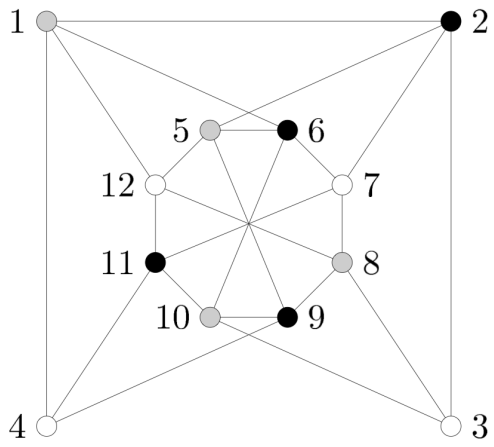

Figure 3.1: A uniquely 3-colourable graph

Many practical applications like the TDMA MAC protocol for wireless networks, depend heavily on algorithms to solve the graph vertex colourability problem. For a more detailed introduction to this problem, see for example, Mnuk [2002] and Loera et al. [2009].

3.5 Computational Complexity

This section aims at reviewing the literature concerning the computational complexity of a Gröbner basis. The main result on this subject has given by Thomas W. Dubé in [Dubé, 1990]. He has proved for the first time an upper bound for the degrees of elements in a reduced Gröbner basis, which depends on the maximum degree d of the generators of the ideal and on the number indeterminates.

Before we continue, it is necessary to know how to present the computational complexity of an algorithm, which can be describe through two different strategies: *time complexity* and *space complexity*. The *Big O Notation*, which was first used by Bachmann in 1894 and then popularized by Landau, is one of the simplest and most intuitive way of providing computational measurements. Basically, the *Big O Notation* describes the asymptotic behavior of functions, and informally, it tells us how fast a function grows or declines.

Definition 3.24. The *Big O Notation* can be defined as the set of functions g mapping $\mathbb{N}$ to $\mathbb{R}$, and for which there exists a real positive number c and a natural number a such that for all $n \geq a$, $|g(n)| \leq c|f(n)|$. In other words, $O(f)$ is the set of functions $\mathbb{N} \longrightarrow \mathbb{R}$

$$O(f) := \{g : \mathbb{N} \longrightarrow \mathbb{R} \mid \exists_c \in \mathbb{R}_{>0}, \exists_a \in \mathbb{N}, \forall_{n \geq a} |g(n)| \leq c|f(n)|\}$$

eventually bounded by f to within a constant factor.

Example. *Assume that the time complexity $f(n)$ of an algorithm is given by the following formula $f(n) = 64537n^6 + 23n^3 + 12$. If we take $c = 2$ and $a = 1$, then it is true that $\forall_{n \geq a}, n^6 \leq c|f(n)|$, so $n^6 \in O(f)$ by definition. Moreover, n^6 multiplied by a scalar c is an upper bound for the whole set, hence the time complexity of this algorithm is said to be $f(n) = O(n^6)$, or simply $O(n^6)$.*

Algorithms are also grouped according to *complexity classes*. Next, we recall the relation between some of the standard complexity classes,

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP} \subseteq \mathrm{EXPSPACE}$$

Although, we recall only the definition of those classes we are interested in, we refer the reader to Papadimitriou [2003] for a complete reading.

Definition 3.25. The complexity class of a problem is called:

1. P if the algorithm has complexity $O(g(n))$, where g is a polynomial in n.

2. SPACE($g(n)$) if all problems are solvable using $O(g(n))$ bits of memory.
3. PSPACE if the corresponding algorithm is solvable in polynomial space (but unlimited time). In other words, PSPACE is $\cup_k$ SPACE(n^k).
4. EXPSPACE if the corresponding algorithm has complexity $O(2^{g(n)})$, where g is a polynomial in n.

Next, we discuss the computational complexity for some of the problems presented in the previous section. But first, we recall again the problems that we are going to analyze. The reader should also be aware that lower bounds of the complexity of these problems yield lower bounds for the complexity of Buchberger's Algorithm.

Definition 3.26. Let $\mathrm{P} = K[x_1, \ldots, x_n]$ be a polynomial ring over K. We define the following problems:

1. The ideal membership problem is defined by
$$S_{\mathrm{im}} = \{(f, f_1, \ldots, f_s) \in \mathrm{P}^{s+1} \mid f \in \langle f_1, \ldots, f_s \rangle\}$$
2. The consistency problem is defined by
$$S_{\mathrm{cons}} = \{(f_1, \ldots, f_s) \in \mathrm{P}^{s} \mid 1 \in \langle f_1, \ldots, f_s \rangle\}$$
3. The radical membership problem is defined by
$$S_{\mathrm{rm}} = \{(f, f_1, \ldots, f_s) \in \mathrm{P}^{s+1} \mid f \in \sqrt{\langle f_1, \ldots, f_s \rangle}\}$$

An upper bound for the ideal membership problem was first given by Hermann, and it claims that it is double exponential in the number of indeterminates.

Theorem 3.27 (Hermann, 1926)**.** *Let $\mathcal{I}$ be an ideal $\langle f_1, \ldots, f_s \rangle \subseteq \mathbb{Q}[x_1, x_2, \ldots, x_n]$ and assume $d = \max\{\deg(f_1), \ldots, \deg(f_s)\}$. If $f \in \mathcal{I}$ then there are $q_1, \ldots, q_s \in \mathbb{Q}[x_1, x_2, \ldots, x_n]$ such that $f = q_1 f_1 + \cdots + q_s f_s$ and*
$$\deg(q_i) \leq \deg(f) + (sd)^{2^n} \text{ for all } i = 1, \ldots, s.$$

Proof. See Hermann [1998]; Seidenberg [1974]; Mayr and Meyer [1982] for a proof. □

This bound can be used to enumerate all the terms that can appear as q_i. Therefore the ideal membership problem leads to a system of linear equations that can be reduced to a rank computation of a matrix of size double exponential in the input size. At this

point, there exist algorithms on a parallel random access machine with a polynomial number of processors using polylogarithmic time, e.g see for example Mulmuley [1986]. Hence, it yields an algorithm in exponential space, assuming as hypothesis the parallel computation thesis, which claims that the time used by a parallel machine is polynomially related to the space used by a sequential machine. Recently, Mayr [1989] has shown that the entries of such matrix can be generated on the fly from the polynomial description. Moreover, that does not affect the algorithm used to compute the rank.

Corollary 3.28 (Mayr, 1989)**.** *The complexity of the ideal membership problem is*

$$S_{\mathrm{im}} \in \mathrm{EXPSPACE}$$

Proof. See, for example, the proof given by Mayr [1989]. □

For the consistency problem the upper degree bound can be improved to be single exponential in the number of indeterminates. Again, by the Rabinovich trick, this also yields an upper bound for the radical membership problem.

Theorem 3.29. *Let P be a polynomial ring over K in n indeterminates and $\mathfrak{I}$ an ideal $\langle f_1, \ldots, f_s \rangle \subseteq P$. Let $\mu = \min\{s, n\}$ and $d = \max\{\deg(f_1), \ldots, \deg(f_s)\}$.*

1. *Assume that f_i has no common zero in $\mathbb{C}^n$, then there are $q_1, \ldots, q_s \in P$ with $1 = q_1 f_1 + \cdots + q_s f_s$ such that*

$$\deg(q_i) \leq \mu n d^{\mu} + \mu d \text{ for } \mathrm{i} = 1, \ldots, \mathrm{s}.$$

2. *Assume that $h(x) = 0$ for all common zeros x of the f_i in $\mathbb{C}^n$, where $h \in P$. Then there are*

$$e \in \mathbb{N}^+ \text{ s.t. } \mathrm{e} \leq (\mu+1)(\mathrm{n}+2)(\mathrm{d}+1)^{\mu+1}, \text{ and}$$

$$q_i \in P \text{ with } \deg(\mathrm{q_i}) \leq (\mu+1)(\mathrm{n}+2)(\mathrm{d}+1)^{\mu+2} \text{ for } \mathrm{i} = 1, \ldots, \mathrm{s}$$

such that

$$h^e = q_1 f_1 + \cdots + q_s f_s$$

Proof. For proofs of these exponential degree bounds, see [Berenstein and Yger, 1990; Brownawell, 1987]. □

Similar techniques can be combined with these bounds to show the following result,

Corollary 3.30. *Both consistency and the radical membership problem have PSPACE complexity.*

$$S_{\mathrm{cons}} \in \mathrm{PSPACE} \text{ and } \mathrm{S_{rm}} \in \mathrm{PSPACE}.$$

Proof. See, for example, the proof given by Mayr [1989]. □

An upper bound for the degree of elements in the reduced Gröbner basis is also known, and was first presented by Dubé [1990].

Theorem 3.31 (Dubé, 1990)**.** *Let $\mathcal{G}$ be the reduced Gröbner basis for an ideal $\mathcal{I} = \langle f_1, \ldots, f_s \rangle \subseteq P$ and $\prec$ any term ordering on P. Let $d = \max\{\deg(f_1), \ldots, \deg(f_s)\}$. Then the degree of polynomials in the reduced Gröbner basis is upper bounded by*

$$d = \max\{\deg(g) \mid \ g \in \mathcal{G}\} \leq 2\left(\frac{d^2}{2} + d\right)^{2^{n-1}}$$

Proof. See, for example, the proof given by Dubé [1990]. □

Roughly, this means that we have a doubly-exponential bound on the degrees of the elements in the Gröbner basis. However, it should also be clear that these bounds strongly depend on the input data. For example, Caniglia et al. studied the the bounds associated with the lexicographical ordering, and the graded reverse lexicographical ordering.

Theorem 3.32 (Caniglia et al., 1989)**.** *The complexity of computing a Gröbner basis for $\mathcal{I}$, with respect to the graded reverse lexicographical ordering is $O(d^k)$, where $k = n^2$ if $\#(\{a \in K^n \mid f_i(a) = 0 \text{ for all } f_i \in \mathcal{I}\}) < \infty$, and $k = n$ if the solutions at infinity are also finite.*

Proof. See, for example, Caniglia et al. [1989, 1993]. □

The same computations with respect to the lexicographical ordering leads to computations with a complexity of $k = n^3$.

Theorem 3.33 (Caniglia et al., 1989)**.** *The complexity of computing a Gröbner basis for $\mathcal{I}$, with respect to the lexicographical ordering is $O(d^{n^3})$.*

Proof. See, for example, Caniglia et al. [1989, 1993]. □

We encourage the reader to read some of the most relevant results in this subject, such as [Bayer and Stillman, 1988; Giusti, 1985, 1987, 1990; Möller and Mora, 1984], which present more special cases where upper bounds can be given.

3.6 Optimization Strategies and Techniques

The Buchberger algorithm requires several decisions during the computational process. At the beginning, we have to fix a particular admissible term ordering. Then at each step of the while-loop, we have to choose a critical pair. Lastly, we have a reduction process that is in general not unique. Hence, it might be possible to have different reduction steps leading to different normal forms.

Although the choices made in the computational process do not have influence in the final reduced Gröbner basis, they are relevant to the performance of the algorithm. Depending on the choices, we might have to compute more S-polynomials than necessary, which are simply discarded after the reduction process (see Theorem 3.6). However, from a computational point of view this is something that we need to avoid.

In this section, we recall a few strategies to improve the computation of a Gröbner basis. The interested reader is also encouraged to read some of the most relevant results on this topic, such as Buchberger [1979, 1985]; Lazard [1983]; Gebauer and Möller [1986]; Giovini et al. [1991]; Möller et al. [1992]; Faugère [1999, 2002]; Caboara et al. [2004, 2002, 1996]; Bigatti et al. [2011]; Gao et al. [2010b], and Gianni et al. [1994].

3.6.1 Strategies Involving the Selection of Term Orderings

The time and space complexity of an algorithm to compute a Gröbner basis is highly dependent on the term ordering used. Experiments show that the total degree inverse lexicographic ordering (DegRevLex) is among the most efficient orderings for the Buchberger algorithm. However, the lexicographic ordering is more useful for solving systems of equations. Hence, it might happen that we have to compute a Gröbner basis with respect to a particular term ordering that is computationally less efficient. A feasible strategy is to compute a Gröbner basis with respect to some computationally efficient ordering and then transform it into a Gröbner basis for the desired term ordering.

The most efficient algorithms for basis conversion are the Gröbner walk algorithm [Collart et al., 1997], see Amrhein et al. [1997] and Tran [2000] for additional variants; the Faugère-Gianni-Lazard-Mora algorithm [Faugère et al., 1993] for zero-dimensional systems, which can be used to convert Gröbner bases with respect to any ordering into Gröbner bases with respect to the pure lexicographic ordering, by means of linear algebra; and by means of the Hilbert functions, see Gianni et al. [1994], Traverso [1996] or the Section 4.4 of this thesis.

Another strategy to optimize the computation of a Gröbner basis, if no particular term ordering is required, is to determine a good term ordering to start with, see, for example, Tran [2005; 2007]; or to seek a more efficient term ordering than the one it has started with, see for example, Caboara and Perry [2012]. If a particular term ordering is required, then we have to convert the basis as mentioned above.

3.6.2 Identification of Useless S-polynomials

The computation and reduction of a S-polynomial to a normal form, is a time-consuming task that should be avoided whenever possible. The worst case is when the normal form is zero, which means that such S-polynomial is not necessary to the final Gröbner basis. A possible computational improvement is to reduce the number of S-polynomials that has to be considered, hence reducing the number of divisions required. In 1979, Buchberger introduced two inexpensive criteria to identify certain pairs of polynomials whose S-polynomials will reduce to zero. An implementation of both criteria is given by Gebauer and Möller [1988].

- *Criterion 1*: Let f and g be two polynomials with disjoint leading terms, i.e. $\gcd(\mathsf{LM}_{\succ}(f), \mathsf{LM}_{\succ}(g)) = 1$. Then $S_{f,g}$ reduces to zero with respect to $\{f, g\}$.
- *Criterion 2*: Suppose there are elements p, f, and g in the current basis $\mathcal{G}$ such that $\mathsf{LM}_{\succ}(p)$ divides $\mathsf{lcm}(\mathsf{LM}_{\succ}(f), \mathsf{LM}_{\succ}(g))$, and $(f, p), (g, p) \notin \mathbb{B}$. Then the pair (f, g) can be simply discarded.

Another way to predict if a S-polynomial reduces to zero involves syzygies. The use of syzygies can be easily understood by using an observation by Lazard [1983]. Lazard pointed out the connection between Gröbner basis and linear basis of the same ideal. Since the Buchberger's algorithm can be considered as a triangularisation of a linear basis by Gaussian elimination, the reduction of a polynomial to zero means a linear dependence of this polynomial on the polynomials employed in the reduction procedure. Because polynomials are of type $t_i f_i$, where $f_i \in R$ and t_i is a term, the linear dependence relation can be written as a syzygial relation

$$\sum_{f_i \in R} g_i f_i = 0$$

where the g_i are linear combinations of terms t_{ij}. Assuming that a suitable basis of the module of syzygies is known, then we can predict every S-polynomial zero reduction. For more details see Möller et al. [1992].

The connection between Gröbner bases and linear algebra methods is the key aspect in many algorithms such as the F4 algorithm by Faugère [1999], and the XL (eXtended Linearization) type algorithms by Courtois et al. [2000]; Ding et al. [2008]; Mohamed et al. [2008], and Mohamed et al. [2010].

3.6.3 Selection of Critical Pairs for the Reduction Process

At each reduction step we have to choose a critical pair for processing, and different choices might lead to different normal forms. Another useful strategy lies on how to order and how to choose critical pairs for processing.

- *Normal strategy* [Buchberger, 1985]. If the input is composed of homogeneous polynomials, then choose a pair (f, g) such that the least common multiple of the leading terms $\mathsf{LT}_{\succ}(f)$ and $\mathsf{LT}_{\succ}(g)$ is minimal in the current term ordering. Some variants use properties such as the total degree [Kreuzer and Robbiano, 2000], indices of the generators of the critical pair, or the age of the critical pairs.
- *"Sugar" and "Double sugar" strategy* [Giovini et al., 1991]. For inhomogeneous polynomials, the *sugar* and *double sugar* strategy give a good choice (refinement and heuristics). In the sugar strategy, critical pairs are ordered with respect to a phantom degree called sugar, which is the degree that a pair would have if the input is homogenized. Hence, we can sort the pairs as they would be sorted in the case of a homogenization, but without the overhead that is required by the homogenization process.
- *Self-saturation strategy* [Bigatti et al., 2011]. This strategy aims at inhomogeneous computations, and describes how an idea centered on the concept of self-saturation allows several improvements in the computation of Gröbner bases via Buchberger's Algorithm. In general, we can say that it seeks the balance between the advantages of homogeneous strategies and the idea of the sugar strategy.

3.6.4 Removal of Superfluous Polynomials

The last step of the computation of the reduced Gröbner basis is to remove all the superfluous polynomials from the basis. Nevertheless, the removal of superfluous polynomials during the computation of an intermediate basis $\mathcal{G}$ is often a good strategy. In other words, at each reduction step, we have to reduce the S-polynomial to the normal form by means of a smaller list of elements $\mathcal{G}$. The removal process can be executed in the following way: for $g, g' \in \mathcal{G}, g \neq g'$, if $\mathsf{LM}_{\succ}(g)$ divides $\mathsf{LM}_{\succ}(g')$, then g' can be

expressed by g and the S-polynomial $S_{g,g'}$. Once we reduce $S_{g,g'}$ into normal form, g' is superfluous and all pairs (g', g'') can be deleted from B. Although g' is discarded, you can still use it in the normal form algorithm.

3.6.5 Selection Strategies for Reducers

The aim of this strategy is to render useless computations useful for further reduction steps. There exists an extensive literature regarding this topic; however, we restrict ourselves to methods that are known to be also applicable for signature-based algorithms. Brickenstein [2004, 2010] discovered during a deeper inspection of F4, a few methods to select good reducers. The idea is to use properties related to the possible reducers, e.g the number of elements in the support, the combination of the length of the support with the information concerning the leading coefficient, to decide which one is the best. The main drawback is the amount of memory that is required, since we have to keep a huge amount of data related to the reducers.

3.6.6 Input Transformation Strategies

It is also possible to benefit from faster computations by working with a transformed input. An optimization for symmetric ideals using this type of strategy was proposed in Steidel [2012]. The idea is to construct and use an appropriate linear transformation that exploits the symmetric properties of the input ideal to obtain smoother operations while performing the Buchberger algorithm. Another known optimization is based on the transformation of non-homogeneous ideals into homogeneous ones, so we can sort polynomials by degree. Then, at the end of computation, we perform the dehomogenization.

3.6.7 Parallel and Modular Strategies

Another optimization strategy concerns the integration of modular computations into the Gröbner bases algorithm. This idea was originally influenced by the work of Borosh and Fraenkel [1966] and Ebert [1983], and initially presented by Traverso [1989] and Winkler [1988].

The motivation behind modular computations is strongly associated to the coefficient growth during the computation steps of a Gröbner basis. In each single reduction step, the leading coefficient c_1 of the reducer p_1 has to be adjusted to match the leading coefficient c_2 of the element to be reduced. Therefore, not only the fraction $\frac{c_2}{c_1}$ must be computed, but also every coefficient in p_1 must be multiplied by this fraction. This

can lead to enormous numbers, whose calculations slow down the Gröbner basis computation tremendously. This is where the concept of modular computations comes into play. The idea is to compute many Gröbner bases over fields of prime characteristic $p < \infty$, and then, with the help of algorithms for the reconstruction of rational numbers [Collins and Encarnación, 1994; Kornerup and Gregory, 1983; Pan and Wang, 2004; Wang et al., 1982; Wang and Pan, 2003], merge these modular Gröbner bases together and lift the coefficients using the Chinese Remainder Theorem.

The use of parallelization has also been suggested. Although the idea is not completely new, it is coming back to fashion again, mainly due to the development of multicore and multiprocessor computers, on which the independent modular computations can be run in parallel. For example, some authors such as Chakrabarti and Yelick [1993] and Vidal [1990] have constructed general algorithms for distributed memory and shared memory machines respectively. Reasonable speedups were obtained on a small numbers of processors. Another approach has been using factorization of polynomials; all generated S-polynomials are factorized on a master node, and the reductions of its factors are carried out on the slave nodes, see Siegl [1994]; Bradford [1990]; Gräbe and Lassner [1994]. In a paper by Reeves [1998] a parallelization at the compiler level for modular coefficient fields is presented. Moreover, in a paper by Leykin [2004] a coarse grained parallelism was studied and implemented both in the commutative and non-commutative case. Recently, Faugère and Lachartre [2010] presented a strategy on how to parallelize the F4 matrix operations in polynomial rings over finite fields, e.g. grouping the matrix into different blocks, using different attempts for sparse, semi-sparse, and dense blocks. Independently, Arnold [2003] and Idrees et al. [2011] presented also similar works.

See surveys by Mityunin and Pankratiev [2007] and Amrhein et al. [1998] for more information regarding parallel and modular strategies.

3.6.8 Signature-based Strategies

In 2002, Faugère proposed together with an algorithm called F5 [Faugère, 2002], the concept of signature of a polynomial. This algorithm was considered for a while the fastest algorithm to compute a Gröbner basis. However, its termination was unclear. Faugère, Hashemi and Ars tried to prove the F5's finite termination problem in their papers [Faugère, 2002; Hashemi and Ars, 2010], but the termination itself was proved only for the case of input being regular polynomial sequence. The correctness of F5 was also shown for any input that terminates. In 2012, Galkin proved the termination for any homogeneous input without any reference to regularity.

Recently, new variants and termination proofs were presented by Stegers [2005]; Eder [2008]; Zobnin [2010]; Eder and Perry [2010]; Eder et al. [2011]; Eder and Perry [2011]; Arri and Perry [2011]; Sun and Wang [2011b,a]; Roune and Stillman [2012]; Eder [2012]; Sun and Wang [2012]. Many of these variants simply add extra criteria to F5, and most of these proofs assume the input to be homogeneous, or the critical pair that is associated to the smallest signature to be computed first.

Worth mentioning are the variants presented by Gao et al. [2010b,a], since they propose new conceptions and techniques to compute a Gröbner basis. e.g., they propose the conception of pairs; generalize the signature order to be an arbitrary one; use arbitrary top reductions instead of F5 reductions; etc. At that time, the termination of Gao-Volny-Wang (GVW) algorithm has an open problem. However, a termination proof was recently given by Huang [2010].

In Chapter 4, we recall the ideas behind signature-based strategies, and we analyse the main differencies between most variants. For now, we close this section with an overview of the relations between signature-based variants.

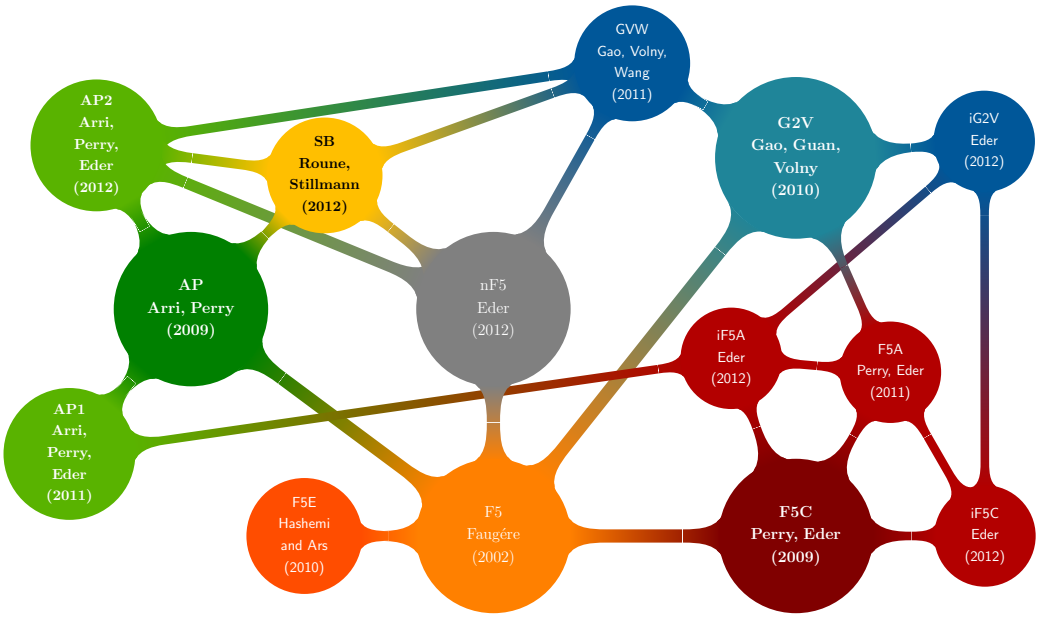

Figure 3.2: The most efficient variants of F5

3.7 Concluding Remarks

In this chapter we have presented a wide range of strategies that can be used to improve the computation of a Gröbner basis. Despite any benefit that these approaches might have, they often have drawbacks, too. For example, one has often to consider restrictions on the input. Moreover, the efficiency of these methods is highly dependent on the behavior of the data during the computations, which cannot be known beforehand.

The signature-based algorithms are not an exception. However, we can see that in most cases they will find more useless critical pairs than Buchberger's Criteria. Within the signature-based algorithms, performance is affected by restrictions on the reduction process and by the overhead that is generated due to how aggressive is the signature-based criteria chosen. Thus, again the question is not about getting a universal best algorithm, but even more about how to combine the signature-based world with already highly efficient improvements of the classic world without harming the performance or causing wrong results.

Chapter 4

New Strategies for Computing Gröbner Bases

The mind that opens to a new idea never returns to its original size.
(Albert Einstein)

This chapter is the connecting link between the theory of Gröbner bases and the concept of signature-based strategies for computing a Gröbner basis, whose introduction follows in Section 4.3. First we recall the notion of syzygies and free resolutions, which are fundamental to fully understand the signature-based algorithms proposed. Then, we introduce the theory on which signature-based algorithms are based. Lastly, we introduce two new optimization techniques for such algorithms. One is a criterion based on the Hilbert Series, and the other one benefits from computer parallelization.

4.1 Syzygies Modules

In this section, we shall see how to compute a system of generators for a given module of syzygies. We start by recalling the concept of the exact sequence of the module of syzygies $\mathrm{Syz}(\mathcal{G})$ and the exact sequence of $\mathrm{Syz}(\mathsf{LT}_{\succ}(\mathcal{G}))$. Then we establish a connection between the module of syzygies $\mathrm{Syz}(\mathcal{G})$ and the Gröbner basis $\mathcal{G}$.

In Chapter 2, we said that a P-module M is finitely generated if there exists elements $m_1, \ldots, m_n$ in M such that $\forall m \in M, \exists r_1, \ldots, r_n$ elements in P with:

$$m = r_1 m_1 + \cdots + r_n m_n$$

In this section, we recall the notion of modules of syzygies, which is given by the following definition.

Definition 4.1. Given a P-module M, the *first syzygy module* of M on a set of generators $(m_1, \ldots, m_n)$ is the kernel the following presentation of M:

$$\begin{array}{ccc} P^n & \xrightarrow{\times(m_1,\ldots,m_n)} & M \longrightarrow 0 \\ (r_1, \ldots, r_n) & \mapsto & (r_1 m_1 + \cdots + r_n m_n) \end{array}$$

In other words, we have that $\mathrm{Syz}_P(m_1, \ldots m_n) := \{(r_1, \ldots r_n) \in P^n \mid \sum_i r_i m_i = 0\}$, and $M \simeq P^n/\mathrm{Syz}_P(m_1, \ldots m_n)$. The elements $(r_1, \ldots r_n)$ are denoted *syzygies* of M.

Next we recall the concept of principal syzygies, whose computation is required in our algorithm.

Definition 4.2. Let $M = \langle m_1, \ldots, m_r \rangle$. Then any element $m_j e_i - m_i e_j$, where $j < i$, is called a *principal syzygy of* M. We denote the module of all principal syzygies by $\mathrm{PSyz}(m_1, \ldots, m_r)$.

In the previous chapter, we have seen that it is usually a good strategy to reduce questions about polynomials to questions about their leading terms. So, the next definition recalls the concept of syzygies on the leading terms of some ideal $\mathcal{I}$. Then, we shall see its relation to the definition of Gröbner bases.

Definition 4.3. Let $\mathcal{I} = (f_1, \ldots f_n)$ be a tuple of n polynomials. The *syzygies on the leading terms* of $\mathcal{I}$ are denoted by:

$$\mathrm{Syz}_P(\mathsf{LT}_\succ(f_1), \ldots, \mathsf{LT}_\succ(f_n)) := \left\{ (h_1, \ldots, h_n) \in P^n \mid \sum_i h_i \, \mathsf{LT}_\succ(f_i) = 0 \right\}.$$

The proposition bellow gives a result analogous to the definition of S-polynomial, which is necessary to introduce our next theorem.

Proposition 4.4. *Let $Syz(f_1, \ldots, f_s)$ be the syzygy module on the leading terms of $f_1, \ldots, f_s$. Consider the pair (f_i, f_j) such that $1 \leq i \leq j \leq s$, and define $t^\mu := \mathsf{lcm}(\mathsf{LT}_\succ(f_i), \mathsf{LT}_\succ(f_j))$. Let*

$$S_{ij} := \frac{t^\mu}{\mathsf{LM}_\succ(f_i)} \mathbf{e}_i - \frac{t^\mu}{\mathsf{LM}_\succ(f_j)} \mathbf{e}_j \in P^n$$

The syzygies $\{S_{ij}\}_{1 \leq i,j \leq s}$ generate $Syz(f_1, \ldots, f_s)$ as a P-module.

Proof. See, for example, Kreuzer and Robbiano [2009]. □

Theorem 4.5. *Let $\mathcal{G} = \{g_1, \ldots, g_s\}$ be a tuple of polynomials, and Syz($\mathsf{LT}_\succ(\mathcal{G})$) the Syzygy module on the leading terms of $\mathcal{G}$. Let S be a homogeneous generating set of Syz($\mathsf{LT}_\succ(\mathcal{G})$). Then we have:*

$$\mathcal{G} \text{ is a Gröbner basis if and only if for all } z \in S, z \cdot \mathcal{G} = \mathsf{NR}_\mathcal{G}\left(\sum_{i=1}^{s} h_i g_i\right) = 0$$

Proof. See, for example, Eisenbud [1995]. □

The advantage of using this criterion is the possibility to take a smaller generating set for Syz($\mathsf{LT}_\succ(\mathcal{G})$) than the $\{S_{ij}\}$. Hence, we can avoid more useless pairs than the Buchberger criterion.

4.2 Free Resolutions

The computation of syzygies is fundamental for free resolution of ideals and modules. A free resolution of a module M (or ideal) is simply a representation of M in terms of generators, the relations between the generators (denoted first syzygies), the relations between the relations of the generators (denoted second syzygies), etc.

Definition 4.6. Consider a sequence of R-modules and homomorphisms

$$\cdots \longrightarrow M_{k+1} \xrightarrow{\phi_{k+1}} M_k \xrightarrow{\phi_k} M_{k-1} \longrightarrow \cdots$$

The expression is called an *exact sequence* if $\mathrm{Im}(\phi_k) = \mathrm{Ker}(\phi_{k+1})$. The inclusion $\mathrm{Im}(\phi_k) \subset \mathrm{Ker}(\phi_{k+1})$ is equivalent to the equality $\phi_k \phi_{k+1} = 0$. A sequence of several mappings is called exact if and only if it is exact at each M_i that is not located at the beginning or the end of the sequence.

Definition 4.7. Let M be a R-module. A *free resolution* of M is an exact sequence of the form

$$\cdots \longrightarrow R^{n_2} \xrightarrow{\phi_2} R^{n_1} \xrightarrow{\phi_1} R^{n_0} \xrightarrow{\phi_0} M \longrightarrow 0$$

Observe that all modules in this sequence except M are free. If there is an $l \in N$ s.t. $n_l \neq 0$ but $n_k = 0$ for all $k > l$, then we say that the resolution is finite. More precisely, the resolution has length l. A finite resolution of length l is usually written as

$$0 \longrightarrow R^{n_l} \longrightarrow R^{n_{l-1}} \longrightarrow \cdots \longrightarrow R^{n_1} \longrightarrow R^{n_0} \longrightarrow M \longrightarrow 0.$$

Now, we shall see how to construct a free resolution of a finitely generated module $M = \langle m_1, \ldots, m_{n_0} \rangle$. We determine a generating set $\{s_1, \ldots, s_{n_1}\}$ of $\mathrm{Syz}(m_1, \ldots, m_{n_0})$, the syzygy module of $(m_1, \ldots, m_{n_0})$. Let

$$\begin{array}{rcccc} \phi_0 & : & R^{n_0} & \longrightarrow & M \\ & & (r_1, \ldots, r_{n_0})^t & \mapsto & \sum r_i m_i \end{array}$$

$$\begin{array}{rcccc} \phi_1 & : & R^{n_1} & \longrightarrow & R^{n_0} \\ & & (r_1, \ldots, r_{n_1})^t & \mapsto & \sum r_i s_i \end{array}$$

Then we have $\mathrm{Im}(\phi_1) = \mathrm{Syz}(m_i) = \mathrm{Ker}(\phi_0)$, so the sequence

$$R^{n_1} \xrightarrow{\phi_1} R^{n_0} \xrightarrow{\phi_0} M \longrightarrow 0$$

is exact. Continuing this process with $\mathrm{Syz}(m_i)$ instead of M, we finally get a free resolution of M.

Example. *Consider the following ideal, which is also a module*

$$\mathfrak{I} = \underbrace{\langle x^2 - x, xy, y^2 - y \rangle}_{F}$$

in $R = K[x, y]$. *In geometric terms,* $\mathfrak{I}$ *is the ideal of* $\{(0,0), (1,0), (0,1)\}$ *in* K^2. *Let*

$$\begin{array}{rcccc} \phi_0 & : & R^3 & \longrightarrow & \mathfrak{I} \\ & & \begin{pmatrix} 1 \\ r_2 \\ r_3 \end{pmatrix} & \mapsto & \underbrace{\langle x^2 - x, xy, y^2 - y \rangle}_{F} \cdot \begin{pmatrix} 1 \\ r_2 \\ r_3 \end{pmatrix} \end{array}$$

The mapping ϕ_0 *represents the generation of* $\mathfrak{I}$ *from the free module* R^3. *Next we shall determine the relations between the generators, i.e. (first) syzygies. The columns of the matrix*

$$B = \begin{pmatrix} y & 0 \\ -x+1 & y-1 \\ 0 & -x \end{pmatrix}$$

generate the syzygy module Syz(F*). So for* ϕ_1 *we have that,*

$$\begin{array}{rcccc} \phi_1 & : & R^2 & \longrightarrow & R^3 \\ & & \begin{pmatrix} 1 \\ r_2 \end{pmatrix} & \mapsto & B \cdot \begin{pmatrix} 1 \\ r_2 \end{pmatrix} \end{array}$$

we get the exact sequence

$$R^2 \xrightarrow{\phi_1} R^3 \xrightarrow{\phi_0} \mathfrak{I} \longrightarrow 0.$$

Hence, the resolution procedure terminates. If (c_1, c_2) *is any syzygy of the columns of* B*, i.e. a second syzygy of* F*, then*

$$c_1 \begin{pmatrix} y \\ -x+1 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ y-1 \\ -x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

By looking at the first component, we get that $c_1 y = 0$*, so* $c_1 = 0$*. Similarly, from the third component we have* $c_2 = 0$*. Hence the kernel of* ϕ_1 *is the zero module* 0*. There are no non-trivial relations between the columns of* B*, so the first syzygy module* $Syz(F)$ *is the free module* R^2*. Finally this leads to the free resolution*

$$0 \longrightarrow R^2 \xrightarrow{\phi_1} R^3 \xrightarrow{\phi_0} \mathfrak{I} \longrightarrow 0.$$

of length 1 of the module $\mathfrak{I}$ *in* $R = K[x, y]$*.*

Lastly, we recall Hilbert's famous "Syzygy Theorem", published in 1890, which states that every finitely generated graded module M over the polynomial ring $\mathrm{P} = K[x_1, \ldots, x_n]$ has a free resolution of length at most n, that is, its n-th syzygy is free.

Theorem 4.8 (Hilbert's Syzygy Theorem)**.** *Every finitely generated graded P-module* M *has a finite free resolution of finitely generated free graded P-modules, which has length at most* n*.*

Proof. Proof of the above theorem can be found in Eisenbud [1995][Corollary 19.7]. □

The length is one less than the number of free modules in the resolution. Hilbert used this result to prove that the Hilbert function $i \longrightarrow \dim_P(M_i)$ is, for large i, a polynomial function of i. For more details, see Section 2.6.

4.3 Introduction to Signature-based Strategies

The main idea of the signature-based strategies for computing a Gröbner basis is to associate each polynomial with a signature on which the criteria and reduction steps depend. Such a concept it was initially proposed by Faugère together with his F5 algorithm [Faugère, 2002], and has become extremely popular partly due to its good performances.

The most efficient variants of F5 are: the F5C algorithm by Eder and Perry [2010], the F5 with extended criteria by Hashemi and Ars [2010], the Gao-Guan-Volny (G2V) [Gao et al., 2010a], and the Gao-Volny-Wang (GVW) [Gao et al., 2010b]. The main differences between these algorithms are the extra conditions implemented to ensure correctness of the strategy.

Suppose, for example, that f and g are two polynomials, where t_f and t_g are terms such that the leading terms of $t_f f$ and $t_g g$ are the same. Then a necessary condition to reject the critical pair of f and g is that, there exists some known polynomial h such that its signature is a factor of $t_f f'$s or $t_g g'$s signature. We shall see that this condition alone is not sufficient to ensure correctness. Thus, existing implementations use different extra conditions.

In this section, we give a brief introduction to signature-based strategies. First, we recall the concept of polynomial signature. Then, we describe a generic footprint for signature-based algorithms. Lastly, we introduce a detailed description of the GVW algorithm. For proofs of the correctness and termination, see for example, Faugère [2002]; Stegers [2005]; Eder [2008]; Eder and Perry [2010]; Hashemi and Ars [2010], and Gao et al. [2010a].

Generic Signature-based Algorithm

Let $\mathrm{P} = K[x_1, \ldots, x_n]$ be a polynomial ring over a field K with n indeterminates. Let $\succ$ be a fixed term ordering in P, which equips its elements with a unique representation. Let $\mathcal{F} = \{g_1, \ldots, g_m\}$ such that $g_i \in \mathrm{P}$, and $\mathcal{J}$ be the ideal generated by the elements of $\mathcal{F}$,

$$\mathcal{J} = \langle g_1, \ldots, g_m \rangle = \{u_1 g_1 + \cdots + u_m g_m : u_1, \ldots, u_m \in \mathrm{P}\} \subseteq \mathrm{P} \tag{4.1}$$

Then we can define the following map

$$\begin{array}{rcccc} \phi & : & \mathrm{P}^m & \longrightarrow & \mathcal{J} \\ & & \sum_{i=1}^m u_i \mathbf{e}_i & \mapsto & \sum_{i=1}^m u_i g_i \end{array}$$

such that the u_i's are polynomials in P, and $\mathbf{e}_1, \ldots, \mathbf{e}_m$ are the canonical generators of the free P-module P^m. From Definition 4.1, we have that the elements $\mathbf{u} \in P^m$ with

$\phi(\mathbf{u}) = 0$ are called *syzygies* of $g_1, \ldots, g_m$. We denote the module of all such syzygies by the short notation $\mathbf{H}$.

$$\mathbf{H} := \mathrm{Syz}(\mathcal{F}) = \{(u_1, \ldots, u_m) \in P^m : u_1 g_1 + \cdots + u_m g_m = 0\} \tag{4.2}$$

Moreover, we denote elements of P^m, which we see as row vectors, with bold letters *e.g.* $\mathbf{g}, \mathbf{u}$ etc. Next, we consider the following P-submodule of $\mathrm{P}^m \times \mathrm{P}$:

$$M = \{(\mathbf{u}, v) \in \mathrm{P}^m \times \mathcal{I} : \mathbf{u}\mathbf{g}^t = v\} \tag{4.3}$$

generated by

$$(\mathbf{e}_1, g_1), (\mathbf{e}_2, g_2), \ldots, (\mathbf{e}_m, g_m) \tag{4.4}$$

Now that all the settings are in place, we introduce our first definition of polynomial signature.

Definition 4.9 (Faugère)**.** Let $\mathrm{P} = K[x_1, \ldots, x_n]$ and $v \in \mathcal{I} \subseteq \mathrm{P}$ a polynomial. Let $u_1, \ldots, u_m \in \mathrm{P}$ and $\mathbf{u} \in \mathrm{P^m}$ such that $v = \phi(\mathbf{u})$, where $\mathbf{u} := \sum_{i=1}^{m} u_i \mathbf{e}_i$. We say that $\mathsf{LT}_\succ(\mathbf{u})$ is a *signature* of $(\mathbf{u}, v) \in \mathrm{P}^m \times \mathrm{P}$.

We can easily deduce from definition that each polynomial $v \in \mathrm{P}$ has a unique minimal signature once we fix a term ordering. Now, we recall the concept of signatures for critical pairs.

Definition 4.10. Suppose $(\mathbf{u}_1, v_1), (\mathbf{u}_2, v_2) \in \mathrm{P}^m \times \mathrm{P}$ are two pairs with v_1 and v_2 both nonzero. Let

$$t_1 = \frac{\mathsf{lcm}(\mathsf{LT}_\succ(v_1), \mathsf{LT}_\succ(v_2))}{\mathsf{LT}_\succ(v_1)}, \quad t_2 = \frac{\mathsf{lcm}(\mathsf{LT}_\succ(v_1), \mathsf{LT}_\succ(v_2))}{\mathsf{LT}_\succ(v_2)}$$

Suppose $\max(t_1 \,\mathsf{LT}_\succ(\mathbf{u}_1), t_2 \,\mathsf{LT}_\succ(\mathbf{u}_2)) = t_i \,\mathsf{LT}_\succ(\mathbf{u}_i)$. Then

- $t_i \,\mathsf{LT}_\succ(\mathbf{u}_i)$ is the *J-signature* of $(\mathbf{u}_1, v_1)$ and $(\mathbf{u}_2, v_2)$, and
- $t_i(\mathbf{u}_i, v_i) = (t_i \mathbf{u}_i, t_i v_i)$ is a *J-pair*[1] of $(\mathbf{u}_1, v_1)$ and $(\mathbf{u}_2, v_2)$, and

Again, we have that a J-signature of $(\mathbf{u}_1, v_1)$ and $(\mathbf{u}_2, v_2)$ is unique. Due to the uniqueness of the J-signature, we only need to keep a J-pair per distinct J-signature. Moreover, we are free to pick between $t_1(\mathbf{u}_1, v_1)$ and $t_2(\mathbf{u}_2, v_2)$ if the equality $t_1 \,\mathsf{LT}_\succ(\mathbf{u}_1) = t_2 \,\mathsf{LT}_\succ(\mathbf{u}_2)$ holds.

The terms t_1 and t_2 used in J-pairs are exactly the same as those used in Buchberger's S-polynomial $t_1 v_1 - c t_2 v_2$, where $c = \mathsf{LC}_\succ(v_1)/\,\mathsf{LC}_\succ(v_2)$. The main difference

[1] J means "joint" of the two pairs

between these two concepts is that J-pairs postpone the computation of S-polynomials until the reduction phase, as we work with their signatures. This idea should become more clear with the introduction of the GVW algorithm.

Lastly, we revise the reduction process, which shall also depend on the concept of signature.

Definition 4.11. Let $(\mathbf{u}_1, v_1)$ and $(\mathbf{u}_2, v_2)$ be two J-pairs in M. Moreover, let $\mathcal{G} \subset M$ with $\#\mathcal{G} = k$. We say that $(\mathbf{u}_1, v_1)$ *reduces signature-safely* to $(\mathbf{u}_2, v_2)$ modulo $\mathcal{G}$ if there exist a sequence $r_0, \ldots, r_k$, where $r_0 = (\mathbf{u}_1, v_1)$ and $r_k = (\mathbf{u}_2, v_2)$, such that for all $i \in \{1, \ldots, k\}$ there exist $g_i = (\mathbf{u}, v) \in \mathcal{G}, t_i \in \mathrm{P}^m$, and $c_i \in K$ fulfilling the following properties:

1. $r_i = r_{i-1} - c_i t_i g_i$,
2. $\mathsf{LT}_\succ(v_i) \succ \mathsf{LT}_\succ(v_{i-1})$, and
3. $t_i \, \mathsf{LT}_\succ(\mathbf{u}) \succ \mathsf{LT}_\succ(\mathbf{u}_{i-1})$

If there exists a pair $(s, h) \in \mathcal{G}$ such that $\mathsf{LT}_\succ(s)$ divides $\mathsf{LT}_\succ(\mathbf{u}_1)$ and $\mathsf{LT}_\succ(h)$ divides $\mathsf{LT}_\succ(v_1)$, then we say that $(\mathbf{u}_1, v_1)$ is *signature-redundant* to $\mathcal{G}$.

So far, we have introduced the notion of J-pair and a procedure to reduce them. However, a Gröbner basis algorithm without any criteria to discard useless critical pairs, is not efficient at all. In the signature-based strategies there exist two main criteria to discard useless critical pairs:

1. The *non-minimal signature criterion* is mainly based on the knowledge of syzygies, respectively of their leading terms, and their comparison to the signatures of the critical pairs.
2. The *rewritable signature criterion* is based on the fact that we can detect relations between polynomials to be computed, simply by looking at the corresponding signatures.

The exact details on how to implement these criteria depend on the specific implementation. However, it should be clear by now how to implement them trivially. Later, we shall see a concrete example with the introduction of the GVW algorithm. For additional details, see for example Faugère [2002]; Stegers [2005]; Eder [2008]; Eder and Perry [2010]; Hashemi and Ars [2010], and Gao et al. [2010a].

The generic footprint of the signature-based strategies is described by the following algorithm.

Input: A set of polynomials $\mathcal{F} = \{g_1, \ldots, g_m\}$ and a fixed term ordering $\succ$ in P
Output: A Gröbner basis $\mathcal{G}$ of the ideal $\mathcal{I} = \langle \mathcal{F} \rangle$
Algorithm:
Initialize $\mathcal{G}$ as $\{(\mathbf{e}_1, g_1), \ldots, (\mathbf{e}_m, g_m)\}$;
Compute all the J-pairs of $(\mathbf{e}_1, g_1), \ldots, (\mathbf{e}_m, g_m)$ to S;
while *S is not empty* **do**
    Take a minimal J-pair $(\mathbf{u}, v)$ from S (with respect to signature);
    Delete the J-pair $(\mathbf{u}, v)$ from the list S;
    while $(\mathbf{u}, v)$ *reduces signature-safely by the pairs in* $\mathcal{G}$ **do**
        Reduce $(\mathbf{u}, v)$ signature-safely to $(\mathbf{u}_i, v_i)$;
        Set $(\mathbf{u}, v) := (\mathbf{u}_i, v_i)$;
    end
    if $v \neq 0$ *and* $(\mathbf{u}, v)$ *is not signature-redundant to* $\mathcal{G}$ **then**
        Append $(\mathbf{u}, v)$ to $\mathcal{G}$;
        Form new J-pairs between $(\mathbf{u}, v)$ and S;
        Add to S only the J-pair with minimal $\mathsf{LT}_\succ(v)$ for each distinct signature $\mathsf{LT}_\succ(\mathbf{u})$;
    end
end
return v-part of $\mathcal{G}$;

Algorithm 3: Generic Signature-based Algorithm

The GVW Algorithm

The main results of this thesis, which are introduced throughout the next sections, are built on top of the GVW algorithm. Hence, we shall also give a technical introduction to this algorithm.

The main difference between signature-based algorithms concerns the way they implement the criteria. This algorithm is not an exception. The definition of signature-safely has now in two particular cases: when v_2 is nonzero and when it is not.

Definition 4.12. Let $(\mathbf{u}_1, v_1), (\mathbf{u}_2, v_2) \in \mathrm{P}^m \times \mathrm{P}$ be any two pairs. If v_2 is nonzero, $\mathsf{LT}_\succ(v_2)$ divides $\mathsf{LT}_\succ(v_1)$ and $\mathsf{LT}_\succ(t\mathbf{u}_2) \succeq \mathsf{LT}_\succ(\mathbf{u}_1)$ then we say $(\mathbf{u}_1, v_1)$ is *top-reducible* by $(\mathbf{u}_2, v_2)$. The corresponding *top-reduction* is

$$(\mathbf{u}_1, v_1) - ct(\mathbf{u}_2, v_2) = (\mathbf{u}_1 - ct\mathbf{u}_2, v_1 - ctv_2), \tag{4.5}$$

where $t = \mathsf{LT}_\succ(v_1)/\mathsf{LT}_\succ(v_2)$ and $c = \mathsf{LC}_\succ(v_1)/\mathsf{LC}_\succ(v_2)$.
When v_2 is zero and $\mathbf{u}_1, \mathbf{u}_2 \neq 0$, then we say that $(\mathbf{u}_1, v_1)$ is top-reducible by $(\mathbf{u}_2, 0)$ if $\mathsf{LT}_\succ(\mathbf{u}_2)$ divides $\mathsf{LT}_\succ(\mathbf{u}_1)$. For such scenario the top-reduction is

$$(\mathbf{u}_1, v_1) - ct(\mathbf{u}_2, v_2) = \left(\mathbf{u}_1 - \frac{\mathsf{LM}_\succ(\mathbf{u}_1)}{\mathsf{LM}_\succ(\mathbf{u}_2)}\mathbf{u}_2, v_1\right)$$

Every time we perform a top-reduction, we cancel the leading term in the v-part without increasing the signature of $(\mathbf{u}_1, v_1)$. Moreover, we can say that,

Definition 4.13. A top-reduction is called *regular top-reduction*, if

$$\mathsf{LT}_\succ(\mathbf{u}_1 - ct\mathbf{u}_2) = \mathsf{LT}_\succ(\mathbf{u}_1),$$

and *super top-reduction* otherwise. This means that the signature of $(\mathbf{u}_1, v_1)$ becomes smaller under a super top-reduction and remains unchanged under a regular top-reduction. Observe that a super top-reduction happens if

$$\mathsf{LT}_\succ(t\mathbf{u}_2) = \mathsf{LT}_\succ(\mathbf{u}_1) \text{ and } \frac{\mathsf{LC}_\succ(\mathbf{u}_1)}{\mathsf{LC}_\succ(\mathbf{u}_2)} = \frac{\mathsf{LC}_\succ(\mathrm{v}_1)}{\mathsf{LC}_\succ(\mathrm{v}_2)}$$

Furthermore, if a pair $(\mathbf{u}_1, v_1)$ is top-reducible by $(\mathbf{u}_2, 0)$, then the top-reduction is always a super top-reduction. Moreover, one can also claim that a pair $(\mathbf{u}_1, 0)$ is never top-reducible by $(\mathbf{u}_2, v_2)$ for $v_2 \neq 0$. Note that we do not have to perform reductions for the super top-reduction case.

The definition of top-reductions given by Faugère [2002] corresponds to these regular top-reductions. However, some of the regular top-reductions allowed in this definition are not allowed in F5 algorithm (e.g. when $\mathsf{LT}_\succ(\mathbf{u}_1) = t\,\mathsf{LT}_\succ(\mathbf{u}_2)$).

Lemma 4.14. *Let t be a term in P. If a pair $t(\mathbf{u}_1, v_1)$ is (regular) top-reducible by $(\mathbf{u}_2, v_2)$, where both v_1 and v_2 are nonzero, then $t_1(\mathbf{u}_1, v_1)$ is a J-pair of $(\mathbf{u}_1, v_1)$ and $(\mathbf{u}_2, v_2)$, where*

$$t_1 = \frac{\mathsf{lcm}(\mathsf{LT}_\succ(v_1), \mathsf{LT}_\succ(v_2))}{\mathsf{LT}_\succ(v_1)}$$

and t_1 is a divisor of t. Furthermore, $t_1(\mathbf{u}_1, v_1)$ is (regular) top-reducible by $(\mathbf{u}_2, v_2)$.

Proof. See Gao et al. [2010a] □

Proposition 4.15. *Suppose that $L = \{(\mathbf{u}_1, v_1), \ldots, (\mathbf{u}_k, v_k)\}$ is a strong Gröbner basis of M, that is, every pair $(\mathbf{u}, v) \in M$ is top-reducible by some pair in L. Then one can derive the following conclusions*

- *The set $\mathcal{G}_0 = \{v_i : 1 \leq i \leq k\}$ is a Gröbner basis of the ideal $\mathcal{I} = \langle g_1, \ldots, g_m \rangle$*
- *The set $\mathcal{G}_1 = \{\mathbf{u}_i : v_i = 0,\ 1 \leq i \leq k\}$ is a Gröbner basis of the syzygy module of $\mathbf{g} = (g_1, \ldots, g_m)$*

A strong Gröbner basis of $M \subset \mathrm{P}^m \times \mathrm{P}$ is a Gröbner basis of M as a submodule of P^{m+1}. However the converse may not be true for an arbitrary submodule M of P^{m+1}. This is why we call the basis a strong Gröbner basis.

Proof. In order to prove that $\mathcal{G}_1$ is a Gröbner basis of the syzygy module of $\mathbf{g}$, one can deduce from equations (4.2) and (4.3) that we must have $(\mathbf{u}, 0) \in M$ for any $\mathbf{u} = (u_1, \ldots, u_m)$ in the syzygy module of $\mathbf{g}$. Now, suppose that $(\mathbf{u}, 0)$ is top-reducible by some pair $(\mathbf{u}_i, v_i)$ in L. Then by Definition (4.13), v_i must be zero. Therefore, $\mathbf{u}_i \in \mathcal{G}_1$ and $\mathsf{LT}_\succ(\mathbf{u})$ is reducible by $\mathsf{LT}_\succ(\mathbf{u}_i)$.

Now, to prove that $\mathcal{G}_0$ is a Gröbner basis of $\mathcal{I}$, assume $v \in \mathcal{I}$ and nonzero. Then we know there exists $\mathbf{u} = (u_1, \ldots, u_m) \in \mathrm{P}^m$ so that $\mathbf{u}\mathbf{g}^t = v$, hence $(\mathbf{u}, v) \in M$. Among all such $\mathbf{u}$, we pick the one having the smallest leading term. Since $(\mathbf{u}, v) \in M$, it is top-reducible by some $(\mathbf{u}_i, v_i)$ where $1 \leq i \leq k$. Assume, by absurd, that $v_i = 0$. Then we could use $(\mathbf{u}_i, 0)$ to reduce $(\mathbf{u}, v)$ to get a $\mathbf{u}'$ such that $\mathbf{u}'\mathbf{g}^t = v$, with $\mathsf{LT}_\succ(\mathbf{u}')$ smaller than $\mathsf{LT}_\succ(\mathbf{u})$, which contradicts the initial minimality of $\mathsf{LT}_\succ(\mathbf{u})$. So $v_i \neq 0$ and $\mathsf{LT}_\succ(v_i)$ divides $\mathsf{LT}_\succ(v)$. Therefore, we have that $\mathcal{G}_0$ is a Gröbner basis of $\mathcal{I}$. □

Lastly, we recall the definition of eventually super top-reducible and we introduce an analogous theorem to Buchberger's theorem.

Definition 4.16. Let L be any set of pairs in M as defined in (4.3). We say that $(\mathbf{u}, v)$ is *eventually super top-reducible* by L if there is a sequence of regular top-reductions of $(\mathbf{u}, v)$ by pairs in L that reduce $(\mathbf{u}, v)$ to a pair $(\mathbf{u}', v')$ that is no longer regular top-reducible by L but is super top-reducible by at least one pair in L.

Theorem 4.17. *Let L be any set of pairs in M as defined in (4.3). Then L is a strong Gröbner basis of M if and only if for every distinct J-signature from L there is at least one J-pair from L with the same J-signature that is eventually super top-reducible by L.*

Proof. See Gao et al. [2010a] for a proof of this theorem. □

The algorithm that puts together all the pieces follows next.

Input:
List of polynomials $g_1, \ldots, g_m \in \mathrm{P} = K[x_1, \ldots, x_n]$,
A term ordering for P, and a term ordering on P^m
Output:
A Gröbner basis V of $\mathcal{I} = \langle g_1, \ldots, g_m \rangle$, and
A Gröbner basis H of $\mathsf{LT}_{\succ}(\mathrm{Syz}(g_1, \ldots, g_m))$, the leading terms of the syzygy module
Variables:
U a list of terms T_i, representing signatures of $(\mathbf{u}_i, v_i) \in M$,
V a list of polynomials v_i for $(\mathbf{u}_i, v_i) \in M$,
H a list of $\mathsf{LT}_{\succ}(\mathbf{u})$ were $\mathbf{u} \in R^m$ is a syzygy found so far,
JP a list of pairs (t, i) s.t. $t(u_i, v_i)$ is the J-pair of $(\mathbf{u}_i, v_i)$ and $(\mathbf{u}_j, v_j)$ for some $j \neq i$.
Algorithm:

```
1  Initialize U as {e_1, ..., e_m};
2  Initialize V as {g_1, ..., g_m};
3  Add the leading terms of the principle syzygies g_j e_i − g_i E_j for 1 ≤ i < j ≤ m to H ;
4  Compute all the J-pairs of (e_1, g_1), ..., (e_m, g_m);
5  Add to the list JP all J-pairs whose signatures are distinct and not reducible by H ;
6  while JP is not empty do
7      Take a minimal pair (t, i) from JP (with respect to signature) ;
8      Delete the pair (t, i) from the list JP ;
9      while t(T_i, v_i) is regular top-reducible by the pairs in(U, V) do
10         Perform regular top-reduction, say to get(T, v) ;
11     end
12     if v = 0 then
13         Append T to H ;
14         Delete every J-pair (t, j) in JP whose signature tT_j is divisible by T ;
15     end
16     if v ≠ 0 and (T, v) is not super top-reducible by (U, V) then
17         Append T to U and v to V;
18         Form all J-pairs for (T, v) and (T_j, v_j), 1 ≤ j ≤ |U| − 1,and ;
19         Consider only J-pairs whose signatures are not reducible by H, and ;
20         Add to JP only the J-pair with minimal LT_≻(v) for each distinct signature T
           ;
21         Add leading terms of the principle syzygies, vT_j − v_j T for 1 ≤ j ≤ |U| − 1, to
           H ;
22     end
23 end
24 return V and H;
```

Algorithm 4: The GVW algorithm

We conclude this section with a proof for the correctness, which is based on the proof presented by Gao et al. [2010b]. For a termination proof, see for example Huang [2010].

Theorem 4.18. *If the Algorithm 4 terminates, then V is a Gröbner basis of $\mathcal{I} = \langle g_1, g_2, \ldots, g_m \rangle$ and $\mathsf{LT}_\succ(Syz(g_1, \ldots, g_m))$ is a Gröbner basis of the leading terms of the syzygy module of $(g_1, g_2, \ldots, g_m)$.*

Proof. To prove the correctness of the algorithm we have to show that:

1. One is allowed to delete J-pairs at line 5, 14, and 20, whose signatures are divisible by $\mathsf{LT}_\succ(\mathbf{u})$, where $\mathbf{u} \in \mathrm{Syz}(g_1, \ldots, g_m)$;

   Proof: Let $(\mathbf{u}, v)$ be any pair whose signature $\mathsf{LT}_\succ(\mathbf{u})$ is divisible by $\mathsf{LT}_\succ(\mathbf{u}')$ for some $\mathbf{u}' \in \mathrm{Syz}(g_1, \ldots, g_m)$. Then we have that $(\mathbf{u}, v)$ is top-reducible by $(\mathbf{u}', 0)$. Moreover, regular top-reductions of $(\mathbf{u}, v)$ would not change the $\mathsf{LT}_\succ(\mathbf{u})$, thus, the pair obtained from $(\mathbf{u}, v)$ after we perform regular top-reductions will be super top-reducible by $(\mathbf{u}', 0)$. Hence $(\mathbf{u}, v)$ is eventual super top-reducible by the current basis. Since it is worthless to reduce $(\mathbf{u}, v)$, we can simply discard it.

2. A pair that is eventually super top-reducible by an intermediate basis is always eventually super top-reducible by the final basis;

   Proof: Let us suppose that the final Gröbner basis computed for M is $\mathcal{G}_k = \{(\mathbf{u}_1, v_1), \ldots, (\mathbf{u}_k, v_k)\}$. At any intermediate step, only $\mathcal{G}_p = \{(\mathbf{u}_1, v_1), \ldots, (\mathbf{u}_p, v_p)\}$ is known for some $p < k$. Suppose that the smallest J-pair from JP is (t, i). If $t(\mathbf{u}_i, v_i)$ is eventually super top-reducible by $\mathcal{G}_p$, then $t(\mathbf{u}_i, v_i)$ remains eventually super top-reducible by $\mathcal{G}_k$, since every $(\mathbf{u}_j, v_j), j > p$, has strictly larger signature than $t(\mathbf{u}_i, v_i)$. If $t(\mathbf{u}_i, v_i)$ is not eventually super top-reducible by $\mathcal{G}_p$, then a new pair $(\mathbf{u}_{p+1}, v_{p+1})$, which is obtained from $t(\mathbf{u}_i, v_i)$ via regular top-reductions by $\mathcal{G}_p$, will be added to $\mathcal{G}_p$. Hence the J-pair $t(\mathbf{u}_i, v_i)$ is eventually super top-reducible by the new basis $\mathcal{G}_{p+1} = \mathcal{G}_p \cup \{(\mathbf{u}_{p+1}, v_{p+1})\}$.

   Observe that $(\mathbf{u}_{p+1}, v_{p+1})$ has exactly the same signature as the J-pair $t(\mathbf{u}_i, v_i)$. Furthermore, all new J-pairs formed using $(\mathbf{u}_{p+1}, v_{p+1})$ will have strictly greater signature than that of $(\mathbf{u}_{p+1}, v_{p+1})$, because we discard all future J-pairs having the same signature. Hence $(\mathbf{u}_{p+1}, v_{p+1})$ can not be top-reducible by any pair $(\mathbf{u}_j, v_j), j > p + 1$, so the J-pair $t(\mathbf{u}_i, v_i)$ remains eventually super top-reducible by $\mathcal{G}_k$. Therefore, any pair that is eventually super top-reducible by current basis remains super top-reducible by the final basis.

3. It is only required to store one J-pair for each signature, which follows directly from Theorem 4.17. Note that the initial basis consists of pairs in (U, V) and $(\mathrm{Syz}(g_1, \ldots, g_m), 0)$.

□

4.4 The Hilbert-Driven Strategy

In this section, we introduce a new criterion to signature-based strategies that is based on the so-called *Hilbert-driven Gröbner basis algorithm*, originally presented by Traverso [1996].

Current attempts based on this strategy have a common drawback: one needs to know the Hilbert Series beforehand in order to take advantage of it. Some work about how to compute it efficiently can be found in Bigatti [1997]. In some special cases, we have all the information we need about the Hilbert Series without any further computations.

Theorem 4.19. *Let $\succ$ be a term ordering on P, and $\mathcal{I} \subset P$ a homogeneous ideal. Then we have the following relation*

$$\mathsf{HP}(P/\mathcal{I}, t) = \mathsf{HP}(P/\,\mathsf{LT}_\succ(\mathcal{I}), t)$$

Proof. See for example Section 5.2 in Greuel and Pfister [2007] or Kreuzer and Robbiano [2000]. □

Using the above theorem one can derive a few other properties, which are nicely described through the following statement.

Corollary 4.20. *Let $\mathcal{I} \in P$ be an ideal and $\succ$ a term ordering. Let $\mathcal{G} = \{g_1, \ldots, g_s\} \subset \mathcal{I}$. Then it holds:*

1. $\mathsf{HF}(P/\,\mathsf{LT}_\succ(\mathcal{I}), d) \leq \mathsf{HF}(P/\,\mathsf{LT}_\succ(\mathcal{G}), d)$ *for all d.*
2. *If* $\mathsf{HF}(P/\,\mathsf{LT}_\succ(\mathcal{I}), d) = \mathsf{HF}(P/\,\mathsf{LT}_\succ(\mathcal{G}), d)$ *for all d, then $\mathcal{G}$ is a Gröbner basis of $\mathcal{I}$.*

Proof. The proof is based on the fact that $\mathsf{LT}_\succ(\mathcal{G}) \subset \mathsf{LT}_\succ(\mathcal{I})$. Having $\mathsf{LT}_\succ(\mathcal{G}) \subset \mathsf{LT}_\succ(\mathcal{I})$, the equality of the Hilbert functions follows from the equality of the leading ideals, i.e. $\mathsf{LT}_\succ(\mathcal{G}) = \mathsf{LT}_\succ(\mathcal{I})$. But this is just the definition of $\mathcal{G}$ being a Gröbner basis of $\mathcal{I}$. □

Corollary 4.21. *Let $\succ_1$ and $\succ_2$ be two term orderings on P and $\mathcal{I} \subseteq P$ an ideal.*

1. *If $\mathcal{I}$ is homogeneous, then* $\mathsf{HF}(P/\,\mathsf{LT}_{\succ_1}(\mathcal{I}), d) = \mathsf{HF}(P/\mathcal{I}, d) = \mathsf{HF}(P/\,\mathsf{LT}_{\succ_2}(\mathcal{I}), d)$ *for all d.*
2. *If $\mathcal{I}$ is inhomogeneous, then* $\mathsf{HF}(P/\,\mathsf{LT}_{\succ_1}(\mathcal{I}), d) = \mathsf{HF}(P/\mathcal{I}, d) - \mathsf{HF}(P/\mathcal{I}, d-1) = \mathsf{HF}(P/\,\mathsf{LT}_{\succ_2}(\mathcal{G}), d)$ *for all d.*

Proof. See for example Traverso [1996]. □

Another important corollary, which is given by Traverso in [Traverso, 1996], describes how to use the Hilbert Series to improve the computations of Gröbner bases for inhomogenous ideals.

Corollary 4.22. *Let $\mathcal{I} \subset P$ be an ideal and $\succ_1$ and $\succ_2$ two term orderings on P. Let $\mathcal{G}_1$ be a Gröbner basis of $\mathcal{I}$ with respect to $\succ_1$. Let $\mathcal{G}_2$ be the Gröbner basis of $\mathcal{G}_1$ with respect to $\succ_2$. Then we can use the following variant algorithm:*

1. *Consider critical pairs by increasing degree.*
2. *If the degree decreases during a reduction step, then the reduced element can be deleted and the next pair can be computed.*

Proof. See for example Traverso [1996]. □

This corollary is very useful when the computation of a Gröbner basis with respect to $\succ_1$ is easier than the computation with respect to $\succ_2$.

The next theorem from Traverso [1996] gives one of the most important results of this section.

Theorem 4.23. *Let $\mathcal{I}$ and $\mathcal{J}$ be two homogeneous ideals in P such that $\mathcal{J} \subset \mathcal{I}$. By Theorem 2.32 there exist polynomials $p(t) = \sum_{i=0}^{v} p_i t^i$ and $q(t) = \sum_{j=0}^{w} q_j t^j$ such that the corresponding Hilbert Series are*

$$\mathsf{HS}(P/\mathcal{I}, t) = \frac{p(t)}{(1-t)^n} \text{ and } \mathsf{HS}(P/\mathcal{J}, t) = \frac{q(t)}{(1-t)^n}.$$

Then the following conditions are equivalent:

1. $\mathsf{HF}(P/\mathcal{I}, t) = \mathsf{HF}(P/\mathcal{J}, t)$ *for all* $1 \leq t \leq d-1$ *and* $\mathsf{HF}(P/\mathcal{I}, d) < \mathsf{HF}(P/\mathcal{J}, d)$.
2. $p(i) = q(i)$ *for* $1 \leq i \leq d-1$ *and* $p(d) < q(d)$.

Proof. See for example Traverso [1996]. □

An important observation is that we can compute the corresponding Hilbert Series of $\mathcal{I}$ without computing a Gröbner basis of $\mathcal{I}$ beforehand, that if, $V(\mathcal{I})$ is a complete intersection. The following lemma shows how.

Lemma 4.24. *If $V(\mathcal{I})$ is a complete intersection for $\mathcal{I} = \langle f_1, \ldots, f_r \rangle$ where f_i is homogeneous of degree $\deg(f_i) = d_i$ for all $1 \leq i \leq r$, then the Hilbert Series is given by*

$$\mathsf{HS}(P/\mathcal{I}, t) = \frac{\prod_{i=1}^{r}(1-t^{d_i})}{(1-t)^n} \tag{4.6}$$

Proof. See for example Section 5 of Kreuzer and Robbiano [2005]. □

Before we introduce the last criterion of this section, it might be useful to describe the Hilbert-driven Gröbner basis algorithm proposed by Traverso. The description bellow is based on the implementation suggested by Gebauer-Möller, which is considered to be the most efficient.

Hilbert-driven Gröbner basis algorithm

Let $\succ_1$ be a term ordering on P and let $\mathcal{I}$ be the ideal for which we want to compute a Gröbner basis. Assume that we know the Hilbert function $\mathsf{HF}(\mathrm{P}/\mathcal{I}, t)$. Keep in mind that it can be a consequence of one of the following facts:

1. A previous Gröbner basis computation for $\mathcal{I}$ with respect to some other term ordering.
2. Or a special shape of the ideal $\mathcal{I}$: a complete intersection where the formula is known (see Equation 4.6).

Let $\mathsf{HF}(\mathrm{P}/\mathcal{I}, t)$ be the known Hilbert function of $\mathcal{I}$. If $\mathcal{I}$ is homogeneous and $\succ_1$ is a term ordering on P, then we can compute the Gröbner basis $\mathcal{G}$ of $\mathcal{I}$ by increasing degree (see Section 3.6.3). Let us assume that we have already an intermediate Gröbner basis $\mathcal{G}$ for some degree $d \geq 0$. Then, we compute the Hilbert function $\mathsf{HF}(\mathrm{P}/\,\mathsf{LT}_{\succ}(\mathcal{G}), t)$. It holds that

$$\mathsf{HF}(\mathrm{P}/\,\mathsf{LT}_{\succ}(\mathcal{G}), t) = \mathsf{HF}(\mathrm{P}/\mathcal{I}, t) \text{ for all } t \leq d.$$

Analogously, we have the following correspondence

$$\mathsf{HF}(\mathrm{P}/\,\mathsf{LT}_{\succ}(\mathcal{G}), t) = \mathsf{HF}(\mathrm{P}/\mathcal{I}, t) + m_t \text{ for all } t, m_t \geq 0.$$

In the Algorithm 5, we initially set the variable m_t to infinity because we do not have any information about $\mathcal{G}$. However, the value shall be readjusted once we compute the first reduction step and the Hilbert function $\mathsf{HF}(\mathrm{P}/\,\mathsf{LT}_{\succ}(\mathcal{G}), t)$ (Line 32).

Input:
An ideal $\mathcal{I} = \langle f_1, \ldots, f_r \rangle \subset \mathrm{P}$, of homogeneous elements, and $\succ$ a term ordering on P
$\mathsf{HF}(\mathbf{P}/\mathcal{I}, t)$ **the Hilbert function of** $\mathcal{I}$
Output: $\mathcal{G}$ a Gröbner basis of $\mathcal{I}$ with respect to $\prec$
Algorithm:

```
1  m_t := ∞ ;
2  G ← f_1;
3  B ← ∅;
4  d' ← 0;
5  for i = 2, ..., r do
6  |   B ← UpdateCriticalPairs(B, G, f_i);
7  |   G ← G ∩ {f_i};
8  end
9  l ⟵ r;
10 while B is not empty do
11 |   d ← min{d | d = deg(S_fg), (f, g) ∈ B};
12 |   B' ← {(f, g) ∈ B | deg(S_fg) = d'};
13 |   B ← B\B';
14 |   while B' is not empty and m_t > 0 do
15 |   |   (f, g) ← First element of B';
16 |   |   B' ← B'\{(f, g)};
17 |   |   h ← S_fg;
18 |   |   h ← NR(h, G);
19 |   |   if h ≠ 0 then
20 |   |   |   f_{l+1} ← h;
21 |   |   |   B ← UpdateCriticalPairs(B, G, f_{l+1});
22 |   |   |   G ← G ∪ {f_{l+1}} ;
23 |   |   |   l ← l + 1;
24 |   |   |   m_t ← m_t − 1 ;
25 |   |   end
26 |   end
27 |   if HF(P/LT_≻(G), t) = HF(P/I, t) for all t then
28 |   |   return G;
29 |   end
30 |   else
31 |   |   d' ← min{t ∈ ℕ | HF(P/LT_≻(G), t) > HF(P/I, t)} ;
32 |   |   m_t ← HF(P/LT_≻(G), d') − HF(P/I, d') ;
33 |   |   B'' ← {(f, g) ∈ B | deg(S_fg)) < d'} ;
34 |   |   B ← B\B'' ;
35 |   end
36 end
37 return G;
```

Algorithm 5: Hilbert-driven variant of Gebauer-Möller

The idea of Algorithm 5 is the following: if $m_t = 0$ for all t, then $\mathcal{G}$ is a Gröbner basis of $\mathcal{I}$, see lines 27-28. Otherwise we know from Theorem 4.23 that there exists some $d' > d$ such that $m_t = 0$ for all $t < d'$ and $m_{d'} \neq 0$. This means that in order to become a d'-Gröbner basis of $\mathcal{I}, \mathcal{G}$ needs $m_{d'}$ more elements in degree d'. Thus we know

that exactly $m_{d'}$ critical pairs are useful. If we have added $m_{d'}$ elements of degree d' to $\mathcal{G}$, we can stop treating any more critical pairs of degree d'. We check the value of m_t before a new critical pair is treated for reduction (Line 14) with that goal. We only have to process critical pairs as long as $m_t > 0$, otherwise we are free to remove all remaining critical pairs of degree d'. After adding those $m_{d'}$ elements to $\mathcal{G}$ we recompute $\mathsf{HF}(P/\,\mathsf{LT}_{\succ}(\mathcal{G}), t)$ and we proceed with the next higher degree.

Unfortunately, this idea cannot be directly applied to inhomogeneous ideals. However, if one has already computed a Gröbner basis $\mathcal{G}_2$ of $\mathcal{I}$ with respect to another term ordering $\succ_2$ on P, then the ideas of Corollary 4.22 can be applied. Another suggestion is to homogenize the generators of $\mathcal{I}$, compute the Gröbner basis $\mathcal{G}^h$ of the homogenized input as mentioned before, and dehomogenize $\mathcal{G}^h$ in the end to receive the requested Gröbner basis $(\mathcal{G}^h)^{deh}$. Other minor ideas concerning the use of the Hilbert Series, can be found in Chapter 5 of [Traverso, 1996], and in [Kreuzer and Robbiano, 2000].

Hilbert Series and the Signature-based Strategy

Now, we shall see how to incorporate this strategy into signature-based algorithms. Our implementation is based on the footprints of the algorithm of Gao et al. [2010b]. Thereby, the output of the algorithm not only returns a Gröbner basis of the ideal $\mathcal{I}$, but also a Gröbner basis of the leading terms of the syzygy module of the same ideal.

The idea of Algorithm 6 is the following. Let α_k and β_k be two variables used to keep track of the number of generators of degree k in the Gröbner basis and in the leading terms of the syzygy module, which shall then used to avoid unnecessary computations of the Hilbert Series. Then, if the equality $\mathsf{HF}(\mathrm{P}/\,\mathsf{LT}_{\succ}(\mathcal{G}), k) = \mathsf{HF}(k) - \mathsf{HF}(\mathrm{P}/H, k-d)$ holds for all k, we have that $\mathcal{G}$ is a Gröbner basis of $\mathcal{I}$ and H is a Gröbner basis for the leading terms of the syzygy module. See lines 28-29. Otherwise, we know that there exists a minimal degree d for each we need to compute more generators. That degree d also gives us a bound for the minimal degree of the S-polynomials that we need to compute.

Next we give the pseudo-code of the algorithm that puts together all the ideas discussed. The reader should keep in mind that this strategy highly depends on the order in which the J-pairs are processed. If the first J-pairs are the useful ones, the optimization is best. If those are at the end of the list of pairs to be reduced, then we still have to compute the zero reductions of all useless pairs before them.

Input:
An ideal $\mathfrak{I} = \langle g_1, \ldots, g_m \rangle \in \mathrm{P} = K[x_1, \ldots, x_n]$,
A term ordering for P, and a term ordering on P^m
$\mathsf{HF}(w) := \mathsf{HF}(\mathbf{P}/\mathfrak{I}, w)$ **the Hilbert function of** $\mathfrak{I}$ **(Optional)**
Output:
A Gröbner basis V for $\mathfrak{I}$ and,
A Gröbner basis H for $\mathsf{LT}_\succ(\mathbf{H})$, the leading terms of the syzygy module
Variables:
U a list of terms T_i, representing sig. of $(\mathbf{u}_i, v_i) \in M$,
V a list of polynomials v_i for $(\mathbf{u}_i, v_i) \in M$,
H a list of $\mathsf{LT}_\succ(\mathbf{u})$ were $\mathbf{u} \in \mathrm{P}^m$ is a syzygy found so far,
JP a list of pairs (t, i) s.t. $t(u_i, v_i)$ is the J-pair of $(\mathbf{u}_i, v_i)$ and $(\mathbf{u}_j, v_j)$ for some $j \neq i$.
Algorithm:

```
1  U = {e_1, ..., e_m} and V = {g_1, ..., g_m};
2  Add the leading terms of the principle syzygies g_j e_i − g_i e_j for 1 ≤ i < j ≤ m to H ;
3  Compute all the J-pairs of (e_1, g_1), ..., (e_m, g_m);
4  Add to the list JP all J-pairs whose signatures are distinct and not reducible by H ;
5  Initialize d with the minimal degree in JP  ;
6  while JP is not empty do
7      Take a minimal pair (t, i) from JP (with respect to signature) ;
8      Delete the pair (t, i) from the list JP ;
9      while t(T_i, v_i) is regular top-reducible by the pairs in (U, V) do
10         Perform regular top-reductions if the degree is ≥ d, say to get (T, v)
           of degree k. Else Goto 6 ;
11     end
12     if v = 0 then
13         Append T to H ;
14         Delete every J-pair (t, j) in JP whose signature tT_j is divisible by T ;
15         β_k ← β_k + 1 ;
16     end
17     if v ≠ 0 and (T, v) is not super top-reducible by (U, V) then
18         Append T to U and v to V;
19         Form all J-pairs for (T, v) and (T_j, v_j), 1 ≤ j ≤ |U| − 1, and ;
20         Consider only J-pairs whose signatures are not reducible by H, and ;
21         Add to JP J-pairs with minimal LT_≻(v) for each distinct signature T ;
22         Add LTs of the principle syzygies, vT_j − v_j T for 1 ≤ j ≤ |U| − 1, to H ;
23         Update β_k ;
24         α_k ← α_k + 1 ;
25     end
26     if HF(P/LT_≻(V), w) = HF(w) − HF(P/H, w − d) for all w  then
27         return V and H;
28     end
29     d ← min{w ∈ ℕ | HF(P/LT_≻(V), w) ≠ HF(w) − HF(P/H, w − d)} ;
30 end
31 return V and H;
```

Algorithm 6: Signature-based algorithm using Hilbert Series

4.5 Modular and Parallelization Strategies

Technology is constantly changing at a fast pace. With the amount of modern multi-core and multiprocessor computers available these days, the question of parallelization of signature-based algorithms comes up quite naturally. There are two different strategies to optimize an algorithm at instruction level: modular and parallelization based strategies.

Introduction to Modular Algorithms

Modular algorithms have the following footprint. First, we have to find a "lucky prime" with high probability, that is, a prime number that does not lead to a great lost of algebraic information; secondly, we compute the object modulo a prime number and we "lift" the coefficients to the integers or rationals; and finally, we check the correctness of result.

Hence, we start by recalling the definition of "lucky" prime. Let $\mathrm{P} = \mathbb{Q}[x_1, \ldots, x_n]$ be the polynomial ring defined over the rationals, and $\succ$ a fixed term ordering in P.

Definition 4.25. Let $\mathcal{G}$ a Gröbner basis of $\mathcal{I} = \langle f_1, \ldots, f_r \rangle$, with respect to some term ordering $\succ$ in P. Let $p \in \mathbb{N}$ be prime number that does not divide the denominator of any coefficient of f_i for $1 \leq i \leq r$.

1. The ideal $\mathcal{I}$ modulo p, corresponds to $\mathcal{I}_p = \langle f_1 + p\mathbb{Z}, \ldots, f_r + p\mathbb{Z} \rangle \subset \mathbb{Z}_p[x_1, \ldots, x_n]$.
2. The set $\mathcal{G}_p \subset \mathbb{Z}_p[x_1, \ldots, x_n]$ denotes the Gröbner basis of $\mathcal{I}_p$.
3. p is said *lucky* for $\mathcal{I}$ if and only if $\mathsf{LM}_{\succ}(\mathcal{G}_p) = \mathsf{LM}_{\succ}(\mathcal{G})$.
4. p is *Hilbert-lucky* for $\mathcal{I}$ if and only if $\mathsf{HF}(\mathcal{I}) = \mathsf{HF}(\mathcal{I}_p)$.

Lemma 4.26. *Let p be a prime number, and $\mathcal{I}$ an ideal in P. Then for any degree d, it holds that*

$$\mathsf{HF}(\mathcal{I}, d) \leq \mathsf{HF}(\mathcal{I}_p, d)$$

Proof. See, the proof of Theorem 5.3 in [Arnold, 2003]. □

Next, we define the m-Farey rational map, which can be used to recover the rational coefficients of $\mathcal{G}$ from the $\mathbb{Z}_{p^i}$ coefficients of $\mathcal{G}_{p^i}$ [Kornerup and Gregory, 1983].

Definition 4.27. Let $m > 0$ be an integer and $p \in \mathbb{N}$ a prime number. The *m-Farey rational map* ϕ_m is defined by

$$\begin{array}{rcccc} \phi_m & : & F_m & \longrightarrow & \mathbb{Z}_p \\ & & \frac{a}{b} & \mapsto & (a + m\mathbb{Z})(b + m\mathbb{Z})^{-1} \end{array}$$

where F_m is the set defined by

$$F_m := \left\{ \frac{a}{b} \mid \gcd(a,b) = 1, 0 \leq a \leq m, 0 < |b| \leq m \right\}.$$

Additionally, under certain conditions, we have that the m-Farey rational map is bijective.

Proposition 4.28. *Let m be an integer greater than zero, and $p \in \mathbb{N}$ a prime number. The m-Farey rational map $\phi_m : F_m \longrightarrow \mathbb{Z}_p$ is bijective if and only if m is the largest integer satisfying $m \leq \sqrt{\frac{p-1}{2}}$.*

Proof. For a proof, see for example, Kornerup and Gregory [1983]. □

Bellow, we describe the basic structure of the modular algorithm to compute a Gröbner basis.

```
   Input: An ideal 𝔍 ⊂ P
   Output: A Gröbner basis 𝔊 for 𝔍 = ⟨f_1, ..., f_r⟩
   Algorithm:
 1 𝔊 ← ∅;
 2 B ← {p prime numbers | p chosen heuristically};
 3 while true do
 4 |   while B ≠ ∅ do
 5 |   |   Choose p from B;
 6 |   |   B ← B\{p};
 7 |   |   𝔊_p ← GrobnerBasis(𝔍_p);
 8 |   |   𝔊 ← 𝔊 ∪ {𝔊_p};
 9 |   end
10 |   RemoveNotLucky(𝔊);
11 |   𝔊 ← Lift(𝔊);
12 |   if Test(𝔊, 𝔍, Q) then
13 |   |   return 𝔊;
14 |   end
15 |   B ← B ∪ { p prime numbers | p ∉ B and p chosen heuristically };
16 end
17 return 𝔊;
```

Algorithm 7: Modular Gröbner basis algorithm (ModGB)

The Algorithm 7 can be translated into the following textual instructions:

1. First, we define a set of prime numbers B, such that, $\forall_{p \in B}$, p does not divide the denominator of any coefficient of the elements f_i. See, line 2.
2. Then, we compute a Gröbner basis for every $\mathfrak{J}_p$ in $\mathbb{Z}_p[x_1, \ldots, x_n]$ such that $p \in B$, and we store it in $\mathfrak{G}$. See, lines 5-8.

3. Next, we have to identify those $\mathcal{G}_p$ whose p's are clearly not lucky for $\mathcal{I}$. Unfortunately, our goal is to compute $\mathcal{G}$. Hence, $\mathcal{G}$ is not known beforehand, and therefore we cannot use the Definition 4.25. Instead, we need a criteria to identify lucky primes with high probability, out of $\mathcal{G}$. One solution is to build sets S_p using the algorithm described by Arnold [2003]. First, we take an element p out of B and we define the set S_p:

$$S_p := \{q \in B \mid \mathsf{LM}_{\succ}(\mathcal{G}_p) = \mathsf{LM}_{\succ}(\mathcal{G}_q)\}.$$

Then, we choose the first element $p' \in B$ that is not in the set S_p, and we build the set $S_{p'}$ analogously to S_p. We repeat this process until all elements of B are added to exactly one set S_p. Then we keep in $\mathcal{G}$ only those Gröbner bases $\mathcal{G}_p$, whose index prime is in the set $S_{p_0} \in S$, where S is the set that contains all S_p, and

$$\#(S_{p_0}) \geq \#(S_p) \text{ for all } \mathrm{S_p} \in \mathrm{S}.$$

Once we remove all those $\mathcal{G}_p$ whose p are not lucky for $\mathcal{I}$, we get the Gröbner bases corresponding to lucky primes for $\mathcal{I}$ with a high probability. This step corresponds to line 10, RemoveNotLucky.

4. Let us assume that, after Step 4, we obtain $\mathcal{G} = \{\mathcal{G}_{p_1}, \ldots, \mathcal{G}_{p_s}\}$. The next step, is to lift the results. Let W be the polynomial ring $\mathbb{Z}_r[x_1, \ldots, x_n]$ over $\mathbb{Z}$, where $r = \prod_{i=1}^{s} p_i$. Once we use the Chinese Remainder Theorem, we get a Gröbner basis $\mathcal{G}_r \subset W$.

$$\begin{array}{ccccccc} \mathbb{Z}_{p_1}[x_1, \ldots, x_n] & \times & \ldots & \times & \mathbb{Z}_{p_s}[x_1, \ldots, x_n] & \longrightarrow & \mathbb{Z}_n[x_1, \ldots, x_n] \\ \mathcal{G}_{p_1} & \times & \ldots & \times & \mathcal{G}_{p_s} & \mapsto & \mathcal{G}_r \end{array}$$

Lastly, we use the Farey rational map ϕ_k to get back to P.

5. Now, we must check that $\mathcal{G}$ is really a Gröbner basis for $\mathcal{I}$ with respect to $\succ$. Otherwise, we cannot ensure that we have enough modular Gröbner bases $\mathcal{G}_p$. Nevertheless, there exists an upper bound for the number of primes that has to be considered. If $\mathcal{G}$ is a Gröbner basis for $\mathcal{I}$ then the number of primes $p \in B$ is enough if

$$\sum_{p \in B} p \geq \max\{2 \cdot |c|^2 \mid c \text{ any coefficient of an element } g \in \mathcal{G}\}$$

Unfortunately, we do not know $\mathcal{G}$ beforehand. Therefore, we have to test if the set $\mathcal{G}$, which is constructed in Step 4, is the requested Gröbner basis. Consequently, $\mathcal{G} = \{g_1, \ldots, g_t\}$ has to pass three different tests. See function Test, Line 12.

First, we choose a prime number $q \notin B$ that does not divide the numerator or the denominator of any coefficient of the generating polynomials f_i for $\mathcal{I}$. We consider the first test passed if $\{g_1 + q\mathbb{Z}, \ldots, g_t + q\mathbb{Z}\}$ is a Gröbner basis for $\mathcal{I}_q$. However, this is not a sufficient condition for checking if $\mathcal{G}$ is a Gröbner basis for $\mathcal{I}$. If $\mathcal{G}$ fails the test, then we do not have enough modular Gröbner basis computed. Hence, we have to compute more; as a second test, we have to check if $\mathcal{I} \subset \langle g_1, \ldots, g_t \rangle$; lastly, we check if $\mathcal{G}$ is a Gröbner basis for $\langle g_1, \ldots, g_t \rangle$. Sadly, this test that must be computed in P. Therefore this test can be very expensive if we have not considered enough modular Gröbner basis $\mathcal{G}_p$.

If $\mathcal{G}$ passes all the tests, then $\mathcal{G}$ is the Gröbner basis for $\mathcal{I}$ with respect to $\succ$, and the algorithm terminates. Otherwise, we have to compute more modular Gröbner basis.

Clearly, the Algorithm 7 is not optimized, it focuses only on the general idea. In a real implementation one can reuse the already computed Gröbner basis $\mathcal{G}_p$, and the Gröbner basis $\mathcal{G}_r$, already lifted by the Chinese Remainder Theorem. Thus, if the test fails, we only have to compute those new $\mathcal{G}_q$.

Having understood this algorithm, the idea of combining it with signature-based algorithms, comes quite easy. The ModGB algorithm does not impose any restrictions on the implementation of GrobnerBasis. Hence, it can be implemented using the signature-based strategy. Next, we shall see a further optimization to ModGB.

Introduction to Parallel Algorithms

A parallel algorithm, as opposed to the traditional sequential one, can execute several operations at the same time. However, the parallelization is only possible if the tasks being parallelized do not have computational dependencies between them. Some of the instructions in ModGB that can be parallelized are the computation of a modular Gröbner basis, and the tests that are required during the computation:

1. Test whether $\mathcal{G}_q := \{g + q\mathbb{Z} \mid g \in \mathcal{G}\}$ is a Gröbner basis for $\mathcal{I}_q$, for some prime $q \in B$. For this, we have to show that $f_i + q\mathbb{Z} \in \langle \mathcal{G}_q \rangle$ and $\mathcal{G}_\mathrm{q} \subseteq \mathrm{GrobnerBasis}(\mathcal{I}_\mathrm{p})$,
2. Test if $\mathcal{I} \subseteq \langle \mathcal{G} \rangle$. That is, $\mathcal{I} \subseteq \langle \mathcal{G} \rangle \Leftrightarrow f_i \in \langle \mathcal{G} \rangle$ for all f_i generating $\mathcal{I}$,
3. Test if $\mathcal{G}$ is a Gröbner basis for $\langle \mathcal{G} \rangle$. Check if all S-polynomials not detected by the Buchberger criteria reduce to zero with respect to $\mathcal{G}$.

Clearly, this parallelization pattern is based on the fact that all parallelized computations complete in a similar timespan. The very same holds for the inclusion checks

for the generators of $\mathcal{I}$ and the S-polynomials. Although, the modular Gröbner basis computations of the $\mathcal{G}_p$ can be parallelized easily using one process per computation, one needs to use multiple threads doing the parallelized tests. Otherwise the overhead of sending and receiving data from one process to the other takes longer than the complete reduction itself. Next, we extend the idea behind the ModGB algorithm.

The Gröbner trace algorithm

In 1988, Traverso presented the so-called *Gröbner trace algorithm* [Traverso, 1989]. On the one hand, many see this algorithm as the origin of the already presented ModGB because the idea of lucky prime numbers and modular attempts are noted first. On the other hand, the Gröbner trace algorithm is a more aggressive implementation of those ideas. Whereas ModGB uses only the idea of finding lucky primes p modulo whose the Gröbner basis computations are done independently, the Gröbner trace algorithm enforces upon all modular computations the same trace.

Definition 4.29. Let $\mathcal{I} = \langle f_1, \ldots, f_r \rangle$ be an ideal and $\succ$ a fixed term ordering in P. The *Gröbner trace* $T(m, S, n, \lambda)$ is defined during the computation of a Gröbner basis $\mathcal{G} = \{g_1, \ldots, g_s\}$ of $\mathcal{I}$ when $f_i = g_i$ for $i \in \{1, \ldots, r\}$, such that

1. m is a finite sequence of the leading terms of $\mathcal{G} : m = (m_1, \ldots, m_s)$,
2. S is a finite sequence of all generated critical pairs: $S = (S_{r+1}, \ldots, S_s)$,
3. n is a finite sequence of finite sequences of integers $n_{j,k} : n = (n_{r+1}, \ldots, n_s)$ such that $n_j = (n_{j,1}, \ldots, n_{j,k_j})$ where $n_{j,k} < j$ for all $j \in \{r+1, \ldots, s\}$. Each $n_{j,k}$ represents the index of the reducer in $\mathcal{G}$ for the *k-th* reduction of the the *j-th* element.
4. λ is a finite sequence of finite sequences of terms $\lambda_{j,k} : \lambda = (\lambda_{r+1}, \ldots, \lambda_s)$ such that $\lambda_j = (\lambda_{j,1}, \ldots, \lambda_{j,k_j})$ for all $j \in \{r+1, \ldots, s\}$. Analogously, this also means that $\lambda_{j,k}$ is the corresponding multiplier for this reduction step.

The main idea of the Gröbner trace algorithm is to store all the essential information related to the computation of a Gröbner basis $\mathcal{G}$, in the Gröbner trace. This includes the leading term of each element in $\mathcal{G}$, which is stored in m, and all computed S-polynomial, which are stored in S. Then reduction steps can be uniquely determined by n and λ.

Assume we have to compute a Gröbner basis $\mathcal{G}$ of an ideal $\mathcal{I}$, but we already have a Gröbner trace T, which could come from another Gröbner basis computation of $\mathcal{I}$. Then $\mathcal{G}$ can be computed through the following algorithm,

Input:
An ideal $\mathfrak{I} = \{f_1, \ldots, f_r\} \subset \mathrm{P}$, and an ordering $\succ$ on P
A Gröbner trace $T = (m, S, n, \lambda)$ of the ideal $\mathfrak{I}$
Output:
$\mathfrak{G}$ a set of polynomials including $\{f_1, \ldots, f_r\}$,
R and E two integer values
Algorithm:

```
1  G ← {f_1, ..., f_r} ;
2  R ← 0, E ← 0 ;
3  for i = r + 1, ..., s do
4      h ← S_i ;
5      for j = 1, ..., j_i do
6          if LT_≻(f) = λ_{i,j} LT_≻(g_{n_{i,j}}) then
7              f ← LC_≻(g_{n_ij}) f − LC_≻(f) λ_{i,j} g_{n_{i,j}};
8          end
9          else if LT_≻(f) < λ_{i,j} LT_≻(g_{n_{i,j}}) then
10             R ← 1;
11         end
12         else
13             E ← 1;
14             return (G, R, E);
15         end
16     end
17     if LT_≻(f) = m_i then
18         g_i ← f;
19         G ← G ∪ {g_i};
20     end
21     else if LT_≻(f) > m_i then
22         E ← 1;
23     end
24     else
25         E ← 2;
26         return (G, R, E);
27     end
28 end
29 return (G, R, E);
```

Algorithm 8: GBTrace - Gröbner trace algorithm

The important idea to be retained is that we only have to compute reduction steps. Everything else is already provided by the Gröbner trace. That is,

1. We choose S-polynomials from $T(m, S, n, \lambda)$, see line 4.
2. We choose, see line 6, the corresponding reducers from T, which are identified by $n_{i,j}$ and $\lambda_{i,j}$.
3. Since the whole set of S-polynomials to be investigated is already given by T in S, this means that we do not compute new critical pairs after line 19.

This algorithm has several advantages when compared to a usual Gröbner basis

computation: it gives the advantage that we do not have to search for any element, and we do not have to use any criteria to check the existence of useless critical pairs. However, it cannot react to changes or unforeseen steps. Thus we have to extend it with two new boolean variables R and E that shall keep track of any problems happening during the computations.

1. We set the variable R to 1 if the leading term of f is lower than expected (Line 9). Hence we are not required to interrupt the computations, since it is might happen that $\mathsf{LT}_\succ(f)$ is equal to the leading term of the next reducer pair stored in T. So even if a redundancy takes place, $\mathsf{LT}_\succ(g_i) = t_i$ can still be fulfilled in Line 17.
2. We set the variable E to 1 or 2 whenever we get an error or not. This happens only if
   (a) If $\mathsf{LT}_\succ(f) > \lambda_{i,j}\,\mathsf{LT}_\succ(g_{n_i,j})$, then the computation cannot go on from this point (Line 12). All following reducers, generated by the lists n_i and λ_i in the Gröbner trace T, have a leading term smaller than $\lambda_{i,j}\,\mathsf{LT}_\succ(g_{n_i,j})$, thus no further reduction for f takes place. At this point the algorithm returns the already computed set $\mathcal{G}$ and marks the error with $E = 1$.
   (b) If $\mathsf{LT}_\succ(f) \neq t_i$ (see Line 21), then we are required to terminate the algorithm with an error, as the following S-polynomials in $S \in T$ would be no longer valid. Here we distinguish between two possible errors: If $\mathsf{LT}_\succ(f) > t_i$ then $E = 1$, otherwise $E = 2$. The motivation behind such distinction is explained after we discuss the algorithm TraceModGB.

Traverso presented several approaches on how to use GBTrace in Gröbner basis computations in [Traverso, 1989]. We will restrict ourselves to the one we are interested most, the modular Gröbner trace computation. The Algorithm 9 (see next page), which contains the pseudo-code for a modular Gröbner trace computation, should also be of clear understanding to the reader, since it is similar to the one already presented, ModGB. The main differences are:

1. We compute a first modular Gröbner basis modulo the prime number p_0 (see Line 5). The function TGB denotes a Gröbner basis algorithm that stores all necessary data for the corresponding Gröbner trace T.
2. We proceed with other modular computations (see Line 11). However, we only have to perform normal Gröbner basis computations. The GBTrace will be used

```
   Input:
     An ideal 𝔍 = {f_1, ..., f_r} ⊂ P, an ordering ≻ on P
   Output:
     A Gröbner basis 𝔊 of 𝔍 w.r.t ≻
   Algorithm:
 1 𝔊 ← ∅, R ← 0, E ← 0, b ← 1;
 2 B ← {p prime numbers | p chosen heuristically} ;
 3 Choose p_0 from B.;
 4 B ← B\{p_0};
 5 (𝔊_{p_0}, T) ← TGB(𝔍_{p_0});
 6 𝔊 ← 𝔊 ∪ {𝔊_{p_0}};
 7 while b = 1 do
 8     while B ≠ ∅ do
 9         Choose p from B;
10         B ← B\{p};
11         (𝔊_p, R, E) ← GBTrace(𝔍_p, T);
12         if E = 0 then
13             𝔊 ← 𝔊 ∪ {𝔊_p}
14         end
15         else if E = 1 then
16             𝔊 ← ∅;
17             (𝔊_p, T) ← TGB(𝔍_p, NF);
18         end
19     end
20     RemoveNotLucky(𝔊);
21     𝔊 ← Lift(𝔊);
22     if Test(𝔊, 𝔍, B) then
23         return 𝔊;
24     end
25     R ← {p prime numbers | p ∉ Q and p chosen heuristically};
26     B ← B ∪ R;
27 end
```

Algorithm 9: TraceModGB - Modular Gröbner trace algorithm

to compute the corresponding sets $\mathcal{G}_p$, which will speed up the computations by large factors.

3. For every error reported from GBTrace, we have to identify how to proceed:

   (a) If $E = 1$ (Line 15), then at some point the leading term of an element computed for $\mathcal{G}_p$ is greater than the corresponding one stored in T. So we must assume that p is lucky and all beforehand used primes were not lucky. At this point we delete all previously computed Gröbner basis from $\mathcal{G}$ (Line 16) and we compute a new Gröbner trace using the lucky prime number p.

   (b) If $E = 2$ we just discard the computed modular Gröbner basis $\mathcal{G}_p$ and we proceed with the remaining computations. We hope for the previous prime

numbers to be lucky and p not lucky.

Clearly, we can add $\mathcal{G}_p$ to $\mathcal{G}$ only if no errors are reported. Then we can go on with the next prime number.

It is clear that one has to think about how to recover T in Line 20, possibly without a complete Gröbner basis computation, which leave us with lot of space for new optimizations. Also, the balance between computing and testing shall be investigated in more detail to receive a better performance. Moreover, Joux and Vitse [2011] showed the potential of this theory by trying to combine together the ideas of tracing with the improved reduction process of F4. Like ModGB, this strategy can be combined with signature-based algorithms.

Parallelization of Signature-based Algorithms

The structure of signature-based algorithms can be parallelized "easily" if a Gröbner basis is computed incrementally. Some strategies that can be implemented to accomplish such goal are the following.

1. Once a Gröbner basis for $\langle f_1, \ldots, f_i \rangle$ is computed one could start several calls of the next incremental step with different initial input elements, say one computes a Gröbner basis for $\langle f_1, \ldots, f_i, f_j \rangle$, the other for $\langle f_1, \ldots, f_i, f_k \rangle$, and so on. Then, we choose the one that finishes first. We repeat this step recursively.
2. A more sophisticated approach is to completely divide the computations and merge them back together step by step. Assume we want to compute a Gröbner basis for $\mathcal{I} = \langle f_1, \ldots, f_r \rangle \subset P$.
   (a) Then one could compute the ceiling $k = \lceil \frac{r}{2} \rceil$ and start the computation of Gröbner basis for $\langle f_1, \ldots, f_k \rangle$ and $\langle f_{k+1}, ..., f_r \rangle$. Depending on the number of available processors unities. We repeat this step recursively.
   (b) As a result, we get a couple of Gröbner basis $\mathcal{G}_1, \ldots, \mathcal{G}_m$ where m denotes number of processors available. The next step is to merge them together, i.e. we can, in parallel compute Gröbner basis for $\mathcal{G}_{1,2} = \mathcal{G}_1 \cup \mathcal{G}_2, \ldots, \mathcal{G}_{m-1,m} = \mathcal{G}_{m-1} \cup \mathcal{G}_m$.
   (c) After all the recursive steps, we will eventually get a Gröbner basis $\mathcal{G}$ for $\mathcal{I}$.

Although, these two ideas can be combined together, the main problem concerns their incremental structure. The Gröbner basis $\mathcal{G}_i$ tend to have lots of elements, thus the algorithm has to process lots of merging steps.

Another way of parallelizing signature-based computations is to parallelize the code using different threads in a single process. Hence, the goal is not about the parallelization of the whole Gröbner basis computation. Instead, the idea is to parallelize only some specific parts such as:

1. Critical pair generation.
2. Criteria checks.
3. Reduction process.
4. Parallelization of multiplication and division of polynomials[1].

Although our aim is to implement these strategies on signature-based algorithms, nearly all of these ideas can or are already applied to standard algorithms for computing a Gröbner basis. Next, we shall see which operations of the former algorithm can be computed in parallel.

1. Line 4, 20, 21 - The criterion that allow us to remove J-Pairs that are reducible by the leading term of the syzygy module, can be computed simultaneously for each J-Pair in JP. Hence, the time complexity is reduced from $O(n)$ to a theoretical $O(1)$. Additionally, we can filter and sort the J-Pairs that have distinct signatures in $O(n \log^2(n))$ instead of $O(n \log n)$.

2. Line 9, 10, 14, 17, 21 - The criterion to verify if the signature is redundant can also be parallelized, as well as the subsequent reduction process. Although, we did not test any specific strategy for reducers, their implementation would have smaller penalizations under this implementation, since the comparison and search time is minimal. The same idea is used in the internal reduction process. If there is a reducer, then it can be found in $O(1)$ time.

3. Line 22 - The computation of the principle syzygies can also be parallelized. Again, the time complexity of the whole operation is reduced from $O(n)$ to a theoretical $O(1)$ time.

[1]The reader should be aware that this task can be really complex and may require several implementation workarounds to be addressed.

4.6 Technical Implementation Overview

The aim of this section is to explain some of the computational tricks that were used to increase the efficiency of our algorithms. Briefly, we describe the strategy that is used to choose J-Pairs, some of the arithmetic simplifications, and an overview of the data-structure used to store the list of J-Pairs.

J-Pair Selection Strategy

We have seen that in signature-based strategies we only have to retain a J-pair for each signature. However, if we have multiple J-pairs with the same signature, then we have to choose the one we want to keep. Some of the possible criteria are:

1. The most recent J-pair encountered with each signature.
2. The J-pair $t_i(\mathbf{u}_i, v_i)$ with the smallest or largest index i.
3. The J-pair $t(\mathbf{u}, v)$ with the largest scale t.
4. The J-pair with a v that has the most/fewest terms.

In our algorithms, we have decided to implement the strategy that keeps the first J-pair encountered. Hence, we are not required to update the search graph, which has to be maintained during the computation of the Gröbner basis. Yet, this topic deserves more attention and further benchmarks.

Simplification of Arithmetic Operations

Our algorithms implement also the arithmetic optimization suggested by Gao et al. [2010a]. The idea is to make all pairs $(\mathbf{u}, v)$ monic, such that the leading coefficient of $\mathbf{u}$ will be equal to 1. Suppose $(\mathbf{u}_1, v_1)$ and $(\mathbf{u}_2, v_2)$ are any two monic pairs. Then a top-reduction would be determined only by $\mathsf{LT}_{\succ}(\mathbf{u}_i)$ and v_i where $i \in \{1, 2\}$. Any other terms of $\mathbf{u}_1$ and $\mathbf{u}_2$ are not necessary at all.

Search Graph Implementation

The fastest known algorithm for sorting a list of n elements has complexity time of $O(n\ log\ n)$. Merge sort belongs to the list of such algorithms, but with the advantage that it is easy to understand and implement. Given a list of n elements, the merge sort algorithm splits the list into two sublists of roughly the same size, and sorts each of them separately. The number of recursions, that is, the amount splits required to

reach the trivial case list is $O(log\ n)$. Each level of the recursion requires $O(n)$ work to split and assemble all lists, which makes it an $O(n\ log\ n)$ algorithm.

We generalize this sorting algorithm by splitting a list of length kt into k sublists of roughly equal size. Each sublist of length t can be sorted ascendingly in $O(t\ log\ t)$ time, so that all k sublists can be sorted in $O(kt\ log\ t)$. Assume that we have k sorted lists. We say that list L_1 is smaller than list L_2 if L_1's first element is smaller than L_2's first element. With this $O(1)$ comparison defined for all k lists, we insert them into a minimum priority queue in $O(k)$ time.

Consider now a typical priority queue, then the dequeue operation would return the smallest of the k lists in $O(log\ k)$ time. Our implementation can give us the smallest element in $O(1)$ time. However, after a dequeue operation, the smallest list may no longer be the smallest within the queue, so we must update its position for a total of $O(log\ k)$ time. Since there are kt total elements within all the lists contained in the priority queue, it takes kt dequeue operations to sort all kt elements, thus taking $O(kt\ log\ k)$ time. Therefore this version of merge sort runs in $O(kt\ log\ t\ +\ kt\ log\ k) = O(kt\ log(kt))$ time, which is exactly the time that we initially claimed.

Chapter 5

Experimental Results

If you think dogs cannot count,
try putting three dog biscuits in your pocket and
then giving Fido only two of them.
(Phil Pastoret)

In this chapter, we shall benchmark some of the fastest algorithms to compute a Gröbner basis. In order to assess their performance, we consider a set of examples that is commonly used to benchmark Gröbner basis algorithms. For more information about the examples that are used, see the Appendix.

We have implemented both the GVW algorithm and new criteria in CoCoALib [2013], and observed that both provide reasonable results. In order to ease the identification of our algorithm against other algorithms, we label it as *New Alg.* Also, the reader should keep in mind that this algorithm returns more than just a Gröbner basis for the input ideal. All tests were run on a Mac OS X with a 2.6 GHZ Intel Core i7 processor and 8GB DDR3 of memory.

Our first benchmark compares the computation times of a Gröbner basis over a field $\mathbb{F}_{32003}$. See results in Table 5.1. There are two conclusions that can be drawn from this benchmark. First, the GVW algorithm with our criteria is in general faster than the original algorithm. Moreover, in some cases the gains can be up to six times faster than GGV. Finally, we can see that there is no clear winner. However, at comparable times, we have that our algorithm provides a Gröbner basis for the input ideal, and a Gröbner basis for the leading terms of the syzygy module of the ideal.

System	F5	F5E	GGV	GVW	New Alg.	CoCoALib
Katsura 9	14.98	14.87	17.63	14.95	15.35	15.12
Katsura 10	153.35	152.39	192.20	152.32	165.32	142.21
Eco 8	2.24	0.38	0.49	0.69	0.35	0.44
Eco 9	77.13	8.19	13.51	8.23	7.21	13.26
F744	19.35	8.79	26.86	10.65	8.69	8.11
Cyclic 7	7.01	7.22	33.85	9.12	7.12	7.78
Cyclic 8	7310.39	4961.58	26242.12	5961	6267.58	4410.39

Table 5.1: Timings in $\mathbb{F}_{32003}$ in seconds

Another way to compare the algorithm's performance is to use a computer independent measure, were we take into consideration the number of J-pairs processed and the number of J-pairs reducing to zero; therefore not producing any useful data. The data belonging to this measure can also be used to relate the amount of time required to compute the Gröbner basis (see Table 5.1) with the number of zero reductions predicted.

System	F5		F5E		GGV		GVW		New Alg.	
Katsura 9	886	0	886	0	886	0	886	0	886	0
Katsura 10	1781	0	1781	0	1781	0	1781	0	1781	0
Eco 8	830	322	565	57	2012	57	602	57	532	50
Eco 9	2087	929	1278	120	5794	120	1778	120	1572	116
F744	1324	342	1151	169	2145	169	1351	169	1091	162
Cyclic 7	1018	76	978	36	3072	36	1073	36	978	30
Cyclic 8	7066	244	5770	244	24600	244	7760	244	7592	239

Table 5.2: Number of critical pairs and zero reductions

A benchmark that might be worth checking is the benefit of using parallelization. Table 5.3 illustrates the performance between the implementation recurring to parallelization and the base implementation.

We can observe performance gains in almost all tests performed. However, we still believe that better results can be obtained if we spend more time optimizing the implementation. The slower times with F744 are a clear indication that further investigation is needed.

System	Without Parallelization	With Parallelization
Katsura 9	15.35	15.22
Katsura 10	165.32	153.11
Eco 8	0.35	0.32
Eco 9	7.21	7.16
F744	8.69	8.92
Ciclic 7	7.12	7.04
Ciclic 8	6267.58	6010.39

Table 5.3: Timings in $\mathbb{F}_{32003}$ for the parallelized algorithm

Figure 5.1 provides visual insights on how the computation time is distributed within the algorithm. We can see that the polynomial reduction is still one of the most critical parts of the algorithm (see Top-Reduction). However, is with surprise that we see that forming and maintaining the list of J-Pairs is also a very expensive process. It is important to highlight that the procedure that is used to create and update new pairs is also responsible for keeping only those signatures that are distinct. Also, it turns out that, with the right heuristics, the computation of the Hilbert Series comes at a very low price.

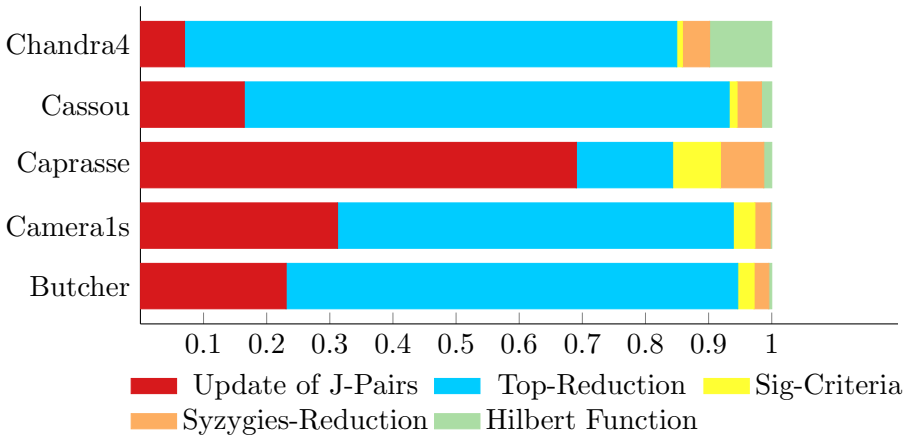

Figure 5.1: Time allocation within the algorithm

For more benchmarks see Appendix 6.

Chapter 6

Conclusions and Future Work

I am not only convinced that what I say is false, but also that what one might say against it is false. Despite this, one must begin to talk about it. In such a case the truth lies not in the middle, but rather all around, like a sack, which, with each new opinion one stuffs into it, changes its form, and becomes more and more firm.
(Albert Musil, Das Hilflose Europa)

In this thesis, we presented a wide range of strategies to improve the efficiency of the computation of a Gröbner basis. Despite any benefit that theses approaches might have, they often have drawbacks, too. For example, one has often to consider restrictions on the input. Moreover, the efficiency of these methods is highly dependent upon the behavior of the data during the computations, which cannot be known beforehand. Thus, these improvements do not provide us with a fail-proof optimization to compute Gröbner bases. To get a Gröbner basis in an efficient way, one has to implement and combine most of the presented ideas with well-designed heuristics.

The signature-based algorithms are not an exception. However, we can see that in most cases they will find more useless critical pairs than Buchberger's Criteria. Within the signature-based algorithms, performance is affected by restrictions on the reduction process and by the overhead that is generated due to how aggressive is the signature-based criteria chosen. Thus, again the question is not about getting an universal best algorithm, but even more about how to combine the signature-based world with already highly efficient improvements of the classic world without harming the performance or causing wrong results.

We developed a new strategy to signature-based algorithms built on top of previous literature about Hilbert functions. This strategy was further improved through

parallelization.

There are two easy conclusions that can be drawn from the benchmarks. First, the our algorithm is in general faster than the original algorithm - GVW, and in some cases it can be up to six times faster than GGV. Finally, we can see that there is not a clear winner, but at comparable times our algorithm provides a Gröbner basis for the input ideal and a Gröbner basis for the leading terms of the syzygy module of the ideal.

Appendices

A. Extended List of Benchmarks

	CoCoALib		GVW		New Algorithm	
Algorithm	Generators	Timings	Generators	Timings	Generators	Timings
vermeer	7	24.514	11	52.985	11	27.968
rose	7	0.264	15	1.942	15	0.661
Katsura5	6	0.073	8	0.447	8	0.179
Cyclic6	17	0.212	25	0.2311	25	0.237
6x7-4-h	525	0.771	529	4.769	529	0.806
6x7-4-hDL	525	0.796	530	0.823	530	1.63
7x8-2-h	588	0.897	590	0.871	590	0.866
boon	12	0.017	15	0.117	15	0.017
aubry2	135	29.712	145	29.993	145	29.791
butcher8	64	0.792	64	0.894	64	0.781
des18_3	104	6.652	114	7.252	114	7.142
geneig	16	0.001	18	0.002	18	0.002
hunecke	588	12.004	798	22.105	798	17.214
kinema	45	0.256	48	2.143	48	2.161
wright	31	0.002	31	0.102	31	0.051
solotarev	10	0.0	12	0.012	12	0.0
rbpl24	55	60.153	56	59.126	56	59.115
noon8	3385	1231.457	4385	1545.613	4385	1415.315
mckay	126	132778.001	146	112370.21	146	102172.127
redeco12	1536	1370.753	1536	1561.821	1536	1491.711

Table A.1. Timings and number of generators using DegRevLex

B. Description of Benchmark Examples

Example		
Boon	Description	**Title**: Neurofysiology, posted by Sjirk Boon
	References	• The system has been posted to the newsgroup sci.math.num-analysis by Sjirk Boon. • P. Van Hentenryck, D. McAllester and D. Kapur: 'Solving Polynomial Systems Using a Branch and Prune Approach' SIAM J. Numerical Analysis, Vol. 34, No. 2, pp 797-827, 1997.
Butcher	Description	**Title**: Butcher's problem

Continued on next page

Table 1 – *Continued from previous page*		
Example		
	References	• The example has been retrieved from the POSSO test suite, available by anonymous ftp from the site gauss.dm.unipi.it.
Butcher8	Description	**Title**: 8-variables version of Butcher's problem The system has 16 regular solutions. Four paths converged to highly singular solutions, which indicates that the system probably has an infinite component of solution.
	References	• W. Boege, R. Gebauer, and H. Kredel: 'Some examples for solving systems of algebraic equations by calculating Groebner bases', J. Symbolic Computation, 2:83-98, 1986. • C. Butcher: 'An application of the Runge-Kutta space'. BIT, 24, pages 425440, 1984.
Camera1s	Description	**Title**: Camera displacement between two positions, scaled first frame. This system models the displacement of a camera between two positions in a static environment, coordinates of matched points in first instance. The coordinates of the frames have been scaled, i.e., all components have been divided by 10.
	References	• Ioannis Z. Emiris. 'Sparse Elimination and Application in Kinematics'. PhD Thesis, Computer Science, University of California at Berkeley, 1994. • Ioannis Z. Emiris. 'A general Solver Based on Sparse Resultants: Numerical Issues and Kinematic Applications', INRIA Rapport de Recherche no 3110, January 1997, 29 pages. Available via anonymous ftp to ftp.inria.fr.
Caprasse	Description	**Title**: The system caprasse of the PoSSo test suite. There are 54 isolated solutions, 6 with zero components which are not counted by the mixed volume.
	References	• The PoSSo test suite

Continued on next page

Table 1 – *Continued from previous page*

Example		
Cassou	Description	**Title**: The system of Pierrette Cassou-Noguès The system is deficient with respect to face normal $(0, 0, 0, -1)$, with corresponding double component of solutions at infinity (b, c, c, e). The corresponding face system is $$\begin{aligned} -8 * b^2 * c^2 * e - 28 * b^2 * c * d * e + 36 * b^2 * d^2 * e &= 0 \\ 16 * c^2 * e^2 - 32 * c * d * e^2 + 16 * d^2 * e^2 &= 0 \\ 40 * c^2 * e^2 - 80 * c * d * e^2 + 40 * d^2 * e^2 &= 0 \\ 22 * c * e - 22 * d * e &= 0 \end{aligned}$$
	References	• T.Y. Li, Tianjun Wang, Xiaoshen Wang. 'Random Product Homotopy with Minimal BKK Bound', in: 'The Mathematics of Numerical Analysis', Edited by Renegar, J. and Shub, M. and Smale, S., Lectures in Applied Mathematics vol 32, 1996. Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics, Park City, Utah, July 17-August 11, 1995, Park City, Utah.
Chandra4	Description	**Title**: The Chandrasekhar H-Equation for n=4 and $c = 0.51234$
	References	• Laureano Gonzalez-Vega: 'Some examples on problem solving by using the symbolic viewpoint when dealing with polynomial systems of equations'. in: 'Computer Algebra in Science and Engineering', Editors: J. Fleischer, J. Grabmeier, F.W. Hehl and W. Kuechlin. Pages 102-116. World Scientific Publishing, 1995. • S. Chandrasekhar: 'Radiative Transfer', Dover, NY, 1960. • C.T. Kelley. 'Solution of the Chandrasekhar H-equation by Newton's method'. J. Math. Phys., 21 (1980), pp. 1625-1628.
Cyclic5-Cyclic9	Description	**Title**: Cyclic n-roots problem

Continued on next page

Table 1 – *Continued from previous page*

Example		
	References	• Góran Bjórck and Ralf Fróberg: 'A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots', in J. Symbolic Computation (1991) 12, pp 329336. • Backelin, J. and Froeberg, R.: 'How we proved that there are exactly 924 cyclic 7-roots' , Proceedings of ISSAC-91, pp 103-111, ACM, New York, 1991. • G. Bjórck and R. Fróberg, R.: 'Methods to "divide out" certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots ', In: Analysis, Algebra and Computers in Math. research, M. Gyllenberg and L.E.Persson eds., Lect. Notes in Applied Math. vol. 564, pp 57-70, Dekker, 1994. • J. Canny and P. Pedersen. An algorithm for the Newton resultant. Technical Report 1394, Comp. Science Dept., Cornell University, 1993. • I.Z. Emiris and J.F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. J. Symbolic Computation, 20(2):117-149, August 1995. • L. Pottier. Bounds for degree of the n-cyclic system. INRIA Sophia-Antipolis, Manuscript, 1995.
Des18-3	Description	**Title**: A 'dessin d'enfant', called des18-3 There are six real and forty complex conjugated solutions.
	References	• Raphael Nauheim. 'Systems of Algebraic Equations with Bad Reduction', Universitaet Heidelberg, Interdisziplinaeres Zentrum fuer wissenschaftliches Rechnen, Preprint 95-46, Dezember 1995. • Birch, B: 'Noncongruence Subgroups, Covers and Drawings', In 'The Grothendieck Theory of Dessins d'Enfants', editor: Schneps, L., London Mathematical Society Lecture Series, Cambridge University Press, pages 25-46, 1994.
Eco8-Eco10	Description	**Title**: n-dimensional economics problem Transform $u = 1/x - n$ and the total degree equals the number of solutions.
	References	• Alexander Morgan. 'Solving polynomial systems using continuation for engineering and scientific problems', Prentice-Hall, Englewood Cliffs, New Jersey, 1987, p. 148.
Geneig	Description	**Title**: Generalized eigenvalue problem

Continued on next page

Table 1 – *Continued from previous page*		
Example		
	References	• M. Chu, T.-Y. Li and T. Sauer, 'Homotopy method for general lambda-matrix problems', SIAM J. Matrix Anal. Appl., vol. 9, No. 4, pp 528-536.
Katsura6-Katsura10	Description	**Title**: Katsura n, a problem of magnetism in physics.
Kinema	Description	**Title**: Robot kinematics problem
	References	• Bellido, A.M.:' Construction de fonctions d'iteration pour le calcul simultane des solutions d'equations et de systemes d'equations algebriques', These 1992, Universite Paul Sabatier, Toulouse. • Anne-Mercedes Bellido: 'Construction of iteration functions for the simultaneous computation of the solutions of equations and algebraic systems'. Numerical Algorithms Vo. 6.
Noon8	Description	**Title**: A neural network modeled by an adaptive Lotka-Volterra system, $n = 3$. The coefficients have been chosen so that full permutation symmetry was obtained. The parameter $c = 1$.
	References	• Karin Gatermann. 'Symbolic solution of polynomial equation systems with symmetry', Proceedings of ISSAC-90, pp 112-119, ACM New York, 1990. • V. W. Noonburg. 'A neural network modeled by an adaptive Lotka-Volterra system', SIAM J. Appl. Math., Vol. 49, No. 6, 1779-1792, 1989. • Jan Verschelde and Ann Haegemans. 'The GBQ-Algorithm for constructing start systems of homotopies for polynomial systems, SIAM J. Numer. Anal., Vol. 30, No. 2, pp 583-594, 1993.
Rbpl24	Description	**Title**: parallel robot with 24 real solutions
	References	From the FRISCO test suite.
Redeco7-Redeco12	Description	**Title**: n-dimensional economics problem. This is the reduced economics problem ($u - n = 1/x - n$).
	References	• Alexander Morgan: 'Solving polynomial systems using continuation for engineering and scientific problems', Prentice-Hall, Englewood Cliffs, New Jersey, 1987, p. 148.

Continued on next page

Table 1 – *Continued from previous page*		
Example		
Solotarev	Description	**Title**: The system of Solotarev from the PoSSo test suite
	References	See the PoSSo test suite
Wright	Description	**Title**: System of A.H. Wright
	References	• M. Kojima and S. Mizuno. 'Computation of all solutions to a system of polynomial equations', Math. Programming, vol 25, pp 131-157, 1983. • A.H. Wright. 'Finding all solutions to a system of polynomial equations', Math. Comp., vol 44, pp 125-133, 1985. • W. Zulehner. 'A simple homotopy method for determining all isolated solutions to polynomial systems', Math. Comp., vol 50, no 161, pp 167-177.

References

Abbott, J. and Bigatti, A. M. (2013). Cocoalib: a c++ library for doing computations in commutative algebra. 4, 79

Ackermann, P. and Kreuzer, M. (2006). Gröbner basis cryptosystems. *Applicable Algebra in Engineering, Communication and Computing*, 17(3):173–194. 29

Adams, W. W. and Loustaunau, P. (1994). *An introduction to Gröbner bases*, volume 3. Amer Mathematical Society. 33

Alasdair, B. S. and de Pennington, A. (1993). Constraint definition system: a computer-algebra based approach to solving geometric-constraint problems. *Computer-Aided Design*, 25(12):741–750. 2

Allgower, E. L. and Georg, K. (1993). Continuation and path following. *Acta numerica*, 2(1):1–64. 2

Amrhein, B., Bündgen, R., and Küchlin, W. (1998). *Parallel completion techniques.*, pages 1–34. Basel: Birkhäuser Verlag. 44

Amrhein, B., Gloor, O., and Küchlin, W. (1997). On the walk. *Theoretical Computer Science*, 187(1–2):179 – 202. 40

Anantha, R., Kramer, G. A., and Crawford, R. H. (1996). Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design*, 28(9):707 – 722. 1, 29

Arnold, E. (2003). Modular algorithms for computing gröbner bases. *J. Symb. Comput.*, 35(4):403–419. 44, 66, 68

Arri, A. and Perry, J. (2011). The f5 criterion revised. *Journal of Symbolic Computation*, 46(9):1017 – 1029. 45

Atiyah, M. F. and MacDonald, I. G. (1969). *Introduction to Commutative Algebra*. Addison-Wesley, London. 5

Bayer, D. and Stillman, M. (1988). On the complexity of computing syzygies. *Journal of Symbolic Computation*, pages 135–147. 39

Becker, E., Mora, T., Marinari, M. G., and Traverso, C. (1994). The shape of the shape lemma. In *Proceedings of the international symposium on Symbolic and algebraic computation*, pages 129–133. ACM. 31

Berenstein, C. A. and Yger, A. (1990). Bounds for the degrees in the division problem. *Mich. Math. J.*, 37(1):25–43. 38

Bigatti, A., Caboara, M., and Robbiano, L. (2011). Computing inhomogeneous gröbner bases. *Journal of Symbolic Computation*, 46(5):498 – 510. 40, 42

Bigatti, A. M. (1997). Computation of Hilbert-Poincaré series. *Journal of Pure and Applied Algebra*, 119(3):237–253. 60

Borosh, I. and Fraenkel, A. S. (1966). Exact solutions of linear equations with rational coefficients by congruence techniques. *Mathematics of Computation*, 20(93):pp. 107–112. 43

Bradford, R. (1990). A parallelization of the buchberger algorithm. In *Proceedings of the international symposium on Symbolic and algebraic computation*, ISSAC '90, pages 296–, New York, NY, USA. ACM. 44

Brickenstein, M. (2004). Neue varianten zur berechnung von gröbner basens. In *Diploma thesis*. University of Kaiserslauter. 43

Brickenstein, M. (2010). Gröbner bases with slim polynomials. In *Revista Matematica Complutense*, pages 453–466. Springer. 43

Brownawell, W. D. (1987). Bounds for the degrees in the nullstellensatz. *Annals of Mathematics*, 126(3):pp. 577–591. 38

Buchberger, B. (1965). *An Algorithm for Finding the Bases Elements of the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal (German)*. Univ. of Innsbruck (Austria). 3, 22, 26

Buchberger, B. (1979). A criterion for detecting unnecessary reductions in the construction of gröbner basis. In *In Proceedings of EUROSAM'79*, pages 3–21. Springer. 40, 41

Buchberger, B. (1985). *Gröbner-Bases: An Algorithmic Method in Polynomial Ideal Theory*. Reidel Publishing Company, Dodrecht - Boston - Lancaster. 2, 40, 42

Buchmann, J., Pyshkin, A., and Weinmann, R.-P. (2006). Block ciphers sensitive to gröbner basis attacks. In *Topics in Cryptology–CT-RSA 2006*, pages 313–331. Springer. 29

Caboara, M., DeDominicis, G., and Robbiano, L. (1996). Multigraded hilbert functions and buchberger algorithm. In Lakshman, Y. N., editor, *ISSAC 96*, pages 72–78, New York. ACM Press. 40

Caboara, M., Kreuzer, M., and Robbiano, L. (2002). Minimal set of critical pairs. In Cohen, B., Gao, X., and Takayama, N., editors, *International Congress of Mathematical Software 2002 (ICMS 2002)*, pages 390–404. World Scientific. 40

Caboara, M., Kreuzer, M., and Robbiano, L. (2004). Efficiently computing minimal sets of critical pairs. In *Special Issue of Journal of Symbolic Computation for ICMS 2002 (The First International Congress of Mathematical Software)*, volume 38, pages 1169–1190. 40

Caboara, M. and Perry, J. (2012). Reducing the size and number of linear programs in a dynamic gröbner basis algorithm. *arXiv*. 41

Cambon, S., Alami, R., and Gravot, F. (2009). A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126. 1, 29

Caniglia, L., Galligo, A., and Heintz, J. (1989). Some new effectivity bounds in computational geometry. In *In Proceedings of the 6th International Conference, on Applied Algebra, Algebraic Algorithms and Error Correcting Codes*, pages 131–151. 39

Caniglia, L., Galligo, A., and Heintz, J. (1993). Equations for the projective closure and effective nullstellensatz. In *Discrete Appl. Math*, pages 11–23. 39

Chakrabarti, S. and Yelick, K. (1993). Implementing an irregular application on a distributed memory multiprocessor. In *In Proceedings of the Fourth ACM/SIGPLAN Symposium on Principles and Practices of Parallel Programming*, pages 169–178. 44

Chen, X., Li, P., Lin, L., and Wang, D. (2006). Proving geometric theorems by partitioned-parametric gröbner bases. In *Proceedings of the 5th international conference on Automated Deduction in Geometry*, ADG'04, pages 34–43, Berlin, Heidelberg. Springer-Verlag. 29

Collart, S., Kalkbrener, M., and Mall, D. (1997). Converting bases with the gröbner walk. *Journal of Symbolic Computation*, 24:465–469. 40

Collins, G. E. and Encarnación, M. J. (1994). Efficient rational number reconstruction. *Journal of Symbolic Computation*, 20:287–297. 44

Courtois, N., Klimov, E., Patarin, J., and Shamir, A. (2000). Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *In Advances in Cryptology, Eurocrypt'2000, LNCS 1807*, pages 392–407. Springer-Verlag. 42

Cox, D. A., Little, J., and O'Shea, D. (2007). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 13, 14, 29, 30

Ding, J., Buchmann, J., Mohamed, M., Moahmed, W., and Weinmann, R. (2008). Mutantxl. *Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC08)*, pages 16–22. 42

Dubé, T. W. (1990). The structure of polynomial ideals and gröbner bases. *SIAM Journal of Computation*, 19:750–773. 26, 36, 39

Ebert, G. L. (1983). Some comments on the modular approach to gröbner-bases. *SIGSAM Bull.*, 17(2):28–32. 43

Eder, C. (2008). On the criteria of the f5 algorithm. In *arXiv:0804.2033v1*. 45, 52, 54

Eder, C. (2012). Sweetening the sour taste of inhomogeneous signature-based groebner basis computations. *arXiv preprint arXiv:1203.6186*. 45

Eder, C., Gash, J., and Perry, J. (2011). Modifying faugère's f5 algorithm to ensure termination. *ACM Commun. Comput. Algebra*, 45(1/2):70–89. 45

Eder, C. and Perry, J. (2010). F5c: a variant of faugère's f5 algorithm with reduced gröbner bases. In *Journal of Symbolic Computation*, pages 1442–1458. 45, 52, 54

Eder, C. and Perry, J. E. (2011). Signature-based algorithms to compute gróbner bases. In *Proceedings of the 36th international symposium on Symbolic and algebraic computation*, ISSAC '11, pages 99–106, New York, NY, USA. ACM. 45

Eisenbud, D. (1995). *Commutative algebra with a view toward algebraic geometry*. Springer Verlag. 49, 51

Eisenbud, D. (2008). *Commutative Algebra: with a View Toward Algebraic Geometry*. Springer Verlag. 5

Essen, A. (1991). Polynomial maps and the jacobian conjecture. *Computational Aspects of Lie Groups and Related Topics*, pages 29–44. 32

Faugère, J.-C. (1999). A new efficient algorithm for computing gröbner bases (f4). In *J. Pure Appl. Algebra*, pages 61–88. 40, 42

Faugère, J.-C. (2002). A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *In Proceedings of ISSAC'02*, pages 75–82. ACM Press, New York, USA. 40, 44, 52, 53, 54, 56

Faugère, J. C., Gianni, P., Lazard, D., and Mora, T. (1993). Efficient computation of zero-dimensional groöbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344. 40

Faugère, J.-C. and Joux, A. (2003). Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. *Advances in Cryptology-CRYPTO 2003*, pages 44–60. 29

Faugère, J.-C. and Lachartre, S. (2010). Parallel gaussian elimination for gröbner bases computations in finite fields. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, pages 89–97. 44

Galkin, V. (2012). Termination of original f5. *arXiv preprint arXiv:1203.2402*. 44

Gao, S., Guan, Y., and Volny, F. I. (2010a). A new incremental algorithm for computing gröbner bases. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, ISSAC '10, pages 13–19, New York, NY, USA. ACM. 45, 52, 54, 56, 57, 76

Gao, S., Volny, F. I., and Wang, M. (2010b). A new algorithm for computing gröbner bases. In *Cryptology ePrint Archive: Report*. 40, 45, 52, 58, 64

Gao, X.-S. and Chou, S.-C. (1992). Implicitization of rational parametric equations. *Journal of Symbolic Computation*, 14(5):459–470. 29

Gebauer, R. and Möller, H. (1986). Buchberger's algorithm and staggered linear bases. In *In Proceedings of SYMSAC'86*, pages 218–221. ACM press, New York, USA. 40

Gebauer, R. and Möller, H. (1988). On an installation of buchberger's algorithm. In *Journal of Symbolic Computation*, pages 275–2861. 41

Gianni, P., Mora, F., Robbiano, L., and Traverso, C. (1994). *Hilbert functions and Buchberger algorithm.* 40

Giovini, A., Mora, T., Niesi, G., Robbiano, L., and Traverso, C. (1991). "one sugar cube, please" or selection strategies in the buchberger algorithm. In *In Proceedings of ISSAC'91*, pages 49–54. ACM Press, New York, USA. 40, 42

Giusti, M. (1985). Note on the complexity of constructing standard bases. *European Conference on Computer Algebral*, pages 411–412. 39

Giusti, M. (1987). Complexity of standard bases in projective dimension zeros. *EURO CAL*, pages 333–335. 39

Giusti, M. (1990). Complexity of standard bases in projective dimension zero ii. *AAECC*, pages 322–3280. 39

Gräbe, H.-G. and Lassner, W. (1994). A parallel gröbner factorizer. In *IN HONG*, pages 5–3. 44

Greuel, G. and Pfister, G. (2007). *A Singular Introduction to Commutative Algebra.* Springer. 60

Hashemi, A. and Ars, G. (2010). Extended f5 criteria. In *Journal of Symbolic computationt*, pages 1330–1340. 44, 52, 54

Hermann, G. (1998). The question of finitely many steps in polynomial ideal theory. *SIGSAM Bull.*, 32(3):8–30. 37

Hillyard, R. C. and Braid, I. C. (1978). Characterizing non-ideal shapes in terms of dimensions and tolerances. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '78, pages 234–238, New York, NY, USA. ACM. 2

Hoffmann, C. M. (1989). *Geometric and solid modeling: an introduction.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 1, 29

Hoffmann, C. M. (1990). A dimensionality paradigm for surface interrogations. *Comput. Aided Geom. Des.*, 7(6):517–532. 1, 29

Huang, L. (2010). A new conception for computing gröbner basis and its applications. *CoRR*, abs/1012.5425. 45, 58

Idrees, N., Pfister, G., and Steidel, S. (2011). Parallelization of modular algorithms. *J. Symb. Comput.*, 46(6):672–684. 44

Jia, X., Choi, Y.-K., Mourrain, B., and Wang, W. (2011). An algebraic approach to continuous collision detection for ellipsoids. *Computer Aided Geometric Design*, 28(3):164 – 176. 1, 29

Joux, A. and Vitse, V. (2011). A variant of the f4 algorithm. In *Proceedings of the 11th international conference on Topics in cryptology: CT-RSA 2011*, CT-RSA'11, pages 356–375, Berlin, Heidelberg. Springer-Verlag. 74

Kondo, K. (1992). Algebraic method for manipulation of dimensional relationships in geometric models. *Computer-Aided Design*, 24(3):141–147. 2

Kornerup, P. and Gregory, R. T. (1983). Mapping integers and hensel codes onto farey fractions. *BIT*, 23(1):9–20. 44, 66, 67

Kreuzer, M. and Robbiano, L. (2000). *Computational Commutative Algebra 1*. Springer. 5, 12, 15, 17, 18, 28, 31, 42, 60, 64

Kreuzer, M. and Robbiano, L. (2005). *Computational Commutative Algebra 2*. Springer, 0 edition. 1, 5, 19, 29, 62

Kreuzer, M. and Robbiano, L. (2009). *Computational Commutative Algebra 1*. Springer Verlag, 2 edition. 19, 24, 48

Lazard, D. (1983). Gröbner-bases, gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, pages 146–156, London, UK. Springer-Verlag. 40, 41

Leykin, A. (2004). On parallel computation of gröbner bases. In *ICPP Workshops*, pages 160–164. 44

Light, R. and Gossard, D. (1982). Modification of geometric models through variational geometry. *Computer-Aided Design*, 14(4):209 – 214. 2

Lin, V. C., Gossard, D. C., and Light, R. A. (1981). Variational geometry in computer-aided design. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '81, pages 171–177, New York, NY, USA. ACM. 2

Loera, J., Lee, J., Margulies, S., and Onn, S. (2009). Expressing combinatorial problems by systems of polynomial equations and hilbert's nullstellensatz. *Combinatorics, Probability & Computing*, 18(4):551. 35

Mayr, E. W. (1989). Membership in polynomial ideals over q is exponential space complete. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science on STACS 89*, pages 400–406, New York, NY, USA. Springer-Verlag New York, Inc. 38, 39

Mayr, E. W. and Meyer, A. R. (1982). The complexity of the word problem for commutative semigroups and polynomial ideals. 37

Michalik, P., Kim, D. H., and Bruderlin, B. D. (2002). Sketch- and constraint-based design of b-spline surfaces. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, SMA '02, pages 297–304, New York, NY, USA. ACM. 1, 29

Mityunin, V. and Pankratiev, E. (2007). Parallel algorithms for gröbner-basis construction. *Journal of Mathematical Sciences*, 142(4):2248–2266. 44

Mnuk, M. (2002). On an algebraic description of colorability of planar graphs. In *Logic, Mathematics and Computer Science: Interactions. Proceedings of the Symposium in Honor of Bruno Buchberger's 60th Birthday. RISC, Linz, Austria*, pages 177–186. 34, 35

Mohamed, M., Cabarcas, D., Ding, J., Buchmann, J., and Bulygin, S. (2010). Mxl 3: an efficient algorithm for computing gröbner bases of zero-dimensional ideals. *Information, Security and Cryptology–ICISC 2009*, pages 87–100. 42

Mohamed, M., Mohamed, W., Ding, J., and Buchmann, J. (2008). Mxl2: Solving polynomial equations over gf (2) using an improved mutant strategy. *Post-Quantum Cryptography*, pages 203–215. 42

Möller, H. and Mora, T. (1984). Upper and lower bounds for the degree of gröbner bases. In *EUROSAM 84*, pages 172–183. 39

Möller, H., Mora, T., and Traverso, C. (1992). Gröbner bases computation using syzygies. In *In Proceedings of ISSAC'92*, pages 320–328. ACM Press, New York, USA. 40, 41

Montes, A. and Recio, T. (2007). Automatic discovery of geometry theorems using minimal canonical comprehensive gröbner systems. In *Proceedings of the 6th international conference on Automated deduction in geometry*, ADG'06, pages 113–138, Berlin, Heidelberg. Springer-Verlag. 29

Mulmuley, K. (1986). A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, pages 338–339, New York, NY, USA. ACM. 38

Nielsen, J. and Roth, B. (1999). On the kinematic analysis of robotic mechanisms. *The International Journal of Robotics Research*, 18(12):1147–1160. 1, 29

Pan, V. Y. and Wang, X. (2004). On rational number reconstruction and approximation. *SIAM J. Comput.*, 33(2):502–503. 44

Papadimitriou, C. H. (2003). Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., Chichester, UK. 36

Recio, T. and Vélez, M. P. (1999). Automatic discovery of theorems in elementary geometry. *Journal of Automated Reasoning*, 23(1):63–82. 29

Reeves, A. A. (1998). A parallel implementation of buchberger's algorithm over$\mathbb{Z}_p$for$p \leq$ 31991. *J. Symb. Comput.*, 26(2):229–244. 44

Roune, B. H. and Stillman, M. (2012). Practical groebner basis computation. 45

Seidenberg, A. (1974). Constructions in algebra. *Transactions of the American Mathematical Society*, 197:pp. 273–313. 37

Siegl, K. (1994). *A Parallel Factorization Tree Gröbner Basis Algorithm.* Johannes Kepler University Linz, Johannes Kepler University, Linz, Austria, 0 edition. 44

Stegers, T. (2005). *Faugère's F5 algorithm revisited.* 45, 52, 54

Steidel, S. (2012). *Gröbner Bases of Symmetric Ideals.* 43

Sun, Y. and Wang, D. (2011a). The f5 algorithm in buchberger's style. *Journal of Systems Science and Complexity*, 24:1218–1231. 45

Sun, Y. and Wang, D. (2011b). A generalized criterion for signature related gröbner basis algorithms. *CoRR*, abs/1101.3382. 45

Sun, Y. and Wang, D. (2012). A new proof for the correctness of the f5 algorithm. *Science China Mathematics*, pages 1–12. 45

Thomas, R. R. (1995). A geometric buchberger algorithm for integer programming. *Mathematics of Operations Research*, 20(4):864–884. 33

Thomas, R. R. (1998). Applications to integer programming. 33

Tran, Q.-N. (2000). A fast algorithm for gröbner basis conversion and its applications. *Journal of Symbolic Computation*, 30(4):451 – 467. 40

Tran, Q.-N. (2005). Ideal-specified term orders for elimination and applications in implicitization. *Tenth International Conference on Applications of Computer Algebra*, pages 15–24. 41

Tran, Q.-N. (2007). A new class of term orders for elimination. *Journal of Symbolic Computation*, pages 533–548. 41

Traverso, C. (1989). Gröbner trace algorithms. In *Proceedings of the International Symposium ISSAC'88 on Symbolic and Algebraic Computation*, ISAAC '88, pages 125–138, London, UK, UK. Springer-Verlag. 43, 70, 72

Traverso, C. (1996). Hilbert functions and the buchberger algorithm. *J. Symb. Comput.*, 22(4):355–376. 40, 60, 61, 64

Vidal, J.-P. (1990). *The computation of Gröbner bases on a shared memory multiprocessor*. Number v. 90-163 in The computation of Gröbner bases on a shared memory multiprocessor. School of Computer Science, Carnegie Mellon University. 44

Wang, P. S., Guy, M. J. T., and Davenport, J. H. (1982). P-adic reconstruction of rational numbers. *SIGSAM Bull.*, 16(2):2–3. 44

Wang, X. and Pan, V. Y. (2003). Acceleration of euclidean algorithm and rational number reconstruction. *SIAM Journal on Computing*, 32:2003. 44

Wilkinson, J. H. (1959). The evaluation of the zeros of ill-conditioned polynomials. part i. *Numerische Mathematik*, 1:150–166. 2

Winkler, F. (1988). A p-adic approach to the computation of gröbner bases. *Journal of Symbolic Computation*, 6(2–3):287 – 304. 43

Winkler, F. (1996). Polynomial Algorithms in Computer Algebra. 29, 31

Ypma, T. J. (1995). Historical development of the newton-raphson method. *SIAM review*, 37(4):531–551. 2

Zobnin, A. (2010). Generalization of the f5 algorithm for calculating gr̈bner bases for polynomial ideals. *Programming and Computer Software*, 36:75–82. 45