university of groningen

faculty of mathematics and natural sciences

An algebraic approach to the Boolean Satisfiability Problem

Master Project Mathematics

July 2016

Student: P.W. Bakker - Klooster

First supervisor: Prof. dr. J. Top

Second supervisor: Prof. dr. L.C. Verbrugge

Abstract

This thesis combines the field of Algebra and the field of Logic by investigating the relation between an algebraic and a propositional approach to the Boolean Satisfiability Problem (SAT). The algebraic variant is obtained by representing propositional formulas by polynomials in $\mathbb{F}_2[x_1, ..., x_n]$. We consider the common zeros of the ideal $\langle f(x_1, ..., x_n), x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ in order to determine whether f is satisfiable. To develop an algorithm which enables us to analyse and solve the problem, several algebraic concepts are considered. The most important results are obtained from concepts related to Gröbner bases. Finally, some algebraic algorithms are obtained which solve SAT. Using the computer algebra system Maple, we get some results on the running time of the algorithms.

Contents

1 Introduction **4**

1.1 Introduction to Logic 4

1.2 Introduction to Algebra 5

2 SAT **6**

2.1 Complexity theory 6

2.2 Complexity of SAT 8

2.3 SAT in algebraic terms 8

3 Gröbner Basis **10**

3.1 Introduction 10

3.2 Monomial orderings 11

3.3 Division Algorithm 13

3.4 Definition of Gröbner basis 15

3.5 Criterion for Gröbner basis 18

4 Theoretical Results **19**

4.1 Algebraic analysis 19

4.1.1 Radicals 19

4.1.2 The Chevalley-Warning theorem 20

4.1.3 Optimization over $\mathbb{Z}[x_1, .., x_n]$ 21

4.1.4 Hilbert's Nullstellensatz 22

4.1.5 Buchberger's Criterion 23

4.2 Logical Properties 24

4.2.1 CNF vs. DNF 24

4.2.2 Horn algorithm 24

4.2.3 Polynomials representing propositional formulas 26

5 Maple Results **26**

6 Conclusion **27**

7 Discussion **28**

A Maple Implementation **30**

1 Introduction

We study the Boolean Satisfiability problem (SAT), especially the relation between a propositional and an algebraic approach to the problem. The problem is known to be $\mathcal{NP}$-complete, so we do not expect to solve the problem in polynomial time. We are primarily interested in which algebraic theorems provide us a proper algorithm. Beforehand we decided to focus mainly on Gröbner bases and this approach provided us the major part of the algorithm. Implementation of this part in Maple enabled us to experiment with different instances, which finally led us to the last step that completed the algorithm.

Since the subject is interesting for different scientific fields, some parts of the thesis could be well-known to some readers. We therefore first display the structure of this thesis. In the remainder of this first chapter a few logic and algebraic notations and terms are fixed. Chapter 2 provides more information about SAT: it recalls its complexity and it presents a few variants of the problem. For readers unfamiliar with complexity theory, an introductory paragraph is included. The chapter finally contains the definition of SAT in algebraic terms which is used in the remainder of the thesis. After these introductory chapters, we analyse a generalization of the problem and come across the concept of Gröbner basis. This is covered in chapter 3. In chapter 4 we will focus again on the specific definition of SAT and apply various algebraic theories to it, including results of chapter 4. It provides a theoretical framework that enables us to check the satisfiability of a propositional formula. The resulting algorithms are presented in chapter 6, which also contains some results from experiments in Maple.

1.1 Introduction to Logic

Consider the countable set $\mathcal{P}$ of propositional variables $\mathcal{P} = \{p_i \mid i \in \mathbb{N}\}$. A *propositional formula* is a well-formed formula (wff) constructed from elements of $\mathcal{P}$ using the Boolean connectives $\neg, \wedge$ and $\vee$:

- Every $p_i \in \mathcal{P}$ is a wff;
- If P and Q are wffs, then so are $\neg P, (P \vee Q), (P \wedge Q)$;
- Nothing else is a wff execpt what can be constructed in finitely many steps from 1 and 2.

For readibility we omit the outer brackets of $(P \vee Q)$ and $(P \wedge Q)$. A *literal* is a propositional variable or its negation, and is called *positive* or *negative*, respectively. A *conjunction (disjunction)* is a propositional formula where $\wedge(\vee)$ is the main connective. A formula is in *Disjunctive Normal Form (DNF)* if it is a disjunction of conjunctions with literals. A formula is in *Conjunctive Normal Form (CNF)* if it is a conjunction of disjunctions with literals. These conjuncts of a CNF are often referred to as *clauses*.

A *truth assignment* v assigns a truth value to propositional formulas. A propositional variable p is true if and only if $v(p) = \mathit{true}$ and false if and only if $v(p) = \mathit{false}$. The truth value of a propositional formula can be determined using the following recursive definitions:

$$
\begin{aligned}
v(\neg P) &= \begin{cases} \mathit{true}, & \text{if } v(P) = \mathit{false} \\ \mathit{false}, & \text{otherwise;} \end{cases} \\
v(P \wedge Q) &= \begin{cases} \mathit{true}, & \text{if } v(P) = v(Q) = \mathit{true} \\ \mathit{false}, & \text{otherwise;} \end{cases} \\
v(P \vee Q) &= \begin{cases} \mathit{true}, & \text{if } v(P) = \mathit{true} \text{ or } v(Q) = \mathit{true} \\ \mathit{false}, & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}
$$

For disjunctions consisting of more than two disjuncts, we consider the leftmost connective as the main connective, for example $p_1 \vee (p_2 \vee p_3)$ instead of $(p_1 \vee p_2) \vee p_3$. This works the same for longer conjunctions.

Definition 1.1.1. *A propositional formula P on a set of Boolean variables is **satisfiable** if there exists a truth assignment v such that $v(P) = true$.*

Definition 1.1.2. *The **Boolean Satisfiability Problem (SAT)** questions whether a given propositional formula P is satisfiable.*

1.2 Introduction to Algebra

Definition 1.2.1. *A **monomial** in $x_1, ..., x_n$ is a product of the form*

$$x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

*where all of the exponents $\alpha_1, ..., \alpha_n$ are nonnegative integers. The **total degree** of this monomial is the sum $\alpha_1 + \ldots + \alpha_n$.*

The ring $k[x_1, ..., x_n]$ contains polynomials in the variables $x_1, ..., x_n$ with coefficients in the field k. For the algebraic variant of SAT we mainly deal with polynomials in $\mathbb{F}_2[x_1, ..., x_n]$. When we add or multiply such polynomials, the operations on the coefficients are taken as modulo 2 operations. Consider for example the polynomials $f = x^2 + x$ and $g = x^2 + 1$, both in $\mathbb{F}_2[x]$. We get $f + g = 2x^2 + x + 1 = x + 1$ since $2 \bmod 2 = 0$.

Definition 1.2.2. *An **ideal** I in $k[x_1, ..., x_n]$ is a nonempty subset $I \subset k[x_1, ..., x_n]$ such that*

- $\forall f, g \in I$: $f + g \in I$;
- $\forall f \in I, \forall h \in k[x_1, ..., x_n]$: $h \cdot f \in I$.

By $I = \langle f_1, ..., f_s \rangle$ in $k[x_1, ..., x_n]$, we denote the ideal over $k[x_1, ..., x_n]$ generated by the polynomials $f_1, ..., f_s$. Any element $f \in I$ can be written as $f = \sum_{i=1}^{s} h_i \cdot f_i$ where $h_i \in k[x_1, ..., x_n]$.

Definition 1.2.3. *Let k be a field, and let $f_1, ..., f_s$ be polynomials in $k[x_1, ..., x_n]$. Then we set*

$$V(f_1, ..., f_s) = \{(a_1, ..., a_n) \in k^n \mid f_i(a_1, ..., a_n) = 0 \text{ for all } 1 \leq i \leq s\}.$$

*We call $V(f_1, ..., f_s)$ the **affine variety** defined by $f_1, ..., f_s$.*

Definition 1.2.4. *Let R be a ring. Then $x \in R$ is called **nilpotent** if there exists a positive integer n such that $x^n = 0$.*

Finally, we note that the length of an element $\alpha \in \mathbb{Z}^n_{\geq 0}$ is defined as $|\alpha| = \sum_{i=1}^{n} a_i$.

2 SAT

Now we have defined the Boolean Satisfiability problem, we will discuss its complexity. For that purpose section 2.1 contains a short introduction to complexity theory, the field that studies the difficulty of computational problems, based on parts of the books *Algorithms and complexity* by Wilf [14], *Scheduling* by Pinedo [11], and *Scheduling Algorithms* by Brucker [3]. We recommend this literature to those who are unfamiliar to this field. Afterwards we consider the complexity of SAT and we finish with an algebraic definition of SAT which forms a starting point for the remainder of the thesis.

2.1 Complexity theory

A computational problem can be solved by developing an algorithm which solves the problem for each instance. The complexity of the problem is then measured by the complexity of the algorithm. To determine the latter, one determines the computational steps involved in the algorithm. The efficiency is then measured by an upper bound $T(n)$ on the number of steps that the algorithm needs to solve the problem for an instance x of length n.

Example. We look for a solution of an instance x of a computational problem. Assume that for input size n of x, the algorithm $\mathcal{A}$ takes a maximum number of steps equal to $T(n) = 1400+50n^2+3n^4$ to obtain a solution. For small instances (small n), the first two terms of $T(n)$ have the largest impact on the number of steps of the algorithm. However for large instances the last term, $3n^4$, counts the most. To get a good approximation of $T(n)$ we look at its asymptotic behaviour. Then only the term that increases the fastest, $3n^4$, is important and we can also ignore its coefficient. Eventually we refer to $\mathcal{A}$ as an $\mathcal{O}(n^4)$ algorithm.

In most cases it is difficult to determine the precise form of $T(n)$, hence it is in general replaced by its asymptotic order. We formalize the idea of the example:

Definition 2.1.1. *$T(n) \in \mathcal{O}(g(n))$ if there exist constants $c > 0$ and a non-negative integer n_0 such that $T(n) \leq cg(n)$ for all integers $n \geq n_0$.*

The function $g(n)$ can be any function of n and it establishes the time complexity of the algorithm. Algorithms with $T(n) \in \mathcal{O}(3^n)$ are for example exponential in the size of the input (at least in the worst case). The algorithm $\mathcal{A}$ in the example was an $\mathcal{O}(n^4)$ algorithm, which is a polynomial time algorithm and is preferred over an exponential time algorithm. The decision

problems for which a polynomial time algorithm exists, are known as the 'easy' decision problems. They define the complexity class $\mathcal{P}$.

Definition 2.1.2. *A decision problem belongs to the* ***class*** $\mathcal{P}$ *if there is an algorithm* $\mathcal{A}$ *and a number* c *such that for every instance* x *of the problem the algorithm* $\mathcal{A}$ *will produce a solution in time* $\mathcal{O}(n^c)$*, where* $n = |x|$*.*

Another class of decision problems is based on the time an algorithm takes to verify whether a given solution is correct or not. The definition of this class $\mathcal{NP}$ is more involved than the definition of $\mathcal{P}$ and will therefore be clarified in an example.

Definition 2.1.3. *A decision problem belongs to the* ***class*** $\mathcal{NP}$ *if there is an algorithm* $\mathcal{A}$ *that does the following:*

(a) For each instance I *with answer 'yes', there is a witness* $W(I)$ *such that when the pair* $(I, W(I))$ *is input to algorithm* $\mathcal{A}$*, it recognizes that* I *leads to 'yes'.*

(b) For each instance I *with answer 'no', there is no choice of a witness* $W(I)$ *that will cause* $\mathcal{A}$ *to recognize that* I *leads to 'yes'.*

(c) $\mathcal{A}$ *is a polynomial time algorithm.*

Example. Consider the instance $I = p_1 \vee p_2$ of the decision problem SAT. This instance leads to yes, so there should exist a confirming witness $W(I)$. Take for example as witness $W(I)$ the truth assignment v such that $v(p_1) = \mathit{true}$ and $v(p_2) = \mathit{true}$. Using this witness we verify that I is indeed satisfiable: $v(I) = v(p_1 \vee p_2) = \mathit{true}$. Thus I and $W(I)$ lead to answer 'yes' to SAT.
Consider the instance $I = p_1 \wedge \neg p_1$ of the decision problem SAT. This instance leads to no, thus by Definition 2.1.3 there should not exists a witness $W(I)$ confirming that I is satisfiable. For SAT a witness occurs in the form of a truth assignment, so there should not exists a truth assignment such that $v(I) = \mathit{true}$. This in indeed the case, since $v(p_1 \wedge \neg p_2) = \mathit{true}$ only if $v(p_1) = \mathit{true}$ and $v(\neg p_1) = \mathit{true}$, which is impossible.

The definition and the example illustrate that $\mathcal{NP}$ indeed provides the class of all decision problems for which a solution can be verified in polynomial time. We now define a subclass of problems in $\mathcal{NP}$ with an important property: if we can solve one problem in this subclass, we can use its algorithm to solve any problem in $\mathcal{NP}$ with just a little more work.

Definition 2.1.4. *Let* P *and* Q *be two decision problems. We say that* P *is* ***quickly reducible*** *to* Q *if there exists a polynomial-time computable function that transforms instances* I_P *of* P *into instances* I_Q *of* Q *such that both instances have the same answer ('yes' or 'no').*

Definition 2.1.5. *A decision problem* Q *is called* $\mathcal{NP}$***-complete*** *if* $Q \in \mathcal{NP}$ *and for all other decision problems* $P \in \mathcal{NP}$ *we have that* P *is quickly reducible to* Q*.*

The family of $\mathcal{NP}$-complete problems plays an important role in complexity theory. If one of the $\mathcal{NP}$-complete problems could be solved in polynomial time, then each problem in $\mathcal{NP}$ would be solved in polynomial time by Definition 2.1.5. The answer to the question whether $\mathcal{P} = \mathcal{NP}$ is a famous open problem in complexity theory.

2.2 Complexity of SAT

Now we talked about $\mathcal{NP}$-complete problems in theory, we ask ourselves whether there exist problems that have the property of Definition 2.1.5. This is indeed the case; in 1971 Stephen Cook discovered the first member of the NP-complete family:

Theorem 2.1 (Cook, [5]). *SAT is $\mathcal{NP}$-complete.*

We consider some modifications of SAT and investigate their complexity. A famous example is *CNF-SAT*, that considers the satisfiability of propositional formulas in CNF. Since each propositional formula can be converted to a CNF formula, CNF-SAT is also in $\mathcal{NP}$. Note that translation to CNF may result in an exponentially larger instance. For that reason, we have chosen to consider propositional formulas in their original form instead of their CNF equivalences.

Another example is the problem 3-*SAT*, which considers the satisfiability of formulas in CNF such that each clause consists of exactly 3 literals. It is $\mathcal{NP}$-complete and is often used to reduce other problems that are $\mathcal{NP}$-complete [10]. In this thesis we do not focus on this kind of formulas, but it could be interesting to investigate them in the same manner.

Finally we consider a subclass of propositional formulas for which a linear time algorithm exists: the set of *Horn formulas* [8]. A formula is in *Horn form* if it is in CNF and if each clause contains at most one positive literal. The algorithm works as follows:

Horn algorithm for propositional formulas

Given Horn-formula P of variables $p_1, ..., p_n$ and disjuncts $D_1, ..., D_m$ follow these steps:

1. Consider clauses D_i that are composed of just one literal.
   (a) If $D_i = \neg D_j$ for two of such clauses **stop**; P is not satisfiable.
   (b) If there exist clauses of one literal, choose one and go to step 2.
   (c) Otherwise go to step 3.
2. Assign a truth value to the atom p_j of D_i such that D_i is satisfied.
   (a) If the literal D_i appears as disjunct of some D_k, then remove D_k from P;
   (b) If the negation of D_i appears as disjunct of some D_k, then remove D_i from D_k;
3. Go to step 1 if there is still a clause composed of just one literal. Otherwise assign *false* to all unassigned variables and **stop**; P is satisfiable.

A proof for the correctness of this algorithm can be found in [2, p.485]. The Horn formulas and Horn algorithm will return in Chapter 5 where we apply it to the algebraic approach of SAT.

2.3 SAT in algebraic terms

In order to analyse the relation between the logical and algebraic description of SAT, we present an algebraic approach to the problem. First we develop a way to translate a propositional formula

P to a polynomial f_P. We consider variables x_i, corresponding to the atoms p_i, representing the truth value of p_i. Evaluation of f_P in $(x_1, ..., x_n)$ should result in a value equivalent to $v(P)$, so we require $f_P \in \mathbb{F}_2[x_1, ..., x_n]$. The computer algebra system Maple[1] provides us with the following recursive definitions of polynomials, where P_i is a propositional formula:

$$\begin{aligned} f_{(\neg P)} &= 1 + f_P; \\ f_{(P_1 \wedge P_2)} &= f_{P_1} \cdot f_{P_2}; \\ f_{(P_1 \vee P_2)} &= 1 + (1 + f_{P_1}) \cdot (1 + f_{P_2}). \end{aligned} \tag{2}$$

In addition we define $f_{p_i} = x_i$. By applying the rules in Equation 2 iteratively, we are able to construct for each propositional formula a polynomial in $\mathbb{F}_2[x_1, ..., x_n]$. In this way, a truth assignment $v(p_i) = \textit{true}$ or $\textit{false}$ corresponds to an evaluation $x_i = 1$ or 0, respectively.

Example. The polynomial representing the propositional formula $P = (p_1 \wedge p_2) \vee \neg p_3$ is obtained using Equation 2 and the definition of f_{p_i}. Remember that operations on coefficients are taken as modulo 2 operations.

$$\begin{aligned} f_P &= f_{(p_1 \wedge p_2) \vee \neg p_3} \\ &= 1 + (1 + f_{p_1 \wedge p_2})(1 + f_{\neg p_3}) \\ &= 1 + (1 + f_{p_1} \cdot f_{p_2})(1 + 1 + f_{p_3}) \\ &= 1 + (1 + f_{p_1} \cdot f_{p_2}) f_{p_3} \\ &= 1 + (1 + x_1 \cdot x_2) \cdot x_3 \\ &= 1 + x_1 \cdot x_2 \cdot x_3 + x_3. \end{aligned}$$

From Equation 2 we can observe that a propositional formula P can be translated into f_P in linear time in the size of P. Writing f_P without brackets, on the other hand, may result in a number of terms which is exponential in the size of P. Consider for example a disjunction of n atoms p_i, which results in a polynomial consisting of 2^n terms:

$$\begin{aligned} f_{p_1 \vee \cdots \vee p_n} &= 1 + (1 + x_1)(1 + x_2) \cdots (1 + x_n) \\ &= 1 + \sum_{(a_1, ..., a_n) \in \{0,1\}^n} x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}. \end{aligned}$$

Now a polynomial can be obtained for each propositional formula, we can define what satisfiability means for the polynomial. We come up with a definition equivalent to Definition 1.1.1.

Definition 2.3.1. *$f \in \mathbb{F}_2[x_1, ..., x_n]$ is **satisfiable** if there exists a point $(a_1, ..., a_n) \in \mathbb{F}_2^n$ such that $f(a_1, ..., a_n) = 1$.*

In algebra terms the decision variant of SAT asks whether a point can be found such that evaluation of the polynomial gives the value 1. This is equivalent to the question whether $f + 1 \in \mathbb{F}_2[x_1, ..., x_n]$ has a zero. Since the variables $x_1, ..., x_n$ represent truth values, we are just interested

[1] https://www.maplesoft.com/support/help/maple/view.aspx?path=Logic%2fExport

in zeros in $\mathbb{F}_2^n$. We set $\bar{f} \equiv f_P + 1 \mod (x_1^2 + x_1, ..., x_n^2 + x_n)$, so that the search for the zeros of $\bar{f}$ provides all and only those zeros of $f + 1$ that are appropriate.

Definition 2.3.2. *$f \in \mathbb{F}_2[x_1, ..., x_n]$ is **satisfiable** if there exists a point $(a_1, ..., a_n) \in \mathbb{F}_2^n$ such that $\bar{f}(a_1, ..., a_n) = 0$.*

Remark. By construction, one obtains that a propositional formula P is satisfiable if and only if the corresponding polynomial f_P is satisfiable.

3 Gröbner Basis

3.1 Introduction

Due to the definition of SAT in the previous chapter, we focus on finding a zero in $\mathbb{F}_2^n$ of a polynomial $f(x_1, ..., x_n)$ over $\mathbb{F}_2$. Since x_i is defined as a truth value in $\{0, 1\}$, this is equivalent to the question whether there exists a solution to the system of equations in Equation 3.

$$\begin{aligned} f(x_1, ..., x_n) &= 0 \\ x_1^2 + x_1 &= 0 \\ x_2^2 + x_2 &= 0 \\ &\vdots \\ x_n^2 + x_n &= 0 \end{aligned} \tag{3}$$

The goal of this chapter is to obtain a method that decreases the difficulty of solving such systems. We focus on the general system of equations with polynomials $f \in k[x_1, ..., x_n]$ for any field k:

$$f_1 = 0, ..., f_s = 0. \tag{4}$$

Example. If all f_i are linear polynomials, we can apply Gaussian elimination and obtain a different system. This new system has the same solution set as the original one, but is easier to solve.

The idea of transforming the system to a similar system with the same solution set and which is easier to solve, can also be used for the case of non-linear, multivariate polynomials. Remember that the common zeros of several polynomials form a variety. The next theorem states a property of varieties and ideals that will be useful to rewrite a system in an appropriate way.

Theorem 3.1. *$V(f_1, ..., f_s) = V(I)$ for each set of polynomials $f_1, ..., f_s \in k[x_1, ..., x_n]$ such that $I = \langle f_1, ..., f_s \rangle$.*

Proof. We first prove that $V(f_1, ..., f_s) \subset V(I)$. Let $z \in V(f_1, ..., f_s)$, then $f_1(z) = f_2(z) = \ldots = f_s(z) = 0$. Any element f of I can be written as $f = \sum_{i=1}^{s} h_i f_i$ for $h_i \in k[x_1, ..., x_n]$. Since $f_i(z) = 0$ for all i, we have that $f(z) = 0$. Thus $z \in V(I)$.

To prove $V(f_1, ..., f_s) \supset V(I)$ we consider $z \in V(I)$. This z is a zero of all polynomials in I, so especially of each generator f_i. Thus $f_i(z) = 0$ for each i so that $z \in V(f_1, ..., f_s)$. □

From this theorem we can conclude that a variety is determined by an ideal. Different generating sets of I represent different systems of equations all of which have the same solution set. Investigating different ways to represent the ideal $I = \langle f_1, ..., f_s \rangle$ is the key in our search for a better system equivalent to Equation 4. We define two problems to control the examination for a good generating set:

- *Ideal Description Problem.* Does every ideal $I \subset k[x_1, ..., x_n]$ have a finite generating set?
- *Ideal Membership Problem.* Given $f \in k[x_1, ..., x_n]$ and an ideal $I = \langle f_1, ..., f_s \rangle$, determine if $f \in I$.

Example. The Ideal Membership Problem is easy to solve when we are dealing with polynomials in just one variable. We can apply the Euclidean algorithm on such polynomials and get that $\langle f_1, ..., f_s \rangle = \langle \gcd(f_1, ..., f_s) \rangle$. Then $f \in \langle f_1, ..., f_s \rangle$ if and only if the remainder on division of f by $\gcd(f_1, ..., f_s)$ is zero.

Translating this successful approach to the general case, we encounter some difficulties because the Euclidean algorithm is not applicable to multivariate polynomials. We thus have to find a similar way to solve the Ideal Membership Problem in general. The next two sections, 3.2 and 3.3, introduce the concept of monomial orderings and take it as a starting point to develop an appropriate division algorithm for multivariate polynomials. The content is based on the explanation of the subject in the books *Ideals, Varieties and Algorithms* by Cox et al. [6] and *Introduction to Gröbner bases* by Adams and Loustaunau [1].
Finally it is important to note from the example that the solution method to the Ideal Membership Problem only works when we already have an affirmative answer to the Ideal Description Problem. This will be given in Theorem 3.5. We will see this coherence between the two problems also in the next chapter when we provide complete solutions to the above questions.

3.2 Monomial orderings

Definition 3.2.1. *A **monomial ordering** $>$ on $k[x_1, ..., x_n]$ is any relation $>$ on $\mathbb{Z}^n_{\geq 0}$, or equivalently, any relation on the set of monomials x^α, $\alpha \in \mathbb{Z}^n_{\geq 0}$, satisfying:*

1. *$>$ is a total ordering on $\mathbb{Z}^n_{\geq 0}$. This means that every pair α and β satisfies exactly one of the following relations: $\alpha < \beta, \alpha = \beta$ or $\alpha > \beta$.*
2. *If $\alpha > \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0}$, then $\alpha + \gamma > \beta + \gamma$.*
3. *$>$ is a well-ordering on $\mathbb{Z}^n_{\geq 0}$. This means that every nonempty subset of $\mathbb{Z}^n_{\geq 0}$ has a smallest element under $>$.*

A monomial ordering $>$ on $\mathbb{Z}^n_{\geq 0}$ enables us to establish the relative position of two monomials: $x^\alpha > x^\beta$ if $\alpha > \beta$. By the first condition of Definition 3.2.1 it is possible to compare each pair of monomials to each other. The requirement of *well-ordering* will be useful later on to ensure termination of algorithms. We now discuss three orderings, all of which are monomial ordering as proved in [6].

Definition 3.2.2 (Lexicographic Order)**.** *Let $\alpha = (\alpha_1, ..., \alpha_n)$ and $\beta = (\beta_1, ..., \beta_n) \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{lex} \beta$ if, in the vector difference $\alpha - \beta \in \mathbb{Z}^n$, the leftmost nonzero entry is positive.*

The lexicographic order (or lex order) arranges monomials in the same way as words are ordered in dictionary. It favors the monomial with the larger power in the first conflicting variable. It is important to realize that the construction of α depends on the ordering of variables in the monomial x^α. We therefore make the assumption that a monomial order is always based on an order on variables such that $x > y > z$ or similarly $x_1 > x_2 > \cdots > x_n$.

Example. Consider the monomials $\boldsymbol{x}^\alpha = x$ and $\boldsymbol{x}^\beta = y^2z^2$. Then $\alpha = (1, 0, 0)$ and $\beta = (0, 2, 2)$, so we have $\alpha - \beta = (1, -2, -2)$. The leftmost nonzero entry is positive, thus $x >_{lex} y^2z^2$.

For one-variable polynomials it is common that terms are ordered by their degrees. From that point of view it is counter-intuitive to say that x is larger than y^2z^2. Therefore we introduce two monomial orderings focusing first on the degree of the monomials: the graded lexicographic order (or grlex order) and the graded reverse lexicographic order (or grevlex order).

Definition 3.2.3 (Graded Lexicographic Order)**.** *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{grlex} \beta$ if $|\alpha| > |\beta|$, or $|\alpha| = |\beta|$ and $\alpha >_{lex} \beta$.*

Definition 3.2.4 (Graded Reverse Lexicographic Order)**.** *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{grevlex} \beta$ if $|\alpha| > |\beta|$, or $|\alpha| = |\beta|$ and the rightmost nonzero entry of $\alpha - \beta \in \mathbb{Z}^n$ is negative.*

The grlex and grevlex orderings both order by total degree, but ties are broken in a different way. In case of equal total degree, grlex uses the lex order to determine the largest monomial. The grevlex order, however, looks at the rightmost nonzero entry and prefers the monomial with the smallest power in the corresponding variable.

Example. Consider x^4yz and x^3yz^2. Both monomials have total degree 6 so we have the case that $|\alpha| = |\beta|$ for $\alpha = (4, 1, 1)$ and $\beta = (3, 2, 1)$. We look at $\alpha - \beta = (1, 0, -1)$ and find that the grlex and grevlex order lead to the same ordering, but for different reasons: $x^4yz >_{grlex} x^3yz^2$ because of the largest power in x, while $x^4yz >_{grevlex} x^3yz^2$ because of the smallest power in z.

Using these monomial orderings, we are able to order the terms of polynomials $f = \sum_\alpha a_\alpha X^\alpha \in k[x_1, ..., x_n]$ by ordering the monomials X^α. In order to be able to define an appropriate division algorithm in the next section, we introduce the following terminology. Note that the maximum should be taken with respect to a given monomial ordering.

Definition 3.2.5. *Let $f = \sum_\alpha a_\alpha X^\alpha$ be a nonzero polynomial in $k[x_1, ..., x_n]$ and let $>$ be a monomial order.*

i. *The **multidegree** of f is* $\text{multideg}(f) = \max(\alpha \in \mathbb{Z}^n_{\geq 0} : \alpha \neq 0)$

ii. *The **leading coefficient** of f is* $\text{LC}(f) = a_{\text{multideg}(f)} \in k$.

iii. *The **leading monomial** of f is* $\text{LM}(f) = x^{\text{multideg}(f)}$.

iv. *The **leading term** of f is* $\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f)$.

3.3 Division Algorithm

We develop a Division Algorithm in $k[x_1, ..., x_n]$ that divides a polynomial $f \in k[x_1, .., x_n]$ by a set of polynomials $f_1, ..., f_s \in k[x_1, ..., x_n]$. The algorithm should enable us to write $f = \sum_{i=1}^{s} a_i f_i + r$ for $a_1, ..., a_s, r \in k[x_1, ..., x_n]$. We think of r as the remainder of f on division by $\{f_1, ..., f_s\}$ and therefore require that none of its terms should be divisible by $\text{LT}(f_i)$ for any i. The computation of these polynomials $a_1, ..., a_s, r$ is based on the long division method for one-variable polynomials, as one can see in the following algorithm.

Division Algorithm in $k[x_1, ..., x_n]$

Data: $f_1, ..., f_s, f$

Result: $a_1, ..., a_s, r$ such that $f = \sum_{i=1}^{s} a_i f_i + r$

initialization $a_1 := 0, ..., a_s := 0, r := 0, p := f$;

while $p \neq 0$ **do**

  if $\{\ i \mid \text{LT}(f_i)$ *divides* $\text{LT}(p)\} \neq \emptyset$ **then**

    $i := min\{\ i \mid \text{LT}(f_i)$ divides $\text{LT}(p)\}$;

    $a_i := a_i + \dfrac{\text{LT}(p)}{\text{LT}(f_i)}$;

    $p := p - \dfrac{\text{LT}(p)}{\text{LT}(f_i)} f_i$;

  else

    $r := r + \text{LT}(p)$;

    $p := p - \text{LT}(p)$;

  end

end

It is important to realize that we first have to select a monomial ordering before we can run the algorithm. To ensure that this algorithm works for each pair of polynomials $f, f_1, ..., f_s \in k[x_1, ..., x_n]$, we state the following theorem and provide a proof using the Division Algorithm in $k[x_1, ..., x_n]$.

Theorem 3.2. *Fix a monomial order and let $F = \{f_1, ..., f_s\}$ be an ordered s-tuple of polynomials*

in $k[x_1, ..., x_n]$*. Then every* f *can be written as*

$$f = \sum_{i=1}^{s} a_i f_i + r \tag{5}$$

where $a_i, r \in k[x_1, ..., x_n]$*, and either* $r = 0$ *or* r *is a linear combination, with coefficients in* k*, of monomials, none of which is divisible by any of the* $\mathrm{LT}(f_1), ..., \mathrm{LT}(f_s)$*. Furthermore, we can show that if* $a_i f_i \neq 0$*, then we have* $\mathrm{multideg}(f) \geq \mathrm{multideg}(a_i f_i)$*.*

Proof. We first show by induction that at every stage of the algorithm we have that

$$f = a_1 f_1 + \cdots a_s f_s + p + r. \tag{6}$$

This is obviously true for the initialization. Suppose now that Equation 6 holds at step m. We want to proof that it also holds for step $m + 1$. There are two possible scenarios for step $m + 1$.

1. If some $\mathrm{LT}(f_i)$ divides $\mathrm{LT}(p)$, then $a_i \mapsto a_i + \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)}$ and $p \mapsto p - \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)} f_i$. Since

$$\left(a_i + \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)}\right) f_i + p - \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)}(f_i) = a_i f_i + p,$$

   and all other variables in f are unaffected, we have that (6) remains true.

2. If $\mathrm{LT}(p)$ is not divisible by some $\mathrm{LT}(f_i)$, then $r \mapsto r + \mathrm{LT}(p)$ and $p \mapsto p - \mathrm{LT}(p)$. It is easy to see that the sum $r + p$ is unchanged under these modifications. Since all other variables in f are unaffected, we have that (6) remains true.

The condition that none of the terms of r is divisible by some $\mathrm{LT}(f_i)$ is satisfied: during execution of the algorithm, the remainder r is changed only by adding a term, $\mathrm{LT}(p)$, that is not divisible by some $\mathrm{LT}(f_i)$. It remains to show that the algorithm terminates. To do this, we focus on the variable p, which is redefined in each step, and claim that it either becomes 0 or its multidegree decreases. It then follows that the algorithm terminates, since the well-ordering property of $>$ ensures that we cannot get an infinite decreasing sequence of multidegrees. Thus the algorithm terminates after finitely many steps, because $p = 0$ eventually happens. To prove our claim about p we consider the two possible redefinitions of p:

1. $p \mapsto p - \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)} f_i$. The leading term of $\frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)} f_i$ is equal to $\frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)} \cdot \mathrm{LT}(f_i) = \mathrm{LT}(p)$. The redefined $p' = p - \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)} f_i$ has therefore a strictly smaller multidegree if $p' \neq 0$.

2. $p \mapsto p - \mathrm{LT}(p)$. This case obviously satisfies our claim.

We finally prove that $\mathrm{multideg}(f) \geq \mathrm{multideg}(a_i f_i)$. Observe that each term of a_i is of the form $\frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)}$ for one of the values of p. From the initialization $p := f$ and the fact that the multidegree of p decreases, we conclude that $\mathrm{LT}(p) < \mathrm{LT}(f)$ for each occurence of p. Thus indeed $\mathrm{multideg}(f) \geq \mathrm{multideg}(a_i f_i)$ when $a_i f_i \neq 0$. □

In the following example we show how the algorithm works and that it unfortunately does not provide a unique remainder.

Example. We divide $f = x^2y + xy^2 + y^2$ by the polynomials $f_1 = xy + 1$ and $f_2 = y^2 + 1$, in this order. The division algorithm works as follows:

$$
\begin{array}{rlll}
a_1\colon & x+y & & \\
a_2\colon & 1 & & r \\
\hline
f_1 = xy+1 & x^2y + xy^2 + y^2 & & \\
f_2 = y^2+1 & & & \\
 & \underline{x^2y - x} & & \\
 & xy^2 + x + y^2 & & \\
 & \underline{xy^2 - y} & & \\
 & x + y^2 + y & & \\
 & \underline{y^2 + y} & \to x & \\
 & y^2 - 1 & & \\
 & \underline{y+1} & & \\
 & 1 & \to x+y & \\
 & \underline{0} & \to x+y+1 &
\end{array}
$$

We obtain that $r = x+y+1$. However, when we divide f first by f_2 and thereafter by f_1, applying the division algorithm results in $r = 2x + 1$.

It is clear that $r = 0$ is a sufficient condition for being in the ideal. From the fact that the remainder is not uniquely determined, we can derive that $r = 0$ is not a necessary condition. In the next sections we focus on modifications of the set $f_1, ..., f_s$ to find sets that do deliver a unique remainder and hence solve the Ideal Membership Problem.

3.4 Definition of Gröbner basis

Introduction of monomials does not only lead to an answer for Ideal Membership Problem but also provides a solution on the Ideal Description Problem.

Definition 3.4.1. *An ideal $I \subset k[x_1, ..., x_n]$ is a **monomial ideal** if there is a subset $A \subset \mathbb{Z}^n_{\geq 0}$ (possibly infinite) such that I consists of all polynomials which are finite sums of the form $\sum_{\alpha \in A} h_\alpha x^\alpha$, where $h_\alpha \in k[x_1, ..., x_n]$. In this case we write $I = \langle x^\alpha \mid \alpha \in A \rangle$.*

To determine a remainder on division for $f \in I = \langle x^\alpha \mid \alpha \in A \rangle$ we need a finite generating set of I. The next theorem provides an answer to the Ideal Description Problem for monomial ideals.

Theorem 3.3 (Dickson's Lemma)**.** *Let $I = \langle x^\alpha \mid \alpha \in A \rangle \subseteq k[x_1, ..., x_n]$ be a monomial ideal. Then I can be written in the form $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$, where $\alpha(1), ..., \alpha(s) \in A$. In particular, I has a finite basis.*

Proof. See [7]. □

This result allows us to give an answer to the Ideal Membership Problem for monomial ideals.

Theorem 3.4. *If $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$ is a monomial ideal, then a polynomial f lies in I if and only if the remainder of f on division by $x^{\alpha(1)}, ..., x^{\alpha(s)}$ is zero.*

Proof. ($\Leftarrow$). True by the definition of an ideal.
($\Rightarrow$). Let $f \in I$. Remember from the definition of monomials ideals, that there exists a (possibly infinite) set $A \subset \mathbb{Z}^n_{\geq 0}$ such that every element of I can be written as a finite sum $\sum_{\alpha \in A} h_\alpha x^\alpha$ for monomials $x^\alpha \in I$ and $h_\alpha \in k[x_1, ..., x_n]$. Thus $f = \sum_{\alpha \in A} h_\alpha x^\alpha$. If we expand each h_α as a linear combination of monomials, we see that every term on the right-hand side of the equation is divisible by some x^α. Since $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$, a monomial x^β lies in I if and only if it is divisible by $x^{\alpha(i)}$ for some $i \in \{1, ..., s\}$ (see Lemma 2.4.2 of [6]). We thus conclude that every term of $\sum_{\alpha \in A} h_\alpha x^\alpha$ is divisible by some $x^{\alpha(i)}$. Hence the left-hand side, f, must have the same property, which results in zero remainder. □

Our next goal is to provide a complete answer to the Ideal Description Problem. When the solution is obtained, it enables us to the solve the Ideal Membership Problem. In our approach we focus on the leading term of each polynomial f, which is unique once a monomial ordering is chosen.

Definition 3.4.2. *Let $I \subset k[x_1, ..., x_n]$ be an ideal other than $\{0\}$.*

i. We denote by $\mathrm{LT}(I)$ the set of leading terms of elements of I.

ii. We denote by $\langle \mathrm{LT}(I) \rangle$ the ideal generated by the elements of $\mathrm{LT}(I)$.

It is important to realize that for $I = \langle f_1, ..., f_s \rangle$, the ideal $\langle \mathrm{LT}(I) \rangle$ is not necessarily equal to $\langle \mathrm{LT}(f_1), ..., \mathrm{LT}(f_1) \rangle$.

Example. Consider the ideal I generated by $f_1 = y^2 + 1$ and $f_2 = x^2 y$. Then $x^2 = x^2 \cdot (y^2 + 1) - y \cdot x^2 y$ is an element of I and thus $x^2 = \mathrm{LT}(x^2) \in \langle \mathrm{LT}(I) \rangle$. But $x^2 \notin \langle \mathrm{LT}(f_1), \mathrm{LT}(f_2) \rangle$, since x^2 is not divisible by the leading terms of f_1 and f_2, y^2 and $x^2 y$ respectively.

This example shows that for $I = \langle f_1, ..., f_s \rangle$ the ideal $\langle \mathrm{LT}(I) \rangle$ is not necessarily equal to $\langle \mathrm{LT}(f_1), ..., \mathrm{LT}(f_s) \rangle$. From Dickson's Lemma (Theorem 3.3), however, we can deduce that for each ideal I there should exist a finite set $\{g_1, ..., g_t\} \subset I$ such that $\langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle$. We furthermore can prove that $\langle \mathrm{LT}(I) \rangle$ is a monomial ideal. These results are presented in the following theorem and a complete proof can be found in [6, p. 76].

Proposition 1. *Let $I \subset k[x_1, ..., x_n]$ be an ideal.*

- *$\langle \mathrm{LT}(I) \rangle$ is a monomial ideal.*
- *There are $g_1, ..., g_t \in I$ such that $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle$.*

Using this proposition and the division algorithm of the previous chapter/paragraph, we are able to prove the existence of a finite set of generators for every ideal I in $k[x_1, ..., x_n]$. This finally answers the ideal description problem.

Theorem 3.5 (Hilbert Basis Theorem)**.** *Every ideal $I \subset k[x_1, ..., x_n]$ has a finite generating set. That is, $I = \langle g_1, ..., g_t \rangle$ for some $g_1, ..., g_t \in I$.*

Proof. When $I = \{0\}$, we take $\{0\}$ as finite generating set. If I contains some nonzero polynomials, we can construct a set $g_1, ..., g_t \in I$ using Proposition 1. This set has the property $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle$. Our claim is that $I = \langle g_1, ..., g_t \rangle$.
Since each $g_i \in I$ we have that $\langle g_1, ..., g_t \rangle \subset I$. To proof that $I \subset \langle g_1, ..., g_t \rangle$, let $f \in I$ be any polynomial. Using the division algorithm for multivariate polynomials, we divide f by the ordered set $\{g_1, ..., g_t\}$. We get

$$f = \sum_{i=1}^{t} a_i g_i + r,$$

where each term of r is not divisible by any $\mathrm{LT}(g_i)$. Note that $r = f - \sum_{i=1}^{t} a_i g_i \in I$. We can proof that $r = 0$ which in turn implies that $f \in \langle g_1, ..., g_t \rangle$. Assume for contradiction that $r \neq 0$. Then $\mathrm{LT}(r) \in \langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle$, so it must be divisible by some $\mathrm{LT}(g_i)$. This contradicts the fact that no term of r is divisible by any $\mathrm{LT}(g_1), ..., \mathrm{LT}(g_t)$.

□

The basis $\{g_1, ..., g_t\}$ defined in the proof of the above theorem satisfies the relation $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle$. We found earlier that not every basis of I has this property. The bases that do have this quality, play an important role in answering the ideal membership problem. Observe that these bases are such that the leading term of any element of I is divisible by one of the $\mathrm{LT}(g_i)$. We give them the following name.

Definition 3.4.3 (Gröbner Basis)**.** *Fix a monomial order. A finite subset $G = \{g_1, ..., g_t\}$ of an ideal I is said to be a **Gröbner basis** if*

$$\langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle.$$

From the proof of Theorem 3.5 we can observe that any Gröbner basis for an ideal I forms in fact a basis of I. It furthermore shows that for every nonzero ideal $I \in k[x_1, .., x_n]$ there exists a Gröbner basis, namely the set $\{g_1, ..., g_t\}$ constructed by Proposition 1. A Gröbner basis of an ideal establishes a useful result with regard to the multivariate division algorithm.

Proposition 2. *Let $G = \{g_1, ..., g_t\}$ be a Gröbner basis for an ideal $I \subset k[x_1, ..., x_n]$ and let $f \in k[x_1, ..., x_n]$. Then there is a unique $r \in k[x_1, ..., x_n]$ with the following two properties:*

- *No term of r is divisible by any $\mathrm{LT}(g_1), ..., \mathrm{LT}(g_t)$.*
- *There is a $g \in I$ such that $f = g + r$.*

In particular, r is the remainder on division of f by G no matter how the elements of G are listed when using the division algorithm.

When developing the division algorithm, we found out that the only drawback was that the remainder was not uniquely determined. Proposition 2 shows that once we obtain a Gröbner basis G for the ideal I we can determine a unique remainder on division by G. We are thus able to solve the ideal membership problem:

Corollary 1. *Let $G = \{g_1, ..., g_t\}$ be a Gröbner basis for an ideal $I \subset k[x_1, ..., x_n]$ and let $f \in k[x_1, ..., x_n]$. Then $f \in I$ if and only if the remainder on division of f by G is zero.*

3.5 Criterion for Gröbner basis

From the previous section we know that once we obtained a Gröbner basis G for an ideal, we can solve the ideal membership problem easily by computing the remainder on division by G. Thus to solve the system

$$f_1 = f_2 = \ldots = f_n = 0,$$

we have to determine whether or not $\{f_1, ..., f_n\}$ already forms a Gröbner basis for $I = \langle f_1, ..., f_n \rangle$. In this section we focus on the question whether a basis is a Gröbner basis. Since this is closely related to the remainder on division, we provide a notation for the latter.

Definition 3.5.1. *We will write $\bar{f}^F$ for the remainder on division of f by the ordered s-tuple $F = (f_1, ..., f_s)$. If F is a Gröbner basis for $(f_1, ..., f_s)$ then we regard F as a set without any particular order (by Proposition 2).*

To be able to determine whether a generating set $G = \{g_1, ..., g_t\}$ is a Gröbner basis, we remember the definition of a Gröbner basis (Definition 3.4.3). We deduce that G is not a Gröbner basis when there is an element $f \in \langle g_1, ..., g_t \rangle$ whose leading term is not divisible by $\mathrm{LT}(g_i)$ for some i. Consider for example the polynomial $f = ax^\alpha g_j + bx^\beta g_k$, with a and b such that $a \cdot \mathrm{LT}(x^\alpha) \cdot \mathrm{LT}(g_j) = -b \cdot \mathrm{LT}(x^\beta) \cdot \mathrm{LT}(g_k)$. This causes a cancellation of the leading terms so that it is not necessarily true that $\mathrm{LT}(f)$ is divisible by some $\mathrm{LT}(g_i)$. This implies that G possibly is not a Gröbner basis. The next definition takes two polynomials to construct such a combination with vanishing leading terms: the S-polynomial.

Definition 3.5.2. *Let $f, g \in k[x_1, ..., x_n]$ be nonzero polynomials.*

1. *If $\mathrm{multideg}(f) = \alpha$ and $\mathrm{multideg}(f) = \beta$, then let $\gamma = (\gamma_1, ..., \gamma_n)$, where $\gamma_i = \max(\alpha_i, \beta_i)$ for each i. We call x^γ the **least common multiple** of $\mathrm{LM}(f)$ and $\mathrm{LM}(g)$, written $x^\gamma = \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))$.*

2. *The **S-polynomial** of f and g is the combination*

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

The following lemma shows that for polynomials of the same degree, every cancellation of leading terms is a result from the kind of cancellations defined by Definition 3.5.2. This fact, in turn, leads to a criterion for Gröbner basis based on the remainder on division of certain crucial S-polynomials. For the interested reader we refer to [6, p. 84-87] for more details.

Lemma 1. *Suppose we have a sum* $\sum_{i=1}^{s} c_i f_i$*, where* $c_i \in k$ *and* $\text{multideg}(f_i) = \delta \in \mathbb{Z}_{\geq 0}^n$ *for all* i*. If* $\text{multideg}(\sum_{i=1}^{s} c_i f_i) < \delta$*, then* $\sum_{i=1}^{s} c_i f_i$ *is a linear combination, with coefficients in* k*, of the S-polynomials* $S(f_j, f_k)$ *for* $1 \leq j, k \leq s$*. Furthermore, each* $S(f_j, f_k)$ *has multidegree* $< \delta$*.*

Proof. See [6, p. 84]. □

Theorem 3.6 (Buchberger's Criterion)**.** *Let* I *be a polynomial ideal. then a basis* $G = \{g_1, ..., g_t\}$ *for* I *is a Gröbner basis for* I *if and only if for all pairs* $i \neq j$*, the remainder on division of* $S(g_i, g_j)$ *by* G *(listed in some order) is zero.*

Proof. See [6, p. 85-87]. □

4 Theoretical Results

The previous chapters introduced the algebraic definition of SAT and presented the theory of Gröbner basis for general ideals I and fields k. In this chapter we combine these two things by applying the theory of Chapter 4 to the satisfiability problem defined in Chapter 3. Our goal is to develop a theoretical framework that enables us to check the satisfiability of a propositional formula. We therefore expand the search for a common zero of the ideal $I = \langle f, x_1^2+x_1, ..., x_n^2+x_n \rangle$, which is done in subsection 4.1. Second, we look into the consequences of certain logical concepts on the algebraic representation. It could provide us with information about some kind of instances that can be ignored for our investigation of SAT in Maple. This part of logical consequences is included in the subsection 4.2.

4.1 Algebraic analysis

We discuss several algebraic methods to find a common zero for an ideal I. Each approach is presented in a subsection and is applied to the ideal $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ in $\mathbb{F}_2[x_1, ..., x_n]$. Although not every approach ends in a useful result, we finally detect a good method to determine the satisfiability of a polynomial. To facilitate this method, we have added a subsection to discuss Buchberger's criterion again.

4.1.1 Radicals

Definition 4.1.1. *The* ***radical*** *of an ideal* I *in* $k[x_1, ..., x_n]$*, denoted by* $\sqrt{I}$*, is defined as*

$$\sqrt{I} = \{f \in k[x_1, ..., x_n] \mid f^n \in I \text{ for some integer } n \geq 1\}.$$

Definition 4.1.2. *An ideal I is **radical** if $\sqrt{I} = I$.*

The reason of introducing the definition of the radical of an ideal is the following property,

$$V\big(\sqrt{I}\big) = V(I).$$

By constructing $\sqrt{I}$ we obtain an ideal which contains I and defines the same variety. Considering $\sqrt{I}$ instead of I may simplify the search for a common zero of I.

Lemma 2. *$I \subset R$ is radical if and only if R/I does not contain nonzero nilpotents.*

Proof. ($\Rightarrow$). Assume that I is radical, but that R/I contains a nonzero nilpotent. Then there exists a nonzero $f \bmod I \in R/I$ such that $(f \bmod I)^n = 0$. This implies $f^n \bmod I = 0$ and thus $f^n \in I$. Since $I = \sqrt{I}$ we also have $f \in I$, which implies that $f \bmod I = 0$. This contradicts the assumption, thus we conclude that R/I does not contain nonzero nilpotents.
($\Leftarrow$). R/I does not contain nonzero nilpotents. To proof that I is radical, we have to proof that for each $f^n \in$, f is also contained in I. $f^n \in I$ means that $f^n \bmod I = (f \bmod I)^n = 0$. Since R/I does not contain nonzero nilpotents $f \bmod I = 0$ so $f \in I$. □

Theorem 4.1. *For each ideal $I \subseteq \mathbb{F}_2[x_1, ..., x_n] = R$ such that $x_j^2 + x_j \in I$ for each j, R/I does not contain nonzero nilpotents.*

Proof. Since $J = \langle x_1^2 + x_1, ..., x_n^2 + x_n \rangle \subset I$ we have by the third isomorphism theorem that

$$R/I \cong (R/J)/(I \bmod J). \tag{7}$$

Any element x in the quotient ring R/J has the property $x^2 = x$, so $x^m = x$ for any positive integer m. That means that R/J does not contain nonzero nilpotents. We conclude from Equation 7 that R/I neither contains nonzero nilpotents. □

From these results we conclude that the ideal $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle \subset \mathbb{F}_2[x_1, ..., x_n]$ is radical. Since we showed $I = \sqrt{I}$, working with the radical It turns out that this approach does not give us any advantage in order to compute the zeros of I.

4.1.2 The Chevalley-Warning theorem

In algebra there exists a theory that guarantees certain polynomial equations to have solutions. It was proved by Ewald Warning [13] and a slightly weaker form was proved by Claude Chevalley [4], both in 1935.

Theorem 4.2 (Chevalley-Warning)**.** *Let $\mathbb{F}$ be a finite field and $\{f_j\}_{j=1}^r \subseteq \mathbb{F}[x_1, ..., x_n]$ be a set of polynomials such that the number of variables satisfies*

$$n > \sum_{j=1}^{r} d_j$$

where d_j is the total degree of f_j. Then for the system of polynomial equations

$$f_j(x_1, ..., x_n) = 0 \text{ for } j = 1, ..., r$$

the number of common solutions $(a_1, ..., a_n) \in \mathbb{F}^n$ is divisible by the characteristic p of $\mathbb{F}$. Or in other words, the cardinality of the vanishing set of $\{f_j\}_{j=1}^r$ is $0 \bmod p$.

Unfortunately, this theorem is not applicable for our general system of equations

$$f(x_1, ..., x_n) = x_1^2 + x_1 = \ldots = x_n^2 + x_n = 0.$$

The last n equations have a total degree of 2, so $\sum_{j=1}^r d_j \geq 2n$. The requirement $n > \sum_{j=1}^r d_j$ therefore cannot be met and we focus on other approaches.

Of course, since we are only interested in solutions in $\mathbb{F}_2^n$, we can ignore the equations $x_i^2 + x_i = 0$ since they are automatically satisfied. That implies we are left with the equation $\bar{f} = 0$ in which $\bar{f}$ has degree $\leq n$, and degree $< n$ precisely when the term $x_1 x_2 \cdots x_n$ does not occur. Assuming we are in this situation, Chevalley-Warning implies that the number of zeros in $\mathbb{F}_2^n$ is even. Unfortunately, this does not help since the difference between 0 solutions and a positive even number of solutions can not be seen from it.

4.1.3 Optimization over $\mathbb{Z}[x_1, .., x_n]$

For now we consider just the zero of a polynomial $f \in \mathbb{F}_2[x_1, ..., x_n]$. We treat this polynomial as the reduction modulo 2 of a polynomial $\tilde{f}$ in $\mathbb{Z}[x_1, ..., x_n]$ with coefficients in $\{0, 1\}$. So when we evaluate the polynomial for an appropriate truth assignment, $\alpha \in \{0, 1\}^n$, we possibly get different outcomes of $\tilde{f}(\alpha)$ and $f(\alpha)$. We know that f is satisfiable if and only if $f(\alpha) = 0$ for some α. This is equivalent to requiring that $\tilde{f}(\alpha)$ is a multiple of 2 for some α. Since $\tilde{f}(\alpha)$ and $\left(\tilde{f}(\alpha)\right)^2$ have the same parity, it suffices to check the parity of $\left(\tilde{f}(\alpha)\right)^2 \in \mathbb{Z}_{\geq 0}$ in order to determine the satisfiability of f. In Definition 4.1.3 we use the multiplicative order $ord_2(x)$ which tells how often x can be divided by 2. We set $ord_2(0) = \infty$.

Definition 4.1.3. *The function F: $\{0, 1\}^n \to \mathbb{Z}_{\geq 0}$ is defined by*

$$\alpha \mapsto \frac{1}{ord_2\left(\left(\tilde{f}(\alpha)\right)^2\right)}. \tag{8}$$

The range of this function is the set $\{\infty, 0, 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, ...\}$. Minimizing this function will give us information about the satisfiability of the formula f. We observe that

- $\min(F) = 0 \Leftrightarrow$ there exists an α with $\left(\tilde{f}(\alpha)\right)^2 = 0$;
- $\min(F) = \infty \Leftrightarrow \left(\tilde{f}(\alpha)\right)^2$ is odd for all α;
- $\min(F) = \frac{1}{n} \Leftrightarrow$ there exists an α such that $\left(\tilde{f}(\alpha)\right)^2 = 2^n \cdot odd$ and for all other nonzero $\beta \in \{0, 1\}^n$ we have $\left(\tilde{f}(\beta)\right)^2 = 2^m \cdot odd$ with $m < n$.

From these observations we can conclude that if the minimum of F is ∞, the logic formula represented by f is not satisfiable. So if we are able to minimize the function in Equation 8, we obtain a way to determine whether a formula is satisfiable. Unfortunately, the function F is not a suitable optimization function. We conclude that this approach also is not successful.

4.1.4 Hilbert's Nullstellensatz

From experiments in Maple, we presumed that the non-existence of a zero for the ideal is related to the appearance of 1 in the ideal. This leads us to the Hilbert's Nullstellensatz. The theorem cannot immediately be applied to our case, but we will derive that it also holds for the ideals $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ in $\mathbb{F}_2[x_1, ..., x_n]$.

Theorem 4.3 (Hilbert's Nullstellensatz)**.** *Let k be an algebraically closed field and let $I \subset k[x_1, ..., x_n]$. Then $V(I) = \varnothing \Leftrightarrow I = k[x_1, ..., x_n]$.*

Proof. See [9]. □

The stumbling point here is that the field k should be algebraically closed. Because $\mathbb{F}_2$ does not satisfy this condition, we introduce its algebraic closure,

$$\overline{\mathbb{F}_2} = \bigcup_{n=1}^{\infty} \mathbb{F}_{2^{n!}}.$$

Consider the ideal $\overline{I} = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ in $\overline{\mathbb{F}_2}[x_1, ..., x_n]$. It follows from Hilbert's Nullstellensatz that $V(\overline{I}) = \varnothing$ if and only if $1 \in \overline{I}$. For general ideals $I \subset \mathbb{F}_2[x_1, ..., x_n]$ the condition $V(I) = \varnothing$ does not necessarily imply that $1 \in I$. Consider for example the ideal $\langle 1 + x + x^2 \rangle$ in $\mathbb{F}_2[x]$. Since $1 + x + x^2$ does not have a zero in $\mathbb{F}_2$, it follows from Theorem 3.1 that $V(I) = \varnothing$. Nevertheless, $1 \notin \langle 1 + x + x^2 \rangle$. However, we can show that for the ideals I representing the satisfiability problem, $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$, the variety $V(I)$ is empty if and only if $1 \in I$. This is done in the remainder of this subsection.

Theorem 4.4. *Let $\overline{\mathbb{F}_2}$ be the algebraic closure of $\mathbb{F}_2$. Given $f \in \mathbb{F}_2[x_1, ..., x_n]$ consider the ideals $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ in $\mathbb{F}_2[x_1, ..., x_n]$ and $\overline{I} = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ in $\overline{\mathbb{F}_2}[x_1, ..., x_n]$. Then $V(I) = \varnothing$ if and only if $V(\overline{I}) = \varnothing$.*

Proof. We prove that $V(I) \neq \varnothing \Leftrightarrow V(\overline{I}) \neq \varnothing$. It then follows that $V(I) = \varnothing$ if and only if $V(\overline{I}) = \varnothing$.
($\Rightarrow$). When $V(I) \neq \varnothing$ there exists an $a \in \mathbb{F}_2^n$ such that $f(a) = 0$ for all $f \in I$. Let h be an element h of $\overline{I}$. Then by the definition of an ideal it can be written as $h = g_0 \cdot f + \sum_{i=1}^{n} g_i \cdot (x_i^2 + x_i)$ where each $g_i \in \overline{\mathbb{F}_2}[x_1, ..., x_n]$. It follows that $h(a) = \sum_{i=0}^{n} g_i \cdot 0 = 0$ and since $a \in \mathbb{F}_2^n \subset \overline{\mathbb{F}_2}^n$ we can conclude that $a \in V(\overline{I})$.
($\Leftarrow$). Assume that $V(I) = \varnothing$ while $V(\overline{I}) \neq \varnothing$. By the definition of varieties it follows that each $a \in V(\overline{I})$ is an element of $\overline{\mathbb{F}_2}^n \setminus \mathbb{F}_2^n$. Since $x_1^2 + x_1, ..., x_n^2 + x_n$ are contained in $\overline{I}$, a zero a must satisfy $a_i^2 + a_i = 0$ for each i. This condition implies that each $a_i \in \{0, 1\}$, which contradicts the fact that $a \notin \mathbb{F}_2^n$. We thus have by contradiction that $V(I) \neq \varnothing$. □

Corollary 2. *$V(I) = \varnothing$ if and only if $1 \in I$.*

Proof. From Theorems 4.3 and 4.4 it follows that $V(I) = \varnothing \Leftrightarrow V(\overline{I}) = \varnothing \Leftrightarrow 1 \in \overline{I}$. We use these equivalences to prove that $1 \in I \Leftrightarrow V(I) = \varnothing$.
($\Rightarrow$). Trivial: $1 \in I$ does not have a zero.
($\Leftarrow$). Notice that $I = \overline{I} \cap \mathbb{F}_2[x_1, ..., x_n]$. Thus if $1 \in \overline{I}$ it follows that $1 \in I$. From Theorem 4.3 and 4.4 we can conclude that $V(I) = \varnothing \Leftrightarrow 1 \in \overline{I}$. □

We finally can use our results in order to redefine the satisfiability of a polynomial f.

Definition 4.1.4. *The polynomial $f \in \mathbb{F}_2[x_1, ..., x_n]$ is satisfiable if and only if $1 \in I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$.*

4.1.5 Buchberger's Criterion

The main results from the theory about Gröbner bases are the solution to the ideal membership problem (Corollary 1, Chapter 3) and Buchberger's criterion (Theorem 3.6). Combining these outcomes and applying them to the ideal $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$ leads to a simplified version of Buchberger's criterion in Theorem 4.5.

Lemma 3. *Let G be the ordered n-tuple $\{x_1^2 + x_1, ..., x_n^2 + x_n\}$. Then $\overline{S(x_i^2 + x_i, x_j^2 + x_j)}^G = 0$ for $1 \leq i, j \leq n$.*

Proof. Using the division algorithm for multivariate polynomials we can divide $S(x_i^2 + x_i, x_j^2 + x_j) = x_j^2 x_i + x_i^2 x_j$ by the ordered set G. We obtain $S(x_i^2 + x_i, x_j^2 + x_j) = x_i(x_j^2 + x_j) + x_j(x_i^2 + x_i) + 0$ which implies that $\overline{S(x_i^2 + x_i, x_j^2 + x_j)}^G = 0$. □

Theorem 4.5. *Let $I = \langle f, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$. Then the basis $G = \{f, x_1^2 + x_1, ..., x_n^2 + x_n\}$ is a Gröbner basis for I if and only if for all i, the remainder on division of $S(f, x_i^2 + x_i)$ by the ordered set $\{x_1^2 + x_1, ..., x_n^2 + x_n, f\}$ is zero.*

Proof. ($\Rightarrow$). For a Gröbner basis G we know that $h \in I$ if and only if $\overline{h}^G$ is zero. Each S-polynomial is by definition contained in the ideal I and thus has a zero remainder.
($\Leftarrow$). Theorem 3.6 states that if $\overline{S(g_i, g_j)}^G = 0$ for all pairs $i \neq j$, then the basis G is a Gröbner basis. Note that we assume $\overline{S(f, x_i^2 + x_i)}$ to be zero and that $\overline{S(g_i, g_j)}^G = 0$ if and only if $\overline{S(g_j, g_i)}^G = 0$. It therefore suffices to prove that for each pair $i \neq j$, $\overline{S(x_i^2 + x_i, x_j^2 + x_j)}^G = 0$. Due to the ordering of G the S-polynomial is first divided by all $x_i^2 + x_i$ and from Lemma 3 we know that this division results in a zero remainder. We conclude that $\overline{S(x_i^2 + x_i, x_j^2 + x_j)}^G = 0$ for each pair i, j. □

We already found that Gröbner bases form the good generating sets for an ideal and we want to determine whether or not the generating set $G = \{f, x_1^2 + x_1, ..., x_n^2 + x_n\}$ of I is already such a good set. To do that, we do not use a monomial order to order the elements of G, but we choose as ordered set $G_{ord} = \{x_1^2 + x_1, ..., x_n^2 + x_n, f\}$. From Theorem 4.5 it then follows that we only have to check $\overline{S(f, x_i^2 + x_i)}^G$ for each i, in order to recognize Gröbner bases.

4.2 Logical Properties

We are interested in possible differences in the algebraic representation of propositional formulas. The first step is to compare their representing formulas, which is done for CNF and DNF formulas in the first subsection. We furthermore consider Horn formulas and develop a Horn algorithm for polynomials. Finally, the interesting question is considered whether every polynomial represents a propositional formula.

4.2.1 CNF vs. DNF

Every DNF-formula can be transformed to a CNF-formula using the De Morgan laws and the distributive laws for conjunctions and disjunctions. Although the polynomial representing a CNF-formula initially could have a different structure then the polynomial for a DNF-formula, there is no longer a difference when we transform them to $\mathbb{F}_2[x_1, ..., x_n]/(x_1^2 + x_1, ..., x_n^2 + x_n)$. To confirm this claim, we show that the algebraic variants of the above laws holds.

Theorem 4.6. *For propositional formulas P, Q and R we have the following:*

- $\bar{f}_{P\wedge(Q\vee R)} = \bar{f}_{(P\wedge Q)\vee(P\wedge R)}$;
- $\bar{f}_{P\vee(Q\wedge R)} = \bar{f}_{(P\vee Q)\wedge(P\vee R)}$;
- $\bar{f}_{\neg(P\wedge Q)} = \bar{f}_{\neg P\vee\neg Q}$;
- $\bar{f}_{\neg(P\vee Q)} = \bar{f}_{\neg P\wedge\neg Q}$.

Proof. We provide a proof for the first equation. The other equations can be verified in the same way. Using the definition of polynomials (Equation 2 of Chapter 2), we obtain (with the notation $\bar{f}_P = p, \bar{f}_Q = q$ and $\bar{f}_R = r$):

$$\bar{f}_{P\wedge(Q\vee R)} = p \cdot ((q+1)(r+1)+1) = p(qr+q+r+2) = pqr + pq + pr,$$
$$\bar{f}_{(P\wedge Q)\vee(P\wedge R)} = (pq+1)(pr+1)+1 = p^2qr + pq + pr + 2 = pqr + pq + pr.$$

So indeed $\bar{f}_{P\wedge(Q\vee R)} = \bar{f}_{(P\wedge Q)\vee(P\wedge R)}$. □

It follows that for equivalent formulas, P_{CNF} and P_{DNF}, the bases $\{\bar{f}_{P_{CNF}}, x_1^2+x_1, ..., x_n^2+x_n\}$ and $\{\bar{f}_{P_{DNF}}, x_1^2+x_1, ..., x_n^2+x_n\}$ are the same. They therefore define the same ideal and are simultaneously a Gröbner basis. It therefore makes no sense to compare their algebraic representations to each other in the investigation in Maple.

4.2.2 Horn algorithm

As mentioned in Section 2.2, Horn sentences form an important subclass of the propositional formulas with respect to their satisfiability. Considering the polynomial of a formula, we can

determine whether it is equivalent to a Horn formula or not. The specific properties of these polynomials explain why the Horn satisfiability algorithm works well.

Let P be a Horn sentence consisting of disjunctions $D_1, ..., D_m$ and atoms $p_1, ..., p_n$. Since $f_P = \prod_{i=1}^{m} f_{D_i}$, we know that P is satisfiable if there exists an $a \in \mathbb{F}_2^n$ such that $f_{D_i}(a) = 1$ for each i. To determine special properties for Horn sentences, we define two disjoint sets of the variables x_i representing atoms p_i:

$$D_i^+ = \{x_i \mid p_i \text{ appears in a positive literal of } D_i\};$$
$$D_i^- = \{x_i \mid p_i \text{ appears in a negative literal of } D_i\}.$$

Since P is a Horn sentence we have that each D_i^+ contains at most one element. In order to be able to distinguish between the variable that occurs in the positive literal and the other variables contained in D_i, we indicate $x_j \in D_i^+$ by y_i if $D_i^+ \neq \varnothing$. For i such that $D_i^+ = \varnothing$ we set $y_i = 0$. This notation allows us to define the polynomial for the conjuncts D_i of a Horn sentence:

$$f_{D_i}(x_1, ..., x_n) = 1 + (1 + y_i) \cdot \prod_{x_j \in D_i^-} x_j.$$

Using this definition we can define the Horn satisfiability algorithm for the algebraic case. For readibility we set $A_i := (1 + y_i) \cdot \prod_{x_j \in D_i^-} x_j$.

Horn algorithm for polynomials

Given Horn polynomial $f_P = \prod_{i=0}^{m} f_{D_i} = \prod_{i=0}^{m} (1 + A_i)$ follow these steps:

1. Consider polynomials f_{D_i} such that A_i is composed of one factor.
   (a) If $f_{D_i} = f_{D_j} + 1$ for two such polynomials **stop**; f_P is not satisfiable.
   (b) Else choose an f_{D_i} and go to step 2.
   (c) Otherwise go to step 3.
2. Assign a value in $\mathbb{F}_2$ to the variable x_j that appears in f_{D_i}, such that $f_{D_i} = 1$.
   (a) If A_i appears as a factor of some A_j, then remove the factor f_{D_j} from f_P;
   (b) If $1 + A_i$ appears as a factor of some A_j, then remove $(1 + A_i)$ from A_j;
3. Go to step 1 if there is still a polynomial f_{D_i} such that A_i is composed of one factor. Otherwise assign 0 to all unassigned variables and **stop**; P is satisfiable.

In principle this is the same algorithm as the Horn algorithm for propositional formulas, in the sense that it takes analogous steps. Since the first is a linear-time algorithm, we have found a linear time algorithm to determine the complexity of Horn polynomials. We therefore do not give special attention to this type of polynomials in the next chapter.

4.2.3 Polynomials representing propositional formulas

Finally, we wonder whether every polynomial $f \in \mathbb{F}_2[x_1, ..., x_n]/(x_1^2 + x_1, ..., x_n^2 + x_n)$ represents a propositional formula. To answer this question we observe that a polynomial is build through addition and multiplication of the variables $x_1, ..., x_n$ and 1. Providing a translation for these two operations enables us to translate each polynomial in $\mathbb{F}_2[x_1, ..., x_n]/(x_1^2 + x_1, ..., x_n^2 + x_n)$ to a propositional formula. The next equations define the propositional formula P corresponding to an operation on the polynomials f and g in $\mathbb{F}_2[x_1, ..., x_n]/(x_1^2 + x_1, ..., x_n^2 + x_n)$:

$$\begin{aligned} P_{f \cdot g} &= P_f \wedge P_g; \\ P_{f+g} &= (P_f \wedge \neg P_g) \vee (\neg P_f \wedge P_g). \end{aligned} \tag{9}$$

In addition we define $P_{x_i} = p_i$. Then $v(P_{x_i}) = \textit{true}$ if and only if $x_i = 1$, and $v(P_{x_i}) = \textit{false}$ if and only if $x_i = 0$. One can verify that a truth assignment determines $v(P_f) = \textit{true}$ or $\textit{false}$ if and only if its corresponding vector $x \in \mathbb{F}_2^n$ leads to $f(x) = 1$ or 0, respectively.

Example. We determine the propositional formula belonging to $x_1x_2 + x_2$ using the rules in Equation 9:

$$\begin{aligned} P_{x_1x_2+x_2} &= (P_{x_1x_2} \wedge \neg P_{x_2}) \vee (\neg P_{x_1x_2} \wedge P_{x_2}) \\ &= (P_{x_1} \wedge P_{x_2} \wedge \neg P_{x_2}) \vee (\neg(P_{x_1} \wedge P_{x_2}) \wedge P_{x_2}). \end{aligned}$$

5 Maple Results

The first result is an algorithm that enables us to check the satisfiability of a random propositional formula using Maple. It is based on the algebraic consequences presented in Chapter 5 and is defined as follows:

Algorithm I

Step 1	Translation of propositional formula P to a polynomial f;
Step 2	Transposition of f to $\bar{f} = f + 1 \in \mathbb{F}_2[x_1, ..., x_n]/(x_1^2 + x_1, ..., x_n^2 + x_n)$;
Step 3	Definition of ideal $\langle \bar{f}, x_1^2 + x_1, ..., x_n^2 + x_n \rangle$;
Step 4	Determine whether or not $\{\bar{f}, x_1^2 + x_1, ..., x_n^2 + x_n\}$ is a Gröbner basis;
Step 5	Computation of Gröbner basis if step 4 returns false;
Step 6	Verification of satisfiability by determining whether $1 \in I$.

Table 1 provides a comparison between the running time of the steps of the algorithm. It measures the time it takes to execute a step for a propositional formula of n variables. The displayed results are the average of at least 10 executions of the algorithm for a propositional formula of n variables. The second step of the algorithm can be executed in two different ways, which is denoted by Step 2.1 and Step 2.2. The implementations can be found in the attachment.

Table 1: Running time (in seconds) of executing the steps of Algorithm I for a propositional formula of n variables.

n	number of terms of f	Step 1	Step 2.1	Step 2.2	Step 3	Step 4	Step 5	Step 6
3	33	0.00	0.48	0.00	0.00	0.00	0.00	0.00
4	1 330	0.00	0.54	0.29	0.00	0.00	0.00	0.00
5	140 924	0.00	324.39	440.27	0.00	0.17	0.00	0.00
6	27 792 050	0.96	> 3705	–	–	–	–	–

Second, a shortcut algorithm is defined in order to establish the impact of different monomial orderings on the running time. On the basis of the results in Table 1, the algorithm is executed for random propositional formulas with $n = 4$ and $n = 5$. The results, obtained by execution of Algorithm II, can be found in Tables 2 and 3, respectively.

Algorithm II

Step 1 Translation of propositional formula P to a polynomial f;

Step 2 Computation of Gröbner basis;

Step 3 Verification of satisfiability by determining whether $1 \in I$.

Table 2: Running time (in seconds) of executing Algorithm II for different monomial orderings.

n	lex	grlex	grevlex
4	0.00	0.67	0.00
4	0.00	0.42	0.00
4	0.00	0.73	0.00
4	0.00	0.52	0.00
4	0.00	1.12	0.00
4	0.00	0.73	0.00
4	0.00	0.00	0.00
4	0.00	0.69	0.00
4	0.00	0.56	0.00
4	0.00	0.67	0.00
average	0.00	0.61	0.00

Table 3: Running time (in seconds) of executing Algorithm II for different monomial orderings.

n	lex	grlex	grevlex
5	0.64	34.62	0.56
5	149.70	151.93	150.94
5	1.06	53.06	0.99
5	0.63	31.68	0.58
5	111.24	113.49	113.24
5	126.97	123.76	120.81
5	1.06	93.93	1.02
5	1.55	134.42	132.29
5	0.59	3.48	0.56
5	1.03	60.73	1.06
average	39.44	80.11	52.20

6 Conclusion

Due to Gröbner bases, Buchberger's criterion and Hilbert's Nullstellensatz, we obtained two algebraic algorithms to solve SAT: Algorithm I and Algorithm II. The latter is the fastest, especially when executed using the lex or grevlex order. It appears that the grlex order is least qualified to check satisfiability using Algorithm II. The first algorithm, Algorithm I, divides the procedure of checking satisfiability in more steps, which was useful to analyse the process and to develop Algorithm II. We can conclude that the major part of the running time of executing Algorithm I is spent on running Step 2. Finally, we observe that the number of terms of $\bar{f}$ grows fast when

increasing the number of propositional variables. This growing number of terms is part of the input size of Step 2.1 and Table 1 shows that its running time is proportional to the number of terms.

7 Discussion

To obtain the results on the running time of the algorithm, we executed the algorithm for random propositional formulas of given number of variables. The disadvantage of the function[2] in Maple that generates random formulas, is that it only provides satisfiable formulas. It was therefore hard to examine the algebraic consequences of unsatisfiable formulas by running a large amount of unsatisfiable instances. For future research we recommend to use benchmarks, which can be found in, for example, [15].
At the end of studying SAT, a global proof was found for the fact that a polynomial f is satisfiable if and only if $\bar{f} = 1$. Due to lack of time the details of these results are not contained in this thesis.

finally we want to give a suggestion for further research. From the results it follows that it takes a lot more time to compute a Gröbner basis for the ideal $\langle f, x_1^2 + x_1, ..., x_n + x_n^2 \rangle$ than for $\langle \bar{f}, x_1^2 + x_1, ..., x_n + x_n^2 \rangle$. One can consider the question if it is possible to find an algebraic explanation for this result. It is also interesting to apply the same approach to some special classes of SAT and to investigate whether algebraic concepts like Gröbner bases can be useful to determine the satisfiability of such a class. For different approaches to SAT see for example [12].

References

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

[2] Dave Barker-Plummer, Jon Barwise, and John Etchemendy. *Language, Proof and Logic.* CSLI Publications, 2011.

[3] Peter Brucker. *Scheduling Algorithms.* Springer, 2007.

[4] Claude Chevalley. Démonstration d'une hypothèse de M. Artin. *Abh. Math. Semin. Univ. Hamb.*, 11:73–75, 1935.

[5] Stephen A. Cook. The complexity of theorem-proving procedures. *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[6] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties and Algorithms.* Springer, 1992.

[7] Leonard Eugene Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913.

[2] http://www.maplesoft.com/support/help/maple/view.aspx?path=Logic/Random

[8] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267 – 284, 1984.

[9] David Hilbert. Ueber die vollen Invariantensysteme. *Mathematische Annalen*, 42:313–373, 1893.

[10] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[11] Michael L. Pinedo. *Scheduling. Theory, Algorithms, and Systems*. Springer, 2008.

[12] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science (Vol. A): Algorithms and Complexity*. MIT Press, Cambridge, MA, USA, 1990.

[13] Ewald Warning. Bemerkung zur vorstehenden Arbeit von Herrn Chevalley. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg (in German)*, 11:76–83, 1935.

[14] Herbert S. Wilf. *Algorithms and complexity*. A K Peters, Ltd, 2002.

[15] Ke Xu. Forced satisfiable CSP and SAT benchmarks of model RB.

A Maple Implementation

First we present two procedures in Maple, *DA* and *S*, computing the remainder on division and the S-polynomial respectively. These two functions are used in the implementation of Step 4. After these implementations, all the procedures belonging to Step 1 till 6 of Algorithm I are displayed in order of their execution. Finally, this appendix contains the code, called *SAT*, for the short Algorithm II.

```
DA := proc (G1 :: list, f, ord, L :: list)
      uses Groebner, ListTools
      local G, n, i, g, q, LG, P, p, r, d, j;
       G := G1:
        n := numelems(G):
        m := numelems(L):
       G := sort(G, (a, b) → TestOrder(a, b, ord(op(Reverse(L))))):
      for i to n do
        q_i := 0:
        g_i := G_i:
        LG_i := LeadingCoefficient(g_i, ord(op(L))) · LeadingMonomial(g_i, ord(op(L))):
       end do:
      p := f:
      r := 0:
      while p ≠ 0 do
          j := 1:
          d := false:
          while j ≤ n and d = false do
            P := LeadingCoefficient(p, ord(op(L))) · LeadingMonomial(p, ord(op(L))):
            if divide(P, LG_j) then
              q_j := q_j + P/LG_j:
              q_j := mod(simplify(q_j), 2):
              p := p - P/LG_j · g_j:
              p := mod(simplify(p), 2):
              d := true:
              else j := j + 1:
              end if:
              end do:
            if d = false then
             r := r + P:
             p := p - P:
             end if:
         end do:
r;
end proc:
```

```
S := proc(f, g, ord, L :: list)
      uses Groebner
      local F, G, n, x, i, s, S;
        F := LeadingMonomial(f, ord(op(L))) :
        G := LeadingMonomial(g, ord(op(L))) :
        n := numelems(L) :
        x := 1 :
        for i to n do
        alpha(i) := max(degree(F, L_i), degree(G, L_i));
        x := x · (L_i)^alpha(i);
        end do:
        s := (x/F · f - x/G · g) :
        S(f, g) := mod(simplify(s), 2);
    end proc :


Step1 := proc(e)
              uses Logic;
              local F;
               F :=mod(Export(e, form = MOD2), 2);
            end proc:


Step2.1 :=proc(F, ord, L)
         uses Groebner
         local f, fnew, n, m, g, gnew, j;
         f :=mod(expand(F), 2) :
         fnew := 0 :
         n := numelems(L) :
         while f ≠ 0 do
             g := op(1, f);
             f := f − g :
             gnew := 1 :
              for j to n do
                 if degree(g, x_j) ≠ 0 then
                    gnew := gnew · x_j :
                 end if:
               end do:
             fnew := fnew + gnew;
          end do:
          fnew :=mod(fnew + 1, 2) :
          end proc:


Step2.2 :=proc(F, ord, L)
         uses Groebner;
         local n, fnew;
         n := numelems(L) :
         fnew := NormalForm(F, [seq(x_i^2 − x_i, i = 1 ..n)], ord(op(L)), characteristic = 2) :
        fnew :=mod(fnew + 1, 2);
        end proc:
```

```
Step3 := proc(f, L)
        uses PolynomialIdeals
        local n, J :
        n := numelems(L) :
        J := [ f, seq(x_i^2 + x_i, i = 1 ..n) ];
        end proc:

Step4 := proc(G :: list, ord, L :: list)
         local n, i, s, j, rest, groebner;
         groebner := true :
         n := numelems(G) :
         for i from 2 to n do
             s := S( G_i , G_1 , ord, L) :
             rest := DA(G, s, ord, L);
              if rest ≠ 0 then
                   groebner := false;
                   break;
              end if;
         end do:
       if groebner = true then print(yes) end if;
       if groebner = false then print(no) end if;
       end proc:

Step5 := proc(G :: list, ord, L :: list, check)
          uses Groebner, PolynomialIdeals;
          local newb, Gbasis;
            if check = no then
              newb := ⟨op(Basis(⟨op(G)⟩ mod 2, ord(op(L))))⟩ mod 2;
            elif check = yes then
              newb := ⟨op(G)⟩ mod 2;
            end if;
            Gbasis := newb;
          end proc:

Step6 := proc(ideal)
           uses PolynomialIdeals;
           local i;
           i := IdealMembership(1, ideal mod 2) :
           if i = true then i := false ;
           elif i = false then i := true;
           end if;
         end proc :

SAT := proc(e, ord, n)
         uses Groebner, Logic, PolynomialIdeals;
         local f, G, M;
         f := Export(e, form = MOD2) + 1 :
         G := ⟨ Basis( ⟨[ f, seq(x_i^2 + x_i, i = 1 ..n) ]⟩ mod 2, ord(seq(x_i, i = 1 ..n)), [seq(x_i, i = 1 ..n) ]) mod 2;
         M := IdealMembership(1, G);
         if M = true then M := unsatisfiable;
            elif M = false then M := satisfiable;
         end if;
         print(M, G);
         end proc:
```