Manuel Hauke, BSc

Signature Gröbner bases
A comprehensive survey and
a new algorithmic approach

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mathematics

submitted to

Graz University of Technology

Supervisors

Christian Rechberger, Univ.-Prof. Dipl.-Ing. Dr.techn.

Institute of Applied Information Processing and Communications, Graz

Graz, August 2020

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

31.08.2020 Manuel Hanke

Date, Signature

Abstract

The computation of Gröbner bases is an often-used tool in modern cryptography. With the upcoming trend of quantum computing, systems based on Gröbner bases will most likely become even more relevant in the near future since the majority of them are assumed to be quantum secure. Doing a detailed examination of existent algorithms and finding possible improvements helps for estimating the security of such systems. Apart from cryptographic applications, Gröbner bases are also used in many different areas and hence, an efficient computation is crucial. For that reason, the research on the computation of Gröbner bases has become a large area in Commutative Algebra in the last two decades.
The aim of this thesis is split into two parts: In the first part, we give a comprehensive overview of the state of the art of current Gröbner basis algorithms. We work out similarities and differences between already existing algorithms in a short, but mathematically rigorous manner. In this process, we close some proof gaps of existing theorems and find new ways how to describe those algorithms understandably.
In the second part, we develop a new algorithm which combines the ideas of two of the most used Gröbner basis algorithms. To do so, we extend the theory about signature Gröbner bases, find new properties and prove them. Furthermore, we implement this new approach and compare it with an often-used algorithm. The new approach seems to work very well for some of the tested ideals, however, we would need a more efficient implementation and further experiments for a fair comparison.

Kurzfassung

Gröbner-Basen und deren Berechnung spielen bereits jetzt in der modernen Kryptographie eine große Rolle, und werden, da die meisten darauf basierenden Kryptosysteme als quantensicher gelten, zukünftig wohl noch relevanter werden. Eine detaillierte Diskussion darüber hilft abzuschätzen, wie sicher diese Systeme tatsächlich sind. Neben der Kryptographie gibt es außerdem noch viele weitere Anwendungen, in denen die Berechnung einer Gröbner-Basis benötigt wird. Gerade in den letzten zwei Jahrzehnten hat sich ein großer Forschungsbereich entwickelt, der sich mit der effizienten Berechung von Gröbner-Basen beschäftigt.

Das Ziel dieser Arbeit besteht aus zwei Teilen: Einserseits werden bereits bekannte Ideen und Algorithmen in einer prägnanten, aber mathematisch rigorosen Weise zusammengefasst sowie auf Gleichheiten und Differenzen zwischen verschiedenen Ansätzen eingegangen. Dabei werden Beweislücken geschlossen sowie alternative Herangehensweisen definiert, die zum besseren Verständnis diverser Algorithmen führen sollen.

Andererseits wird die Idee eines neuen Algorithmus erklärt, der die Ansätze zweier bekannter und vielfach eingesetzter Algorithmen kombiniert. Dafür wird die Theorie der Signatur-Gröbner-Basen erweitert, wobei neue Eigenschaften gefunden und bewiesen werden. Weiters wird dieser Algorithmus implementiert und einem in der Praxis verwendeten Algorithmus gegenübergestellt. Für einige untersuchte Ideale erscheint dieser neu entwickelte Ansatz als vielversprechend, jedoch benötigt es für einen absoluten Vergleich eine effizientere Programmierung sowie weitere Forschung.

Acknowledgements

Before I start with my thesis, I would like to thank several people who have helped me to reach this point in my life. Without them, this would definitely not have been possible.
I would like to express my gratitude towards my supervisor Univ.-Prof. Christian Rechberger and my co-supervisor Assoc.Prof. Christian Elsholtz for their useful comments and remarks during the whole process of writing this master thesis. In this context, I especially would like to thank Reinhard Lüftenegger, MSc for his dedication to improving this thesis. I enjoyed the technical discussions we had as well as the motivational comments he gave me as a friend.
Furthermore, I would like to thank my fellow students for the numerous discussions and exercises we did together during the whole studies. Naming all of those explicitly would be beyond the scope of this acknowledgment, but I would like to mention Robert, Richard, Lorenz, Andreas and Thomas since those five people did not only accompany me during my studies, but also became close friends of mine.
Acknowledging friends that helped me with my studies, I definitely also have to mention my best friend and longtime roommate Herwig who shares my passion for science and motivated me during the years to study long hours. Similarly, I would like to express my gratitude to Philipp and Matthias who are close friends of mine with whom I could talk whenever I faced difficulties.
Of course, I would also like to thank my parents and grandparents who supported me financially over the last 24 years and thus made my studies possible. Their suggestions and considerations of my state of mind in stressful situations helped me to focus on my studies. Furthermore, my gratitude goes out to my sister Evita who helped me in the first years of university to get along.
Lastly, I also want to thank my girlfriend Anna for proofreading my thesis and especially for keeping me motivated as well as for having my back during the whole process of creating this thesis.

Manuel Hauke

Graz, 31. 8. 2020

Contents

Basic Notations

We will state some basic notations which will be used in the whole thesis in that way, if not stated differently:

- $\mathbb{N} = \{0, 1, ...\}$
- $\mathbb{N}_\infty := \mathbb{N} \cup \{\infty\}$
- $\mathbb{K}$ denotes an arbitrary field.
- $x := [x_1, ..., x_n]$.
- $P = P(n) := \mathbb{K}[x_1, ..., x_n]$
- $T = T(n) := \{x^a = x_1^{a_1} x_2^{a_2} ... x_n^{a_n}, a = (a_1, ..., a_n) \in \mathbb{N}^n\}$, the set of all multivariate terms.

1 Multivariate Quadratic polynomial systems and Gröbner bases

1.1 The Multivariate Quadratic problem

Looking for problems which are assumed to be hard even with quantum computing power, one possibility is to look at the following:

Problem 1.1.1 (Multivariate Quadratic(MQ))**.** Given a finite field $\mathbb{F}_q$, polynomials $p_1, ..., p_m \in \mathbb{F}_q[x_1, x_2, ..., x_n]$ of total degree at most 2. Try to find an element $x \in \mathbb{F}_q^n$ such that for all $i \in \{0, ..., m\}$:

$$p_i(x) = 0.$$

We can base a signature scheme (see e.g. [14]) on this hardness assumption:

1. Take some sufficiently large integers n, m with $n > m$.

2. Pick quadratic polynomials $f_1, ..., f_m \in \mathbb{F}_q[x]$ which have well-behaving properties to solve the system. This point will be discussed afterwards.

3. Choose some random matrices $U \in GL_m(\mathbb{F}_q), S \in GL_n(\mathbb{F}_q)$.

4. Compute
$$(p_1(x), ..., p_m(x)) := (f_1(x \cdot S), ..., f_m(x \cdot S)) \cdot U$$

5. Publish $p(x) := (p_1(x), ..., p_m(x))$ as public key, keep $f(x) := (f_1(x), ..., f_m(x)), U, S$ as private key.

6. If you want to sign a message $M \in \mathbb{F}_q^m$, compute a solution $s' \in \mathbb{F}_q^n$ such that
$$f(s') = M \cdot U^{-1}$$
and publish the signature $s = s' \cdot S^{-1}$.

7. Someone can verify the signature by testing if $p(s) = M$. This must hold since
$$p(s) = p(s' \cdot S^{-1}) = f(s' \cdot S \cdot S^{-1}) \cdot U = f(s') \cdot U = M \cdot U^{-1} \cdot U = M.$$

Remark 1.1.1. *An attacker would have to solve Problem 1.1.1 in order to produce a fake signature. This is assumed to be hard as long as the linear transformations U, S hide the given structure of $(f_1, ..., f_m)$.*

There are many suggestions how to choose $(f_1, ..., f_m)$. One of those is the so called Oil-and-Vinegar-scheme:
For that reason, split the variables into

$$x_1, ..., x_{n-m} = \{x_i\}_{i \in V} \text{ (vinegar variables)}$$

and

$$x_{n-m+1}, ..., x_n = \{x_i\}_{i \in O} \text{ (oil variables)}.$$

The concluding idea is as follows: Construct the f_k in such a way that there are no quadratic monomials in $x_i x_j$ for $i, j \in O$. This means that

$$f_k = \sum_{i,j \in V} a_{i,j,k} x_i x_k + \sum_{i \in V, j \in O} b_{i,j,k} x_i x_j + \sum_{i=0}^{n} c_{i_k} x_i + d_k$$

with $a_{i,j,k}, b_{i,j,k}, c_{i_k}, d_k \in \mathbb{F}_q$ chosen arbitrarily. Solving

$$f(s') = M \cdot U^{-1}$$

works now in the following way:

1. Choose randomly fixed values $a_1, ..., a_{n-m} \in \mathbb{F}_q$ for the vinegar variables.

2. Due to the structure of f_k, the resulting polynomial $f_k(a_1, ..., a_{n-m}, x_{n-m+1}, ..., x_n)$ is linear, hence the problem is reduced to solving a m-dimensional linear system. Since this linear system is invertible with high probability (see the result below), we can find a solution for the polynomial system rather fast.

3. If the linear system turns out to be not invertible, try again with new random values for the oil variables.

Remark 1.1.2. *One can show that the probability of a random matrix $U \in \mathbb{F}_q^{n \times n}$ being invertible is*

$$\prod_{i=1}^{n} (1 - \frac{1}{q^i}).$$

This equals for large n and $q = 2$ about 30%. For small n or larger q, this probability is even higher, so one should not need many iterations to obtain an invertible system. [19]

There are many different cryptographic schemes based on the hardness of the MQ Problem. For each scheme there are several approaches to attack such schemes. As we will concentrate on Gröbner bases which are used to solve the MQ Problem directly, our approach works for all schemes based on this problem.

1.2 Introduction of Gröbner bases

This chapter repeats widely known results about Gröbner bases which mostly go back to [4]. All theorems in this chapter are widely known and their proofs can be found in nearly every classical literature about Gröbner bases, e.g. [6]. Note that we stayed close to this literature during this Chapter. To define a Gröbner basis, we first need some basic algebraic definitions and results:

Definition 1.2.1 (Term order). *We call $<$ a term order on T if it satisfies the following properties:*

(i) $\forall a \in \mathbb{N}^n \backslash \{0\} : 1 < x^a$.

(ii) $\forall c \in \mathbb{N}^n \backslash \{0\} : x^a < x^b \Rightarrow x^{a+c} < x^{b+c}$.

(iii) $<$ *is a strict total order.*

Remark 1.2.2. *There is a canonical order-preserving isomorphism between $(T(n), \cdot)$ and $(\mathbb{N}^n, +)$ via*

$$f : T(n) \to \mathbb{N}^n \tag{1.1}$$

$$x^a = x_1^{a_1} \cdot x_2^{a_2} \cdot \ldots \cdot x_n^{a_n} \mapsto (a_1, ..., a_n). \tag{1.2}$$

Hence, we can also define the term order on $\mathbb{N}^n$ and take over this order to $T(n)$ via f^{-1}.

Definition 1.2.3 (special term orders).

Let $a = (a_1, ..., a_n), b = (b_1, ..., b_n) \in \mathbb{N}^n$. Set

$$|a| := \sum_{i=1}^{n} a_i.$$

The following term orders are often used in Gröbner basis context:

(i) The ***lexicographic order*** $<_{lex}$ *as*

$$a <_{lex} b :\Leftrightarrow \exists\, k \in \{1, ..., n\} \text{ such that } a_k < b_k \text{ and } \forall i < k : a_i = b_i.$$

(ii) The ***degree lexicographic order*** $<_{deg}$ *as*

$$a <_{deg} b :\Leftrightarrow |a| < |b| \text{ or } |a| = |b| \text{ and } a <_{lex} b.$$

(iii) *The* ***degree lexicographic order*** $<_{grevlex}$ *as*

$$a <_{grevlex} b :\Leftrightarrow |a| < |b| \text{ or } |a| = |b| \text{ and } a >_{lex} b.$$

Definition 1.2.4. *Let*

$$f = \sum_{t \in T} c_t t \in P$$

with $c_t \neq 0$ *finitely often,* $<$ *a fixed term order. Then we define ...*

(i) *... the* ***terms*** *of* f *as*

$$T(f) := \{x^a : c_a \neq 0\}.$$

(ii) *... the* ***coefficient*** *of* f *corresponding to the term* t *as*

$$C_t(f).$$

If $f \neq 0$*, we define...*

(iii) *... the* ***leading term*** *of* f *as*

$$LT(f) := \max\{t \in T(f)\}$$

where the maximum is taken with respect to the chosen term order.

(iv) *... the* ***leading coefficient*** *of* f *as*

$$LC(f) := C_{LT(f)}(f).$$

(v) *... the* ***leading monomial*** *of* f *as*

$$LM(f) := LC(f) \cdot LT(f).$$

Remark 1.2.5. *Some authors use the terms "leading term" and "leading monomial" the other way round. Nevertheless, we will stick to that notion defined in Definition 1.2.4 since it seems to be the more common one.*

We have everything prepared to define Gröbner bases now:

Definition 1.2.6 (Gröbner basis). *Let* $I \subseteq P$ *an ideal,* $<$ *a term order on* T*.*

(i) *We call a set* $G = \{g_1, ..., g_s\} \subseteq I$ ***finite basis*** *for* I *if*

$$I = \langle g_1, ..., g_s \rangle.$$

*(ii) We call a set $G = \{g_1, ..., g_s\} \subseteq I$ **Gröbner basis** for I with respect to $<$ if*

$$\langle LT(I)\rangle = \langle LT(g_1), ..., LT(g_s)\rangle$$

where

$$LT(I) := \{x^a : \exists f \in I : LT(f) = x^a\}.$$

Remark 1.2.7. *One can easily show that each Gröbner basis G is a basis for I and thus a finite basis, but conversely not every finite basis is a Gröbner basis.*

Theorem 1.2.8. *Each ideal $I \neq \{0\}$ has a Gröbner basis.*

Proof. See [6, Chapter 2, §5, Corollary 6, p.75]. □

1.3 Solving polynomial equations via Gröbner bases

At that point, one might ask why we considered Gröbner bases to solve multivariate polynomial systems and therefore the MQ Problem. We will see this in the following section.

Definition 1.3.1 (Variety). *Let $F \subseteq P$, then we define the **variety** of F denoted as $V(F)$ by*

$$V(F) = \{a \in \overline{\mathbb{K}}^n : \forall f \in F : f(a) = 0\}$$

where $\overline{\mathbb{K}}$ denotes the algebraic closure of $\mathbb{K}$. We define

$$V_{\mathbb{K}}(F) = \{a \in \mathbb{K}^n : \forall f \in F : f(a) = 0\}.$$

Proposition 1.3.2.

(i) $V(F) = V(\langle F\rangle)$.

(ii) If $\mathbb{K} = \mathbb{F}_q$, then $V(F \cup \{x_i^q - x_i, i \in \{1, ..., n\}\}) = V_{\mathbb{K}}(F)$.

Using the definition about varieties, the MQ Problem can be reformulated to finding an element $a = (a_1, ..., a_n) \in V_{\mathbb{F}_q}(f_1, ..., f_m)$. For finite fields, using the **field equations**

$$\mathbb{K}_i := x_i^q - x_i, i \in \{1, ..., n\}, \tag{1.3}$$

this is further equivalent to finding some $a \in V(F)$ where $F = \{f_1, ..., f_m, \mathbb{K}_1, ..., \mathbb{K}_n\}$. The following statement is crucial to building the connection between this task and Gröbner bases:

Definition 1.3.3 (Elimination ideal). *Given an ideal $I \subseteq P, l \in \{0, ..., n-1\}$, we define the l-th elimination ideal I_l as*

$$I_l := I \cap \mathbb{K}[x_{l+1}, ..., x_n].$$

Remark 1.3.4. *One can show straightforward that I_l is an ideal of $\mathbb{K}[x_{l+1}, ..., x_n]$.*

Theorem 1.3.5 (Elimination theorem). *Let $I \subseteq P$ be an ideal with Gröbner basis G with respect to $<_{lex}$ and let $l \in \{0, ..., n-1\}$. If $I_l \neq \emptyset$, then*

$$G_l := G \cap \mathbb{K}[x_{l+1}, ..., x_n]$$

is a Gröbner basis of I_l.

Proof. See [6, Chapter 3, §1, Theorem 2, p.113]. □

For that purpose, we compute a Gröbner basis G for I (we will see in the next chapter how) with respect to $<_{lex}$. Starting with $l := \max\{i \in \{1, ..., n-1\} : I_i \neq \emptyset\}$, we can find $G_l = \{g_1, ..., g_j\}$ easily by Theorem 1.3.5. Since we get for $k > l$ that $G_k = \emptyset$, we can determine l easily. Observe that G_l contains only polynomials in $\mathbb{K}[x_l]$. Solving these univariate polynomials is easy and fast in comparison to computing the whole solution. Of course, there are several approaches to solving univariate polynomials, but we will not concentrate on that in this thesis.
After solving this univariate system induced by G_l, we have a finite number of possible solutions for a_l, leaving $(a_{l+1}, ...a_n)$ arbitrary. To extend those partial solutions coordinatewise, consider the following theorem:

Theorem 1.3.6 (Extension theorem).
Let $I = \langle f_1, ..., f_s \rangle \subseteq P, (a_2, ..., a_n) \in V(I_1)$. Write

$$f_i(x_1, ..., x_n) = g_i(x_2, ..., x_n) x_1^{deg_{x_1}(f_i)} + r_i(x_1, ...x_n)$$

for uniquely determined $0 \neq g_i \in \mathbb{K}[x_2, ..., x_n], r_i \in P, deg_{x_1}(r_i) < deg_{x_1}(f_i)$.
If $(a_2, ..., a_n) \notin V(g_1, ..., g_s)$, then there exists an $a_1 \in \mathbb{K}$ such that $(a_1, ..., a_n) \in V(I)$.

Proof. See [6, Chapter 3, §1, Theorem 3, p.115]. □

Theorem 1.3.6 helps in the following sense: Assume we have such a partial solution $(a_{l+1}, ..., a_n) \in V(I_l)$ and want to extend it to $I_{l-1} = \langle G_{l-1} \rangle$ with $G_{l-1} = \{g_1, ..., g_s\} \subseteq \mathbb{K}[x_l, x_{l+1}, ..., x_n]$. We will check via Theorem 1.3.6 whether $(a_{l+1}, ..., a_n)$ can be extended. If this is the case, compute

$$g_1(x_l, a_{l+1}, a_{l+2}, ..., a_n), ..., g_s(x_l, a_{l+1}, a_{l+2}, ..., a_n) \in \mathbb{K}[x_l]. \tag{1.4}$$

This gives us a univariate polynomial system again. Doing this step for each coordinate yields a way to solve the multivariate polynomials and therefore the MQ Problem.

2 Computation of Gröbner bases

2.1 Gröbner basis properties

As in section 1.2, most results in this section are widely known. If not stated otherwise, the theorems and their proofs can be found in [6]. To define a Gröbner basis, we first need some basic algebraic definitions and results:

Definition 2.1.1 (Reduction of polynomials). *Let G be a finite set of nonzero polynomials, $f, f' \in P$, $<$ a term order.*

*(i) We say f is **reducible** by G if there exist*

$$t \in T(f), g \in G : LT(g)|t.$$

*Otherwise, we call f **irreducible** by G.*

*(ii) We say f is **top-reducible** by G if there exists*

$$g \in G : LT(g)|LT(f).$$

*Otherwise, we call f **top-irreducible** by G.*

*(iii) We define a **(top-)reduction step** for f by*

$$f \xrightarrow[G]{} f'$$

if $f' = f - cug$, where $c \in \mathbb{K}\backslash\{0\}, u \in T, g \in G$ such that

$$LM(cug) = C_t(f)t$$

for some $t \in T(f)$.

*(iv) We say f **can be reduced** to f' by G, denoted by $f \xrightarrow[G]{*} f'$, if there exist $k \in \mathbb{N}, f_1, ..., f_k \in P$ such that*

$$f = f_1 \xrightarrow[G]{} f_2 \xrightarrow[G]{} f_3 \xrightarrow[G]{} ... \xrightarrow[G]{} f_{k-1} \xrightarrow[G]{} f_k = f'.$$

(v) *We say f' is a* ***normal form of*** $\mathbf{f}$ ***with respect to*** *G if $f \xrightarrow[G]{*} f'$ and f' is irreducible with respect to G. We denote the set of all normal forms of f with respect to G by $\overline{f}^G$.*

Remark 2.1.2.

1. *Note that $f \xrightarrow[G]{*} g$ contains the case of zero reduction steps. Hence, $f \xrightarrow[G]{*} f$ holds trivially.*

2. *The zero polynomial is by definition irreducible since $T(0) = \emptyset$.*

3. *f is irreducible is equivalent to $f \xrightarrow[G]{*} g \Rightarrow f = g$. This follows by the fact that if there exist $t \in T(f), g \in G : LT(g)|t$, we can always find suitable $c \in \mathbb{K}\backslash\{0\}, u \in T$ such that $LM(cug) = C_t(f)t$.*

4. *$\overline{f}^G \neq \emptyset$ for all $f \in P$ since we will see that f can only be reduced finitely often until it is irreducible.*

One characterizing property of a Gröbner basis is the uniqueness of the normal form:

Theorem 2.1.3. *Let f be a polynomial, G a Gröbner basis of an ideal I. Then f has a unique normal form, i.e. $|\overline{f}^G| = 1$ or equivalently, $f \xrightarrow[G]{*} g$, $f \xrightarrow[G]{*} h$ with g, h irreducible, implies $g = h$. Conversely, if all $f \in P$ have a unique normal form, then G is a Gröbner basis.*

Proof. See [25, Theorem 5.35, p.206]. □

Corollary 2.1.4. *Let f be a polynomial, G a Gröbner basis of an ideal I.*
The following are statements are equivalent:

(i) $f \in I$.

(ii) $\overline{f}^G = \{0\}$.

(iii) $f \xrightarrow[G]{*} 0$.

The following lemma and its proof were found by the author itself. It is an equivalent Gröbner basis characterization besides many others:

Lemma 2.1.5. *Let $G \subseteq P$ and let $\sim_G$ be relation on P defined as follows:*

$$f \sim_G g :\Leftrightarrow \overline{f}^G \cap \overline{g}^G \neq \emptyset.$$

Then $\sim_G$ is an equivalence relation if and only if G is a Gröbner basis.

Proof. Obviously, $\sim_G$ is always reflexive and symmetric, so it suffices to prove the transitivity. Assume $f \sim_G g, g \sim_G h$. If G is a Gröbner basis, we have by the uniqueness of the normal form that $\overline{f}^G = \overline{g}^G, \overline{g}^G = \overline{h}^G$ and hence, $f \sim_G h$. Conversely, let $\sim_G$ be an equivalence relation on P. Furthermore, let $f \in P$ and let g, h be two arbitrary normal forms of f. Then $f \sim_G g, f \sim_G h$ implies by transitivity that $g \sim_G h$. Since g, h are irreducible, we have $\overline{g}^G = \{g\}, \overline{h}^G = \{h\}$. Thus $g = h$ follows which implies by Theorem 2.1.3 that G is a Gröbner basis. □

Note that we can compute a normal form of f with respect to G quite easily by the following algorithm:

Algorithm 1: Division algorithm

Input: finite basis G for an ideal I, polynomial $f \in \mathbb{K}[x_1, ..., x_n]$.

Output: A polynomial $f' \in \overline{f}^G$.

```
1 Set f' := f;
2 while ∃g ∈ G, t ∈ T(f') : LT(g)|t do
3     f' = f' − C_t(f') t/LM(g) g;
4 end
5 return f';
```

Remark 2.1.6. *The correctness of Algorithm 1 is straightforward. For the termination, consider the following: We extend the term order on polynomials by*

$$f < g :\Leftrightarrow \max\{T(f) \triangle T(g)\} \in T(g)$$

where $\triangle$ denotes the symmetric difference and the maximum is taken with respect to the chosen term order. One can see that $f \xrightarrow[G]{*} g$ implies $f > g$. In that way, it follows that we can only take a finite number of reduction steps for an arbitrary but fixed polynomial f since no infinite strictly descending chain of polynomials exists.

To use all those considerations for a Gröbner basis algorithm, we will study a criterion to check whether some set $G \subseteq P$ is indeed a Gröbner basis. For that, we need another definition:

Definition 2.1.7 (S-polynomials). *Let $f, g \in P \setminus \{0\}$. We define the* ***S-polynomial*** *of f and g as*

$$Spol(f, g) := \frac{\operatorname{lcm}(LT(f), LT(g))}{LM(f)} f - \frac{\operatorname{lcm}(LT(f), LT(g))}{LM(g)} g.$$

Remark 2.1.8. *Note that those S-polynomials are constructed in a way that a cancellation of the leading monomials takes place. This means for $f, g \in I$,*

$$LT(Spol(f, g)) \in \langle LT(I) \rangle,$$

but in general

$$LT(Spol(f, g)) \notin \langle LT(f), LT(g) \rangle.$$

This gives a hint that these S-polynomials might be crucial in a Gröbner basis test:

Theorem 2.1.9. *Let $G = \{g_1, ..., g_k\}$ be a basis for $I = \langle G \rangle$. Then*

$$G \text{ is a Gröbner basis } \Leftrightarrow \forall i, j \in \{1, ..., k\}, i \neq j : Spol(g_i, g_j) \xrightarrow[G]{*} 0.$$

Proof. We will see this criterion in a more general way later, for a direct proof see [6, Chapter 2, §6, Theorem 6, p.82]. □

2.2 Buchberger Algorithm

Using the criterion from Theorem 2.1.9, the idea of constructing a first Gröbner basis algorithmically is straightforward:

Algorithm 2: Buchberger Algorithm

Input: Set of polynomials $F = \{f_1, ..., f_k\}$.

Output: Gröbner basis G of $\langle F \rangle$ with $F \subseteq G$.

Set $G = F$;
Set $B = \{(g_i, g_j) : g_i, g_j \in G\}$;
while $B \neq \emptyset$ **do**
    Select some $(g_i, g_j) \in B$ and remove it from B;
    Compute a normal form $f' \in \overline{Spol(g_i, g_j)}^G$ (e.g. with Algorithm 1);
    if $f' \neq 0$ **then**
        Set $G = G \cup \{f'\}$;
        Set $B = B \cup \{(g_i, f')\}$;
    end
    return G;
end

The correctness proof is as follows:

1. Since $f' \in \overline{Spol(g_i, g_j)}^G \in \langle G \rangle$ for $g_i, g_j \in G$, we have that $F \subseteq \langle G \rangle = I$ and therefore $\langle G \rangle = I$.

2. By Theorem 2.1.9, G is a Gröbner basis for $\langle G \rangle = I$.

3. To prove termination, one can show that

$$\langle LT(G) \rangle \subsetneq \langle LT(G \cup \{f'\}) \rangle$$

each time we add some $f' \in \overline{Spol(g_i, g_j)}^G$ in the algorithm. Assuming that the algorithm does not terminate, we get an infinitely long strictly ascending chain of ideals. But this is not possible since the polynomial ring is known to be Noetherian.

Remark 2.2.1. *There are two main drawbacks of using this standard Buchberger Algorithm in the described form:*

1. *The obtained Gröbner basis might be larger than necessary. In general, it produces different outputs for different choices depending on the order in which the pairs* (g_i, g_j) *are examined.*

2. *We get no information about the running time. In general, the Buchberger Algorithm tends to be computationally expensive.*

The first drawback can be treated very well by the introduction of minimal Gröbner bases:

Definition 2.2.2 (Minimal Gröbner basis). *Let* G *be a Gröbner basis of* I*. Then* G *is said to be* ***minimal*** *if all* $g \in G$ *fulfill the following two conditions:*

(i) $LC(g) = 1$.

(ii) $LT(g) \notin \langle LT(G \backslash \{g\}) \rangle$.

To construct such a minimal Gröbner basis from a basis obtained by the Buchberger Algorithm (or other Gröbner basis algorithms), one simply has to normalize all polynomials to fulfill condition (i). This does not change the generated ideal at all. To fulfill (ii), consider the following result: Assume $LT(g) \in \langle LT(G \backslash \{g\}) \rangle$ for some $g \in G$. Then

$$\langle LT(G \backslash \{g\}) \rangle = \langle LT(G) \rangle = \langle LT(I) \rangle$$

since G is a Gröbner basis. Therefore, we just have to iteratively go through G, searching for $g \in G$ fulfilling $LT(g) \in \langle LT(G \setminus \{g\}) \rangle$. Simply removing this g (justified by a short consideration, see e.g. [6, Chapter 2, §7, Lemma 3, p.89]) and repeating this step leads to a minimal Gröbner basis. It turns out that all minimal Gröbner bases have the same cardinality and are of minimal cardinality among all Gröbner bases. Nevertheless, in general there still exist many different minimal Gröbner bases. One can see this since the elements in a minimal Gröbner basis do not need to be fully reduced with respect to G. Hence, we define a reduced Gröbner basis in the following way:

Definition 2.2.3 (Reduced Gröbner basis). *Let* G *be a Gröbner basis of* I*. Then* G *is said to be* ***reduced*** *if all* $g \in G$ *fulfill the following two conditions:*

(i) $LC(g) = 1$.

(ii) $T(g) \cap \langle LT(G \backslash \{g\}) \rangle = \emptyset$.

Note that a reduced Gröbner basis can be shown to be unique. To obtain the reduced Gröbner basis from a minimal one, we compute some

$$g' \in \overline{g}^{G \backslash \{g\}}$$

for $g \in G$ and set

$$G = (G \backslash \{g\}) \cup \{g'\}.$$

Repeating this for all $g \in G$, we get a reduced Gröbner basis. These steps can all be computed relatively quickly in comparison to the original Gröbner basis algorithm, hence, the first point is no problem any longer. The second one concerning the efficiency of Gröbner basis algorithms is a huge research area and will be treated in the main part of this thesis.

2.3 Improvements in the Buchberger Algorithm

Considering the efficiency and possible improvements for the Buchberger Algorithm is a whole research area. There are many suggestions and ideas which decrease the running time of the Buchberger Algorithm. We can mainly optimize the following things in the standard Buchberger Algorithm:

1. Choose the best order in which new S-polynomials are examined (Selection strategy).
2. Check, with several criteria, whether $Spol(g_i, g_j) \xrightarrow[G]{*} 0$ without computing it explicitly.
3. Represent the algorithm in matrix formulation to speed up reduction steps (Matrix formulation).

2.3.1 Selection strategy

The **selection strategy** describes a rule after which the next pair (g_i, g_j) to compute $f' \in \overline{Spol(g_i, g_j)}^G$ is chosen. We will present now one of the most used strategies, namely the **normal strategy**. To understand the idea, we will start with a digression into homogeneous ideals. We will stay close to the definitions and results in [25].

Definition 2.3.1.

*(i) Let $f \in P$. We say f is **homogeneous of degree** $\mathbf{d}$ if*

$$f = \sum_{i=1}^{k} c_i t_i, \quad c_i \in \mathbb{K} \backslash \{0\}, t_i \in T, deg(t_i) = d \in \mathbb{N}.$$

(ii) f is homogeneous if there exists a $d \in \mathbb{N}$ such that f is homogeneous of degree d.

(iii) Let I be an ideal. Then I is called homogeneous if there exists a generating set F such that all $f \in F$ are homogeneous.

Remark 2.3.2. *It can be shown that for each homogeneous ideal I, there exists a **finite** homogeneous basis.*

The current strategy only works for homogeneous input sets F, but in general, we do not have such an input. The following definitions and results show us that we can assume each input set to be homogenous:

Definition 2.3.3 (Homogenization). *Let $f \in P$ with $deg(f) = d$. Define the **homogenization of f** as the polynomial*

$$f^* := X_0^d f(\frac{x_1}{x_0}, ..., \frac{x_n}{x_0}) \in \mathbb{K}[x_0, x_1, ..., x_n].$$

For $F = \{f_1, ..., f_m\}$, we define $F^* := \{f_1^*, ..., f_m^*\}$.

Remark 2.3.4. *The definition of the homogenized polynomial above can be described as "filling up" all terms of f by the appropriate power of the new variable x_0 to obtain a polynomial which is homogeneous of degree d.*

Definition 2.3.5 (Dehomogenization). *Let $f \in \mathbb{K}[x_0, x_1, ..., x_n]$ homogeneous of degree d. We define the **dehomogenization of f** (with respect to x_0) by*

$$f_* := f(1, x_1, ..., x_n).$$

For $F = \{f_1, ..., f_m\}$, we define $F_* := \{f_{1*}, ..., f_{m*}\}$.

Remark 2.3.6. *Note that these two operations are somehow inverse to each other: For $f \in \mathbb{K}[x_1, ..., x_n]$, we have*

$$(f^*)_* = f$$

and for $f \in \mathbb{K}[x_0, x_1, ..., x_n]$, we have

$$(f_*)^* = x_0^d f$$

for some $d \in \mathbb{N}$.

Let us extend our term order $\leq$ of $\mathbb{K}[x_1, ..., x_n]$ to $\leq'$ operating on $\mathbb{K}[x_0, x_1, ..., x_n]$ in the following way: Let $t_1 = s_1 x_0^k, t_2 = s_2 x_0^l \in T(x_0, ..., x_n)$ with $s_1, s_2 \in T(x_1, ..., x_n)$. Then

$$t_1 \leq' t_2 :\Leftrightarrow \begin{cases} s_1 < s_2 \text{ or} \\ s_1 = s_2 \text{ and } l \leq k. \end{cases}$$

Extending the term order in that way, we can state the following theorem:

Theorem 2.3.7. *Let G be a homogeneous Gröbner basis of $\langle F^* \rangle$ with respect to $\leq'$. Then G_* is a Gröbner basis of $\langle F \rangle$ with respect to $\leq$.*

Proof. See [25, Chapter 10.3, Lemma 10.57, p.483]. □

Definition 2.3.8 (d-Gröbner basis). *Let $G = \{g_1, ..., g_s\} \subseteq \mathbb{K}[x_0, ..., x_n]$ be a homogeneous for some ideal I. We call G a d-Gröbner basis for some $d \in \mathbb{N}_\infty$ if for all $f \in I$ with $deg(f) \leq d$:*

$$f \xrightarrow[G]{*} 0.$$

Definition 2.3.9. *Let $d_1, d_2 \in \mathbb{N}_\infty$ such that $d_1 \leq d_2$. Define $[d_1, d_2]$-Buchberger(F) as the output obtained by the (classical) Buchberger Algorithm when considering only critical pairs $(g_1, g_2) \in G$ with the property that*

$$d_1 \leq deg(\text{lcm}(LT(g_1), LT(g_2))) \leq d_2.$$

Lemma 2.3.10. *Let F be a set of homogeneous polynomials, $d_1 \leq d_2 < d_3 \in \mathbb{N}_\infty$.*

(i) $[d_2 + 1, d_3]$*-Buchberger*$([d_1, d_2]$*-Buchberger*$(F)) = [d_1, d_3]$*-Buchberger*(F).

(ii) $[0, d_1]$*-Buchberger*(F) *is a* d_1*-Gröbner basis.*

Proof. See [25, Chapter 10.2, Lemma 10.35, p.470]. □

This enables us to adapt the classical Buchberger Algorithm in the following way: Starting with $d = 1$, iteratively run a $[d, d]$-Buchberger on the previous output, increasing d by one each time. Note that after d_1 steps, we obtain a d_1-Gröbner basis by Lemma 2.3.10. The following theorem shows us why this strategy is advantageous:

Theorem 2.3.11. *Let $d \in \mathbb{N}$, G a d-Gröbner basis of $\langle F^* \rangle$. Let $p \in \mathbb{K}[x_1, ..., x_n]$ with $deg(p) = d' \leq d$. Then the following statements are equivalent:*

(i) There exist polynomials $p_f \in \mathbb{K}[x_1, ..., x_n]$ such that

$$p = \sum_{f \in F} p_f f, \quad \max\{deg(p_f f) : f \in F\} \leq d.$$

(ii) $x_0^{d-d'} p^* \xrightarrow[G]{*} 0.$

Proof. See [25, Chapter 10.3, Theorem 10.55, p.480]. □

We can exploit Theorem 2.3.11 in the following way: When computing $[d + 1, d + 1]$-Buchberger, assume we have to examine $Spol(g_i, g_j)$ with

$$deg(lcm(LT(g_i), LT(g_j))) = d + 1, \quad deg(lcm(LT(g_{i*}), LT(g_{j*}))) = d' \leq d.$$

Setting

$$p := Spol(g_{i*}, g_{j*}) = Spol(g_i, g_j)_*,$$

we have fulfilled (i) of Theorem 2.3.11. Since $Spol(g_i, g_j) = x_0 \cdot x_0^{d-d'} p^*$, we know that

$$Spol(g_i, g_j) \xrightarrow[G]{*} 0.$$

Hence, we do not explicitly need to compute a normal form of this S-polynomial and can save some computational power. This strategy is called **normal strategy**. It is known that the normal strategy works specifically well on term-orders which are ordered by degree. Thus, we can compute a Gröbner basis with respect to $<_{lex}$ for nonhomogeneous F in the following way:

1. Homogenize F.
2. Compute a Gröbner basis G of F^* with respect to some degree-order, e.g. $<_{grevlex}$.
3. Transform the Gröbner basis via the FGLM Algorithm (see section 2.4) into a Gröbner basis with respect to $<_{lex}$ where x_0 is rated the least significant.
4. Dehomogenize the obtained Gröbner basis to obtain a Gröbner basis with respect to $<_{lex}$ for F.

Of course, we can use the normal selection strategy (choosing an S-polynomial of lowest degree) as well for nonhomogeneous ideals. The problem is that we lose the nice properties of discarding S-polynomials as above. To overcome this, we can implicitly compute the homogeneous version by carrying a phantom degree for x_0, called sugar variable, for each polynomial. We do not explicitly homogenize our polynomials, but treat the polynomials in that way. This leads to the **sugar strategy**. We define the sugar S_f for a polynomial f in the following way:

(i) For the starting polynomials $f_i \in F$, we set $S_{f_i} := \deg(f_i)$.

(ii) For $t \in T$ we define $S_{t \cdot f} := S_f + \deg(t)$.

(iii) For $g \in P$, we define $S_{f+g} := \max\{S_f, S_g\}$.

Our selection always chooses a critical pair with the lowest sugar degree of the corresponding S-polynomial. In most cases, this is known to behave much better than the normal strategy, especially when using pure lexicographic orders (see [16]).

2.3.2 Reduction criteria

There are several criteria on how to see in advance whether an S-polynomial can be reduced to 0 without even computing it. We will come back to that in the discussion of signature Gröbner bases later where we will define rather complex, but very powerful criteria. For now, we only propose some criteria which can be implemented rather easy in the classical Buchberger Algorithm:

Lemma 2.3.12 (First Buchberger criterion). *Let $G \subseteq P$, $f, g \in G$ and let*

$$\mathrm{lcm}(LT(f), LT(g)) = LT(f)LT(g). \tag{2.1}$$

Then $Spol(f, g) \xrightarrow[G]{*} 0$.

Proof. See [6, Chapter 2, §9, Proposition 4, p.101]. □

Remark 2.3.13. *The condition of (2.1) is equivalent to $LT(f), LT(g)$ containing only disjoint variables x_i.*

Note that it is rather easy to implement this criterion into a classical Buchberger Algorithm. For a second criterion, we need a deeper understanding of the role of the S-polynomials:

Definition 2.3.14 (Syzygy). *Let $F = (f_1, ..., f_m)$ be an m-tuple of polynomials, $\boldsymbol{h} = (h_1, ..., h_m) \in P^m$.*

*(i) We define the **evaluation homomorphism** with respect to F by*

$$\begin{aligned} v_F : P^m &\to P \\ \boldsymbol{h} &\mapsto \sum_{i=1}^{m} h_i f_i. \end{aligned}$$

*(ii) $\boldsymbol{h}$ is called a **syzygy** (on F) if $v_F(h) = 0$.*

*(iii) We call a syzygy $\boldsymbol{h}$ **homogeneous** if for all $i, j \in \{1, ..., m\}$ with $h_i, h_j \neq 0$,*

$$deg(f_i h_i) = deg(f_j h_j)$$

holds.

(iv) We denote the set of all syzygies on F by $Syz(F)$.

Remark 2.3.15.

1. *Since $Syz(F) = ker(v_F)$, it is straightforward to show that $Syz(F)$ is closed under addition and component-wise multiplication of polynomials, hence, $Syz(F)$ is a P-submodule of P^m. We will go more into detail about the module P^m and syzygies in Section 3.*

2. *The word "S-polynomials" in [4] was invented to abbreviate "Syzygy polynomials". This is explained by the following observation: The S-polynomials $Spol(f_i, f_j)$ induce homogeneous syzygies on $LT(F)$ by $\boldsymbol{s_{i,j}} = (h_1, ..., h_m)$ via*

$$h_k := \begin{cases} 0 & \text{if } k \notin \{i, j\} \\ \frac{\mathrm{lcm}(LT(f_i), LT(f_j))}{LM(f_i)} & \text{if } k = i \\ -\frac{\mathrm{lcm}(LT(f_i), LT(f_j))}{LM(f_j)} & \text{if } k = j \end{cases}$$

Note that $v_F(\boldsymbol{s_{i,j}}) = Spol(f_i, f_j)$.

Lemma 2.3.16. *$Syz(LT(F))$ is finitely generated (as a P-module) by*

$$\{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\}.$$

Proof. See [6, Chapter 2, §9, Proposition 8, p.104]. $\square$

Theorem 2.3.17. *Let $G = \{g_1, ..., g_s\}$ be a basis for an ideal I and $\{\boldsymbol{s_1}, ..., \boldsymbol{s}_k\}$ a homogeneous basis for $Syz(LT(G))$. Then*

$$G \text{ is a Gröbner basis of } I \Leftrightarrow \forall i \in \{1, ..., k\} : v_G(\boldsymbol{s}_i) \xrightarrow[G]{*} 0.$$

Proof. See [6, Chapter 2, §9, Theorem 9, p.104]. $\square$

Remark 2.3.18. *Since $\{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\}$ is by Lemma 2.3.16 a homogeneous basis for $Syz(LT(G))$, Theorem 2.1.9 follows directly from Theorem 2.3.17.*

Note that $\{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\}$ is not necessarily linearly independent. Therefore, it often suffices to take a proper subset as a basis of $Syz(LT(G))$: If $\boldsymbol{S} \subseteq P^m$ generates $Syz(LT(G))$ and $\boldsymbol{S} \setminus \{\boldsymbol{s}_{i,j}\}$ generates $Syz(LT(G))$ as well, we do not need to explicitly compute the reduction of $Spol(f_i, f_j)$ by Theorem 2.3.17. An easy application of this principle is stated in the following criterion:

Lemma 2.3.19 (Buchberger's second criterion). *Let $\boldsymbol{S} \subseteq \{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\}$ generate $Syz(LT(G))$. Suppose there exist $i, j, k \in \{1, ..., m\}$ such that:*

(i) $LT(g_k) | \operatorname{lcm}(LT(g_i), LT(g_j))$.

(ii) $\boldsymbol{s}_{i,k}, \boldsymbol{s}_{j,k} \in S$.

Then $\boldsymbol{S} \setminus \{\boldsymbol{s}_{i,j}\}$ generates $Syz(LT(G))$ (and hence, $f' \in \overline{Spol(g_i, g_j)}^G$ does not need to be computed).

Proof. See [6, Chapter 2, §9, Proposition 10, p.106]. $\square$

To implement this in the Buchberger Algorithm, one simply checks when a critical pair (g_i, g_j) is examined whether there is a $k \notin \{i, j\}$ with (g_i, g_k) and (g_j, g_k) not in current B (notations from Algorithm 2) with

$$LT(g_k) | \operatorname{lcm}(LT(g_i), LT(g_j)).$$

If this is the case, one can remove (g_i, g_j) from B without computing the remainder. The correctness of this procedure follows from the discussion above.

2.4 FGLM Algorithm

The following algorithm is due to Faugère, Gianni, Lazard and Mora (see [13]). It transforms a Gröbner basis from an arbitrary term order into a Gröbner basis with respect to $<_{lex}$. This turns out to be useful because computing a Gröbner basis directly in lexicographic order is often very time-consuming. We will follow the explanations of [5]. First, we need a small excursion into algebra: Let G be a Gröbner basis for ideal I, then we know that for $f \in P$, $\overline{f}^G$ contains exactly one element. Thus, we will write in this section $\overline{f}^G$ as the element contained in $\overline{f}^G$. Remember that

$$\begin{aligned} f \in I \Leftrightarrow \overline{f}^G &= 0, \\ \overline{f}^G = \overline{g}^G &\Leftrightarrow f - g \in I. \end{aligned}$$

One can show that

$$\overline{f}^G + \overline{g}^G = \overline{f+g}^G \tag{2.2}$$

$$\overline{\overline{f}^G \cdot \overline{g}^G}^G = \overline{fg}^G. \tag{2.3}$$

With that in mind, we can find a correspondence to the algebra $\mathbb{K}[x_1, ..., x_n]/I$: We know that for $[f], [g] \in \mathbb{K}[x_1, ..., x_n]/I$, we have

$$[f] = [g] \Leftrightarrow f - g \in I, \quad \overline{f}^G \in [f].$$

Thus, we can see $\overline{f}^G$ as a standard representative of $[f]$ with operations defined as in (2.2) and (2.3). These elements are $\mathbb{K}$-linear combinations of terms x^a with $x^a \notin \langle LT(I) \rangle$. Furthermore, one can show that the cosets $[x^a]$ are all linearly independent. Hence,

$$B := \{[x^a] : x^a \notin \langle LT(I) \rangle\}$$

is a basis for $\mathbb{K}[x_1, ..., x_n]/I$ as a $\mathbb{K}$-vector space. This leads to the following theorem:

Theorem 2.4.1. *The following statements are equivalent:*

(i) B is a finite set ($\Leftrightarrow \mathbb{K}[x_1, ..., x_n]/I$ is a finite-dimensional $\mathbb{K}$-vector space).

(ii) $|V(I)| < \infty$.

Proof. See [6, Chapter 5, §3, Theorem 6, p.230]. □

Definition 2.4.2. *An ideal $I \subseteq P$ is called **zero-dimensional** if $|V(I)| < \infty$.*

Having that considerations in mind, the FGLM Algorithm works as follows: As input we have a Gröbner basis G with respect to an arbitrary term order. We construct two lists G_{lex}, B_{lex}

which are empty at the beginning. The list $G_{lex} = (g_1, ..., g_k)$ will be a Gröbner basis with respect to $<_{lex}$ at the end of the algorithm with

$$LT(g_1) <_{lex} LT(g_2) <_{lex} ... <_{lex} LT(g_k).$$

The algorithm has a main while-loop where elements $x^a, a = (a_1, ..., a_n)$, starting with $x^a = x_1$, are examined as long as $x^a \neq x_n^k$ with $x_n >_{lex} x_{n-1} >_{lex} ... >_{lex} x_1, k \in \mathbb{N}$. The next iteration starts with the next term (with respect to $<_{lex}$) which is not divisible by any $LT(g_i)$, $g_i \in G_{lex}$. The algorithm computes the reduction $\overline{x^a}^G$ and checks if $\overline{x^a}^G$ is linearly dependent of $\overline{B_{lex}}^G := \{\overline{x^b}^G : x^b \in B_{lex}\}$. If it is, the algorithm computes $c_b \in \mathbb{K}$ such that

$$\overline{x^a}^G - \sum_{b:\overline{x^b}^G \in \overline{B_{lex}}^G} c_b \overline{x^b}^G = 0.$$

Note that

$$g := x^a - \sum_{b:x^b \in B_{lex}} c_b x^b \in I,$$

and, therefore, we add this g to G_{lex}. Since the x^a are chosen increasingly with respect to $<_{lex}$, we can guarantee that $LT(g) = x^a$. If $\overline{x^a}^G$ is linearly independent of $\overline{B_{lex}}^G$, we add x^a to B_{lex}.

Theorem 2.4.3. *Let I be a zero-dimensional ideal. Then the FGLM Algorithm terminates and returns a Gröbner basis with respect to $<_{lex}$.*

Proof. (compare [5, Chapter 2, Theorem 3.4, p.49]) We start with the termination: Assuming that the algorithm does not terminate, we go infinitely often through the main loop, hence G_{lex} or B_{lex} must be infinite. Assume first that B_{lex} is infinite: This implies that $\mathbb{K}[x_1, ..., x_n]/I$ is an infinite-dimensional vector space, contradicting that I is zero-dimensional by Theorem 2.4.1. Hence G_{lex} must be infinite. Assume that during the algorithm $G_{lex} = \{g_1, ..., g_k\}$ and g_{k+1} is added to G_{lex}. Since all terms x^a are chosen in a way that they are not divisible by any $LT(g_i)$ with $1 \leq i \leq k$ and $LT(g_{k+1}) = x^a$, we know that $LT(g_{k+1})$ is not divisible by any $LT(g_i)$. Conversely, $LT(g_{k+1})$ does not divide any $LT(g_i)$. This implies that we have an infinite set of terms, namely $\{LT(g_1), LT(g_2), ...\}$ where no term divides any other term. This contradicts Dickson's Lemma, a famous order-theoretic result found in most classical literature.
To prove that $G_{lex} = \{g_1, ...g_l\}$ is a Gröbner basis with respect to $<_{lex}$, assume to the contrary that $\langle LT(I)\rangle \neq \langle LT(G_{lex})\rangle$. Since $G_{lex} \in I$, this implies $\langle LT(G_{lex})\rangle \subsetneq \langle LT(I)\rangle$. This means that there exists a $g \in I$ such that $LT(g)$ is not divisible by any $LT(g_i), 1 \leq i \leq l$. We can assume without loss of generality that g is already reduced with respect to G_{lex}, otherwise we take $g' := \overline{g}^{G_{lex}}$. If $LT(g) > LT(g_l) = x_n^k$, then clearly $LT(g_l)$ divides $LT(g)$, which is a contradiction. Hence, there exists an $i_0 \in \{0, ..., l\}$ such that $LT(g) < LT(g_{i_0})$. Since g is reduced, all non-leading terms of g are not divisible by any $LT(g_i)$ with $i < i_0$. This implies that all non-leading terms of g are contained in B_{lex}. Since $LT(g)$ is not divisible by any $LT(g_i)$, it would have been examined in the algorithm. But since all non-leading terms of g

are already in B_{lex}, there exists a linear combination and g would have been added to G_{lex}, a contradiction. □

Remark 2.4.4. *For the correctness of the FGLM Algorithm, we need a zero-dimensional ideal I. This is no real restriction in our setting because most ideals used in cryptography are of this property. Note that there exist rather fast Gröbner basis conversion algorithms even for non-zero-dimensional ideals as well, but we will not discuss them further in this thesis.*

2.5 F4 Algorithm

2.5.1 Matrix representation

We will change the perspective of multivariate polynomials and the corresponding algorithms to compute Gröbner bases. In this section we will follow the description of [24].

Definition 2.5.1 (Macaulay matrix). *Let $F = \{f_1, ..., f_k\} \in P$, $<$ a term order. Let*

$$T(F) := \bigcup_{i=1}^{n} T(f_i) = \{t_1, ..., t_l\}$$

with $t_1 < t_2 < ... < t_l$.
Then we define the matrix $Mac(F) \in \mathbb{K}^{k \times l}$ via $Mac(F)_{i,j}$ being the coefficient of t_j in the polynomial f_i which is possibly zero.

Remark 2.5.2. *Considering F as an ordered set, this matrix is unique.*

Example 2.5.3: Set $F = (f_1, f_2, f_3)$ with

$$\begin{aligned} f_1 &= 2x_1x_2 + x_3^2 - x_1 + x_3 + 1 \\ f_2 &= x_1^2 + x_2x_3 + 3x_3^2 + 1 \\ f_3 &= x_2^2 + 5x_2x_3 + x_3^2 + x_3. \end{aligned}$$

Choosing term order $<_{deglex}$, we get $Mac(F) =$

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cccccccc} x_1^2 & x_1x_2 & x_2^2 & x_2x_3 & x_3^2 & x_1 & x_3 & 1 \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 0 & 1 & -1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 3 & 0 & 0 & 1 \\ 0 & 0 & 1 & 5 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

We can construct new polynomials $g \in I$ by multiplying current rows by polynomials. This increases the number of rows by 1, but also the number of columns in case we generate new terms not yet in $T(F)$. Furthermore, we can construct new rows by linear combinations of such products of monomials and current rows. We can change the entries in row i by a somehow extended Gaussian row elimination: Since $g_i - pg_j \in I$ for $g_i, g_j \in I, p \in P, LT(g_i) \geq LT(pg_j)$,

we can subtract row j p times from row i. Note that p can be a polynomial and not only a scalar as in normal Gaussian elimination! This extended Gaussian elimination is the matrix equivalent to the reduction of polynomials defined in Definition 2.1.1. Hence we have a Gröbner basis as soon as arbitrary rows corresponding to polynomials in I can be reduced to a zero row by extended Gaussian elimination. To see this, we describe the Buchberger Algorithm as matrix formulation:
The S-polynomials

$$Spol(g_i, g_j) := \frac{\operatorname{lcm}(LT(g_i), LT(g_j))}{LM(g_i)} g_i - \frac{\operatorname{lcm}(LT(g_i), LT(g_j))}{LM(g_j)} g_j$$

are polynomials created by the operations described above. Computing $f' \in \overline{Spol(g_i, g_j)}^G$ is equivalent to using extended Gaussian elimination steps on the row corresponding to $Spol(g_i, g_j)$.

Remark 2.5.4. *From an arbitrary Gröbner basis in matrix form, we obtain a reduced one by doing more extended Gaussian elimination steps. The Gröbner basis is reduced as soon as the matrix is in reduced row-echelon form.*

2.5.2 The algorithm

The main idea of the F4 Algorithm is to speed up the time needed to compute the reduction steps via efficient matrix algorithms, reducing several S-polynomials at once. The algorithm does not specify the selection strategy, so one can choose a suitable one, e.g. normal strategy or sugar strategy. In contrast to the Buchberger Algorithm, a set of critical pairs is selected at the same time. To grasp the main ideas, we need some subroutines and definitions. We will take the statements and notations used in [15] and [22]. We start the discussion by writing down the pseudocode of the F4 Algorithm:

Algorithm 3: F4 main routine

Input: $F = \{f_1, ..., f_m\} \subseteq P$, term order $<$.
Output: A Gröbner basis G for $\langle F \rangle$ with respect to $<$.

Set $G = F$;
Set $B = \{(g_i, g_j) : g_i, g_j \in G, i \neq j\}$;
while $B \neq \emptyset$
  $Sel = Select(B)$; //some set selection strategy
  $B = B \setminus Sel$;
  $L = \{\frac{\operatorname{lcm}(LT(g_i),LT(g_j))}{LM(g_i)} g_i : (g_i, g_j) \in Sel\} \cup \{\frac{\operatorname{lcm}(LT(g_i),LT(g_j))}{LM(g_j)} g_j : (g_i, g_j) \in Sel\}$;
  $\tilde{H}^+ = ReductionF4(L, G)$;
  for $f \in \tilde{H}^+$
    $B = B \cup \{(f, g) : g \in G\}$;
    $G = G \cup \{f\}$;

end
end
return G;

This routine looks very similar to Buchberger's Algorithm. The main difference is the function $ReductionF4$:

Algorithm 4: ReductionF4

Input: sets of polynomials L, G
Output: set of polynomials $\tilde{H}^+$, corresponding to reduced S-polynomials in Buchberger's Algorithm

$H = SymbolicPreprocessing(L, G)$;
$\tilde{H}$ = Set of polynomials obtained by the reduced row echelon form of $Mac(H)$;
$\tilde{H}^+ = \{f \in \tilde{H} : LT(f) \notin LT(H)\}$;
return $\tilde{H}^+$;

The idea is to obtain $\tilde{H}^+ = \{f \in \tilde{H} : f$ is top-irreducible by $G\}$. A matrix Buchberger Algorithm would apply extended Gaussian reduction on $Mac(G \cup L)$ to reduce all elements in L via elements of G. Instead of this, one first multiplies the rows of G by all needed terms to reduce the elements in L afterwards via standard Gaussian elimination. These rows are found by a so-called $SymbolicPreprocessing$:

Algorithm 5: Symbolic Preprocessing

Input: set of polynomials L, current basis G
Output: set of polynomials H containing L

$H = L$;
$Done := LT(H)$;
while $T(H) \neq Done$ **do**
    Select $m \in T(H) \setminus Done$;
    $Done = Done \cup \{m\}$;
    if $LT(g) | m$ *for some* $g \in G$ **then**
        $H := H \cup \{g \frac{m}{LT(g)}\}$;
    end
end
return H;

Remark 2.5.5. *It would make sense to start with an empty list Done, but it will turn out that the resulting list is the same when starting with* $Done = LT(L)$. *Therefore, this initialization saves some iterations.* [15]

Theorem 2.5.6. *The F4 Algorithm described above is correct.*

To obtain the proof, we need the following lemmas:

Lemma 2.5.7.

(i) Let H be a set of polynomials and $H^- \subseteq H$ with the property that

$$|H^-| = |LT(H)|, \quad LT(H^-) = LT(H). \tag{2.4}$$

Then $G' := \tilde{H}^+ \cup H^-$, interpreted as a set of vectors, is a triangular basis of the vector space generated by the rows of $Mac(H)$. In particular, all polynomials f, whose vector interpretation is in this vector space, can be reduced to 0 via G'.

(ii) Let $\tilde{H}^+ = ReductionF4(L, G)$. Then for all f in the vector space generated by L, $f \xrightarrow[G \cup \tilde{H}^+]{*} 0$.

Proof. (compare [22, Theorem 4.14])

(i): Since all leading terms in G' are different, the corresponding vectors are linearly independent. Furthermore,

$$LT(G') = LT(\tilde{H}^+) \cup LT(H^-) = LT(\tilde{H}^+) \cup LT(H) \supseteq LT(\tilde{H})$$

holds where $\tilde{H}$ denotes the reduced row echelon form of $Mac(H)$. Since $\tilde{H}$ generates the vector space which is spanned by $Mac(H)$ and

$$|G'| = |LT(G')| \geq |LT(\tilde{H})| = |\tilde{H}|,$$

the result follows.

(ii): Let H be the output of $SymbolicPreprocessing(L, G)$. By construction, $L \subseteq H$ and therefore, f is in the vector space generated by $Mac(H)$. Taking a subset $H^- \subseteq H$ with properties as in (2.4), we know by (i) that f can be written as a linear combination

$$f = \sum_{i=1}^{k} c_i h_i^+ + \sum_{j=1}^{l} c'_j h_j^-, \quad h_i^+ \in \tilde{H}^+, h_j \in H^-, c_i, c'_j \in \mathbb{K}.$$

Since h_j^- was constructed during Symbolic Preprocessing, $h_j^- = t_j g_{j'}$ for some $t \in T, g_{j'} \in G$. Thus, we can write

$$f = \sum_{i=1}^{k} c_i h_i^+ + \sum_{j=1}^{l} c'_j t_j g_{j'}, \quad g_{j'} \in G.$$

Since all leading terms of $\tilde{H}^+ \cup H^-$ differ, we can iteratively reduce f either via $c_i h_i^+$ or via $(c_j t_j) g_{j'}$ and hence, $f \xrightarrow[G \cup \tilde{H}^+]{*} 0$ follows.

□

Lemma 2.5.8. *Let $\tilde{H}^+$ be the output of ReductionF4(L, G) in some iteration. Then for all $h \in \tilde{H}^+, LT(h) \notin \langle LT(G)\rangle$.*

Proof. (compare [22, Lemma 4.18]) Assume to the contrary that there exists an $h \in \tilde{H}^+$ such that $t = LT(h) \in \langle LT(G)\rangle$. Then there is a $g \in G$ such that $HT(g)|HT(h)$. Since

$$t \in T(\tilde{H}^+) \subseteq T(\tilde{H}) \subseteq T(F),$$

t has been selected in *SymbolicPreprocessing* and $g\frac{t}{LT(g)}$ (or some other polynomial with leading term t) has been added to H. This implies $LT(h) = t \in LT(H)$, a contradiction. □

Proof of Theorem 2.5.6. The termination follows directly from Lemma 2.5.8 and the Noetherian property of P. Note that we start with $G = F$ and all elements in $\tilde{H}^+$ obtained during the algorithm are linear combinations of elements in L or monomial multiples of polynomials in G. Since these are both contained in $\langle G\rangle$, we have $\langle G\rangle = \langle F\rangle$ during the whole algorithm, hence G is a basis for $\langle F\rangle$.
To prove the Gröbner property, we show that for all $(f, g) \notin B$

$$\mathit{Spol}(f, g) \xrightarrow[G]{*} 0$$

holds after the iteration when (f, g) was removed from B. To see this, observe that if $(f, g) \notin B$, it has been selected by some *Sel* already. Hence $\mathit{Spol}(f, g)$ is in the vector space generated by L and therefore, $\mathit{Spol}(f, g) \xrightarrow[G]{*} 0$ follows from Lemma 2.5.7.

□

The big advantage of F4 in comparison to the Buchberger Algorithm is that many S-polynomials are reduced at the same time. The reduction is computed by (normal) Gaussian elimination which is a well-studied algorithm where highly optimized matrix algorithms can be applied. This tends to be computationally much faster than working with polynomials.

2.6 M4GB Algorithm

2.6.1 Idea

In this section, we describe a rather new algorithm which seems to compute Gröbner basis empirically very fast. The idea is similar to F4, but reduces the S-polynomials sequentially and not as a set. We will take the ideas from [20], but describe an easier variant which contains the main ideas. This would lead to a less efficient algorithm when implemented in precisely that manner, but should be easier to understand. The overall structure of the algorithm is basically the same as in the Buchberger Algorithm, but we reduce polynomials only by so-called tail-irreducible elements:

Definition 2.6.1 (Tail-irreducibility). *Let $f \in P$, $G \subseteq P$. We define ...*

(i) ... $Tail(f) := f - LM(f)$.

(ii) ... $T_G(f) := \{t \in T(f) : t$ is reducible with respect to $G\}$.

*(iii) A polynomial f is called **tail-irreducible** (with respect to G) if $T_G(Tail(f)) = \emptyset$.*

Let f be an S-polynomial which needs to be reduced by a set G (like in the Buchberger Algorithm) and let $t \in T_G(f), C_t(f) = c$. Assume we have a polynomial $h \in \langle G \rangle : LT(h) = t$ which is tail-irreducible with respect to G and we apply the reduction step

$$f \to f - h.$$

The tail-irreducibility of h leads to the following advantage: If h is not known to be tail-irreducible, we would have to consider all terms

$$s \in T(f - h) \subseteq (T(f) \cup T(h)) \backslash \{t\}$$

and check whether they are reducible by G. However by construction, $T_G(f - h) = T_G(f) \backslash \{t\}$ and hence, no new reducible terms are created in the reduction process. This leads in most cases to a faster reduction of f. To construct such h in a computationally cheap way, we observe the following: Since $t \in T_G(f)$, there exists $g \in G, v \in T$ such that $LT(vg) = t$. Doing a reduction on $Tail(vg)$ with respect to G yields by adding $LM(vg)$ a tail-irreducible polynomial with leading term t. To make this algorithm more efficient, one saves these tail-irreducible elements in a set of polynomials M. The advantage is the following: If $t \in T_G(g)$ for some further S-polynomial g, we already have such a tail-irreducible element with leading term t and can reduce g by this element straight away. However, we need to be careful about the following problem: If we add a new polynomial f' to G, this element might no longer be tail-irreducible. One could update all those elements in M instantly, but to save some computation time, we install so-called generations: If some polynomial f' is added to G, a new generation starts. All elements in M now belong to an earlier generation and hence, they might be tail-reducible. Therefore, we need to check if those polynomials added to G after the creation of tail-irreducible elements in M might reduce those. If so, we need to tail-reduce these elements until they are tail-irreducible again. To work out this idea more precisely, consider the following pseudocode:

Algorithm 6: M4GB main routine

Input: Set of polynomials $F = \{f_1, ..., f_k\}$.

Output: Gröbner basis G of $\langle F \rangle$ with $F \subseteq G$.

Set $G = F$;

Set $M = G$;

Set $P = \{Spol(g_i, g_j) : g_i, g_j \in G\}$;

while $P \neq \emptyset$ **do**
  Select some $f = Spol(g_i, g_j)$ from P;
  $P = P \backslash \{f\}$;
  $f' := M4GB\text{-}Reduction(f, G, M)$;
  if $f' \neq 0$ **then**
    Set $P = P \cup \{Spol(f', g) : g \in G\}$;
    Set $G = G \cup \{LC(f')^{-1} \cdot f'\}$;
    Set $M = M \cup \{LC(f')^{-1} \cdot f'\}$;
    Set new generation;
  end
end
return G;

Algorithm 7: M4GB-Reduction

Input: polynomial f, sets of polynomials G, M.
Output: polynomial $f' \in \overline{f}^G$, possibly updated M.

Set $f' = f$;
for $t \in T(f)$ **do**
  if $\exists m \in M, v \in T : LT(vm) = t$ **then**
    if $\exists m' \in M : LT(m') = t$ **then**
      if m' *is from earlier generation* **then**
        $m'' = M4GB\text{-}Reduction(Tail(m'), G, M)$;
        Replace m' by m'' in M and in G if $m' \in G$;
        Mark m'' with current generation;
        $m' = m''$;
      end
    end
    else
      $m' = M4GB\text{-}Reduction(Tail(vm), G, M)$;
      Add m' to M;
      Mark m' with current generation;
    end
    $f' = f' - C_t(f)m'$;
  end
end
return f';

2.6.2 Correctness Proof

We will only sketch the proof of correctness here. For a more detailed proof see Remark 4.2.7, since the correctness of the M4GB Algorithm follows from a special case of the statements proven in Chapter 4. Consider the following definition for a short proof sketch:

Definition 2.6.2 (M4GB-Invariant). *A tuple of sets (G, M) where $G, M \subseteq P$ is said to fulfill the **M4GB invariant** if it fulfills the following properties:*

(i) All leading terms in M are unique, i.e. $f, g \in M, LT(f) = LT(g)$ implies $f = g$.

(ii) $G \subseteq M$.

(iii) All elements in M are top-reducible by G.

Lemma 2.6.3. *(G, M) fulfills the M4GB invariant during the whole algorithm.*

Proof. The M4GB invariant holds trivially after initialization for input sets F with distinct leading terms. Reducing all polynomials $f \in F$ by $F \setminus \{f\}$, we can assume without loss of generality that F fulfills this property. To see that (i) holds, note that we add a polynomial m to M only if $LT(m) \notin M$, otherwise we replace the old element. For (ii), note that if we add some polynomial g to G, we add it to M as well. To prove (iii), we just need to look at the construction of polynomials in M: Those polynomials are constructed by tail-reducing top-reducible elements and hence, are top-reducible by themselves. □

Proposition 2.6.4. *If f is reduced by some $m \in M$ during the execution of M4GB-reduction, then m is tail-irreducible with respect to G.*

Proof. If $m \in M$, it was tail-irreducible at the point of its creation by definition. Observe that m stays tail-irreducible until we add some new polynomial to G and therefore, increase the generation. Since we reduce f only by polynomials which are marked with the current generation, m is tail-irreducible. □

Lemma 2.6.5. *Let (G, M) fulfill the M4GB invariant and let $\tilde{f}$ be the output of M4GB-Reduction(f, G, M). Then $f' \in \overline{f}^G$.*

Proof. See Remark 4.2.7. □

The correctness of Algorithm 3 follows now immediately by Theorem 2.1.9 since the output is the same as in the Buchberger Algorithm.

Lemma 2.6.6. *Algorithm 3 terminates.*

Proof. The while-loop terminates once more by the Noetherian property and the same argumentations as in the Buchberger Algorithm. To see that the iterative call of M4GB-reduction terminates, note that the leading terms of the polynomials in a chain of such calls strictly decrease. Hence, an infinite recursion depth would lead to an infinite strictly descending chain of polynomials, a contradiction. □

3 Signature Gröbner bases/F5

3.1 Basic definitions

In this section, we will introduce the basic ideas of signature Gröbner bases. The aim of constructing these bases is to detect reductions to zero in advance. The following discussion is based on [8], [10], and [17].

3.1.1 The module P^m and signatures

The idea is, given an input set $F = \{f_1, ..., f_m\}$, to employ the free P-module P^m with generators $\boldsymbol{f_1}, ..., \boldsymbol{f_m}$. Note that this notation is taken on purpose as we will see later. To make use of this module, we need to define module terms and extend the term order on P^m:

Definition 3.1.1 (Module terms). *Let $\boldsymbol{f_i}$ denote the i-th generator of P^m and let $<$ be a term order. We define ...*

*(i) ... the **module terms***

$$T_m := \{t\boldsymbol{f_i}, t \in T, i \in \{1, ..., m\}\}.$$

*(ii) ... a **compatible extension of** $<$ to T_m as a total order $\boldsymbol{<}$ that fulfills:*

$$\forall t, u \in T, \forall i \in \{1, ..., m\} : t < u \Rightarrow t\boldsymbol{f_i} \boldsymbol{<} u\boldsymbol{f_i}.$$

Definition 3.1.2 (Important extensions on T_m). *Let $<$ be a term order on T, $t, u \in T$, $i, j \in \{1, ..., m\}$. Remember the evaluation homomorphism*

$$\begin{aligned} v = v_F : P^m &\to P \\ h &\mapsto \sum_{i=1}^{m} h_i f_i. \end{aligned}$$

We define ...

(i) ... the position over term extension $t\boldsymbol{f_i} \boldsymbol{<_{pot}} u\boldsymbol{f_j} \Leftrightarrow \begin{cases} i > j \text{ or} \\ i = j, t < u. \end{cases}$

(ii) ... the term over position extension $t\boldsymbol{f_i} \boldsymbol{<_{top}} u\boldsymbol{f_j} \Leftrightarrow \begin{cases} t < u \text{ or} \\ t = u, i > j. \end{cases}$

(iii) ... the weighted order extension $t\boldsymbol{f_i} <_{\boldsymbol{w}} u\boldsymbol{f_j} \Leftrightarrow \begin{cases} LT(v(t\boldsymbol{f_i})) < LT(v(u\boldsymbol{f_j})) \text{ or} \\ LT(v(t\boldsymbol{f_i})) = LT(v(u\boldsymbol{f_j})), i > j. \end{cases}$

Definition 3.1.3. *Let*

$$\boldsymbol{g} = \sum_{i=1}^{m} g_i \boldsymbol{f_i}, \quad \boldsymbol{h} = \sum_{i=1}^{m} h_i \boldsymbol{f_i}$$

be two nonzero module elements where

$$g_i = \sum_{j=1}^{n_i} c_j g_{i,j}, c_j \in \mathbb{K}\backslash\{0\}, g_{i,j} \in T.$$

Furthermore, let $<$ *be a total order on* T_m*. We define ...*

(i) ... the ***module leading term*** *of* g *as*

$$MLT(g) := \max\{g_{i,j}\boldsymbol{f_i} : i \in \{1, ..., m\}, j \in \{1, ..., n_i\}\}$$

where the maximum is taken with respect to $<$.

(ii) ... the ***module leading monomial*** *of* g *as*

$$MLM(g) := c_{i,j} g_{i,j} \boldsymbol{f_i}$$

where $g_{i,j}\boldsymbol{f_i} = MLT(\boldsymbol{g})$.

(iii) We extend $<$ *to* P^m *in the following way: Let* $\boldsymbol{g}, \boldsymbol{h} \in P^m$*, then*

$$\boldsymbol{g} < \boldsymbol{h} :\Leftrightarrow MLT(\boldsymbol{g}) < MLT(\boldsymbol{h}).$$

If $MLT(\boldsymbol{g}) = MLT(\boldsymbol{h})$*, we define equality with respect to* $<$ *as* $\boldsymbol{g} \sim \boldsymbol{h}$ *to not confuse this with equality in common sense.*

(iv) We say $\boldsymbol{h} \,|\, \boldsymbol{g}$ *if there exists a monomial* m *such that* $m\boldsymbol{h} = \boldsymbol{g}$.

(v) We call a module element $\boldsymbol{g}$ ***monic*** *if* $LC(v(\boldsymbol{g})) = 1$.

Remark 3.1.4.

1. *It is easy to see that* $<$ *forms a quasi-order on* P^m*. If we say* $\boldsymbol{g} \leq \boldsymbol{h}$*, we mean* $\boldsymbol{g} < \boldsymbol{h}$ *or* $\boldsymbol{g} \sim \boldsymbol{h}$.

2. *As in many definitions of divisibility and orders,* $\boldsymbol{h} \,|\, \boldsymbol{g}$ *implies* $\boldsymbol{h} \leq \boldsymbol{g}$.

3. *If* $<$ *is a compatible extension of a given term order,*

$$MLT(g) = max\{LT(g_i)\boldsymbol{f_i} : i \in \{1, ..., m\}\}.$$

To see the connection between P^m and $\langle F \rangle$, observe that $v_F(P^m) = \langle F \rangle$. This implies that each element in P^m can be associated with a unique element of $\langle F \rangle$, induced by this homomorphism. We will use the additional algebraic structure on P^m to get rid of many useless S-polynomials. For that, we introduce the notion of signatures:

Definition 3.1.5 (Signature). *Let $\boldsymbol{f} \in P^m \setminus \{\boldsymbol{0}\}$ and let $<$ be a compatible extension of a term order on P^m. Then we define the **signature** of a module element $\boldsymbol{f}$ as*

$$Sig(\boldsymbol{f}) := MLM(\boldsymbol{f}).$$

We create the formal symbol ∞ as a signature with the property

$$\forall \boldsymbol{f} \in P^m \setminus \{0\} : Sig(\boldsymbol{f}) < \infty.$$

Remark 3.1.6. *Note that the set of signatures is $S := \{c\boldsymbol{t} : c \in \mathbb{K} \setminus \{0\}, t \in T_m\} \cup \{\infty\}$. Furthermore, observe that for all signatures $c\boldsymbol{t} \in S$, we have that $c\boldsymbol{t} \sim \boldsymbol{t}$. It is easy to check that $\sim$ forms an equivalence relation and hence, ${}^{S}\!/\!{}_{\sim} \cong T_m \cup \{\infty\}$.*

As a next step, we need to define the reduction on P^m equivalent to "normal" reduction on polynomials to fit our purposes. Note the similar notation to normal reduction defined in Definition 2.1.1.

3.1.2 Reduction on P^m

Definition 3.1.7 (*Sig*-(top-)reduction). *Let $\boldsymbol{f} \in P^m, \boldsymbol{G} \subseteq P^m \setminus \{\boldsymbol{0}\}$ a finite set of monic module elements.*

- *$\boldsymbol{f}$ is said to be **Sig-reducible** (with respect to $\boldsymbol{G}$) if there exist $t \in T(v(\boldsymbol{f})), \boldsymbol{g} \in \boldsymbol{G}$:*

  (i) $LT(v(\boldsymbol{g}))|t$, in that case set $u = \frac{t}{LT(v(g))}$.

  (ii) $Sig(\boldsymbol{f}) \geq Sig(u\boldsymbol{g})$.

  If these properties are fulfilled, we define $\boldsymbol{f} - C_t(f)u\boldsymbol{g}$ as the outcome of the Sig-(top-)reduction. We denote this by

  $$\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{} \boldsymbol{f} - C_t(f)u\boldsymbol{g}$$

  and say

  $$\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} \boldsymbol{h}$$

  if finitely many reduction steps are done. This also includes the case that no Sig-reduction steps are done at all, hence $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} \boldsymbol{f}$ follows trivially.

- *If $\boldsymbol{f}$ is not Sig-reducible, we call it **Sig-irreducible**.*

- *If $t = LT(f)$, we call it a **Sig-top-reduction**.*

- *We say* $\boldsymbol{g}$ ***Sig-(top-)reduces*** $\boldsymbol{f}$ *and* $\boldsymbol{f}$ *is* ***Sig-(top-)reduced*** *to* $\boldsymbol{f} - u\boldsymbol{g}$.

- *If* $Sig(\boldsymbol{f}) > Sig(u\boldsymbol{g})$, *we call it a* ***regular*** *reduction. We denote this by*

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G,reg}]{} \boldsymbol{f} - u\boldsymbol{g}$$

and a finite number of regular reductions to a module element $\boldsymbol{h}$ *by*

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G,reg}]{*} \boldsymbol{h}.$$

- *If* $Sig(\boldsymbol{f}) \sim Sig(u\boldsymbol{g})$, *we call it a* ***singular*** *reduction.*

- *We say* $\boldsymbol{f'} \in P^m$ *is a* ***normal form of*** $\boldsymbol{f}$ ***and*** $\boldsymbol{G}$ *if* $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} \boldsymbol{f'}$ *and* $\boldsymbol{f'}$ *is Sig-irreducible. We define*

$$\overline{\boldsymbol{f}}^{\boldsymbol{G}} := \{\boldsymbol{f'} \in P^m : \boldsymbol{f'} \text{ is a normal form of } \boldsymbol{f} \text{ and } \boldsymbol{G}\}.$$

- *Similarly, we denote term order on polynomials by* $<$ *and order extensions on module terms as* $\boldsymbol{<}$.

- *Analogously, we say* $\boldsymbol{f'}$ *is a* ***regular normal form of*** $\boldsymbol{f}$ ***and*** $\boldsymbol{G}$ *if* $\boldsymbol{f} \xrightarrow[\boldsymbol{G,reg}]{*} \boldsymbol{f'}$ *and* $\boldsymbol{f'}$ *is regularly Sig-irreducible. We define*

$$\overline{\boldsymbol{f}}^{\boldsymbol{G,reg}} := \{\boldsymbol{f'} \in P^m : \boldsymbol{f'} \text{ is a regular normal form of } \boldsymbol{f} \text{ and } \boldsymbol{G}\}.$$

- *We say*

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$$

if $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} \boldsymbol{h}$ *where* $\boldsymbol{h}$ *is a syzygy. This notation is justified by* $v(\boldsymbol{h}) = 0$.

Remark 3.1.8. *To emphasize the difference between polynomials and module elements, we will use the following notations:*

- *For polynomials, we use a normal lower letter, e.g.* $g \in P$.

- *For vectors in* P^m, *we use a bold lower letter, e.g.* $\boldsymbol{g} = (g_1, ..., g_m) \in P^m$. *To be consistent, we write* $\mathbf{0}$ *for the zero element in* P^m.

- *In that way, we are consistent with using* $\boldsymbol{f_i}$ *for the unit vectors of* P^m *and* f_i *for the polynomials spanning the ideal* $\langle F \rangle$ *since* $v(\boldsymbol{f_i}) = f_i$.

- *Analogously, we handle the set equivalent: If some set is a subset of* P, *we write a normal capital letter, e.g.* $G \subseteq P$, *for subsets of* P^m, *we write a bold capital letter, e.g.* $\boldsymbol{G} \subseteq P^m$.

- *Note that both* $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$ *and* $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} \boldsymbol{0}$ *are well-defined, but differ in meaning:* $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$ *only signals that* $\boldsymbol{f}$ *can be reduced to a syzygy whereas* $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} \boldsymbol{0}$ *means that* $\boldsymbol{f}$ *can actually be reduced to the zero element of* P^m.

To shorten the notations, we make the following assumptions:

- *If not stated otherwise, the evaluation homomorphism is* $v = v_F$ *where* $F = \{f_1, ..., f_m\}$. *All other properties (reducible/irreducible,...) are implicitly defined with respect to* G *respectively* $\boldsymbol{G}$, *depending on whether the elements are polynomials or module elements. Furthermore, all elements in* $\boldsymbol{G}$ *are assumed to be nonzero and monic. We can assure that in every algorithmic consideration by computing* $\boldsymbol{g'} = LC(g)^{-1}\boldsymbol{g}$, *not losing any property needed.*

- *For given* $\boldsymbol{g} \in P^m$, *we define* $g := v(\boldsymbol{g})$.

- *Analogously, for given* $\boldsymbol{G} \subseteq P^m$, *we define* $G := v(\boldsymbol{G})$.

Remark 3.1.9. *If* $F = \{f_1, ..., f_m\}$ *is the input set, we always assume this input set to be interreduced, meaning that* f_i *is irreducible with respect to* $F \backslash \{f_i\}$. *We can state this without loss of generality since we can do a corresponding preprocessing, not changing the ideal generated by the polynomials.*

3.1.3 Signature Gröbner bases

Definition 3.1.10 (Signature Gröbner basis). *Let* $F = \{f_1, ..., f_m\}, \boldsymbol{s} \in T_m \cup \{\infty\}$. *We define* $\boldsymbol{G} \subseteq P^m$ *to be a signature Gröbner basis for* $\langle F \rangle$ *up to signature* $\boldsymbol{s}$ *if for all* $\boldsymbol{f} \in P^m$ *such that* $Sig(\boldsymbol{f}) < \boldsymbol{s}$:

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0.$$

If $\boldsymbol{G}$ *is a signature Gröbner basis up to* ∞ *(i.e. for all possible signatures),* $\boldsymbol{G}$ *is a signature Gröbner basis.*

Remark 3.1.11. *Note that* $\boldsymbol{h} \in P^m$ *is a syzygy if and only if* $\boldsymbol{h}$ *is Sig-irreducible and* $\boldsymbol{h} \xrightarrow[\boldsymbol{G}]{*} 0$.

Proposition 3.1.12. *If* $\boldsymbol{G}$ *is a signature Gröbner basis for* $\langle F \rangle$, *then* $G := v(\boldsymbol{G})$ *is a Gröbner basis for* $\langle F \rangle$.

Proof. Since $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$ implies $f \xrightarrow[v(\boldsymbol{G})]{*} 0$, this follows directly by a standard characterization of Gröbner bases. □

3.2 First algorithmic ideas

3.2.1 Criterion for Signature Gröbner bases

A natural question arising is how to determine efficiently when a set of module elements forms a signature Gröbner basis. As in the polynomial case, S-polynomials, now defined on module elements, play a crucial role:

Definition 3.2.1 (S-polynomials for module elements). *Let $\boldsymbol{f}, \boldsymbol{g} \in P^m$. We define the S-polynomial of $\boldsymbol{f}$ and $\boldsymbol{g}$ as*

$$Spol(\boldsymbol{f}, \boldsymbol{g}) := \frac{\operatorname{lcm}(LT(f), LT(g))}{LM(f)}\boldsymbol{f} - \frac{\operatorname{lcm}(LT(f), LT(g))}{LM(g)}\boldsymbol{g}.$$

If $Spol(\boldsymbol{f}, \boldsymbol{g}) = u\boldsymbol{f} - v\boldsymbol{g}$ and $Sig(u\boldsymbol{f}) \nsim Sig(v\boldsymbol{g})$, we call the pair $(\boldsymbol{f}, \boldsymbol{g})$ ***regular****, if $Sig(u\boldsymbol{f}) \sim Sig(v\boldsymbol{g})$* ***singular****.*

The following theorem is essential for the computation of signature Gröbner bases. Note the similarity to the criterion for Gröbner bases stated in Theorem 2.1.9:

Theorem 3.2.2. *(compare [17, Theorem 1]) Let $\boldsymbol{s} \in T_m, \boldsymbol{G} \subseteq P^m$ with $\{\boldsymbol{f_i} \leq \boldsymbol{s}, i \in \{1, ..., m\}\} \subseteq \boldsymbol{G}$. Assume that for all regular pairs $(\boldsymbol{g_i}, \boldsymbol{g_j})$ where $\boldsymbol{g_i}, \boldsymbol{g_j} \in \boldsymbol{G}$, $Sig(Spol(\boldsymbol{g_i}, \boldsymbol{g_j})) < \boldsymbol{s}$:*

$$Spol(\boldsymbol{g_i}, \boldsymbol{g_j}) \xrightarrow[\boldsymbol{G},reg]{*} 0 \text{ or } Spol(\boldsymbol{g_i}, \boldsymbol{g_j}) \xrightarrow[\boldsymbol{G},reg]{*} \boldsymbol{h}$$

for $\boldsymbol{h}$ singularly Sig-top-reducible. Then $\boldsymbol{G}$ is a signature Gröbner basis up to $\boldsymbol{s}$. In particular, if $\overline{Spol(\boldsymbol{g_i}, \boldsymbol{g_j})}^{\boldsymbol{G},reg}$ contains a syzygy or a singularly Sig-top-reducible element for all pairs $(\boldsymbol{g_i}, \boldsymbol{g_j})$, then $\boldsymbol{G}$ is a signature Gröbner basis.

To prove Theorem 3.2.2, we need some smaller statements before:

Proposition 3.2.3. *Let $\boldsymbol{f}, \boldsymbol{g} \in P^m \backslash \{0\}, \boldsymbol{G} \subseteq P^m \backslash \{0\}$.*

(i) Let $\boldsymbol{f}, \boldsymbol{g}$ be Sig-top-reducible or syzygies with $Sig(\boldsymbol{f}), Sig(\boldsymbol{g}) \leq Sig(\boldsymbol{f} + \boldsymbol{g})$. Then at least one of the following three conditions is fulfilled:

a) $\boldsymbol{f} + \boldsymbol{g}$ is Sig-top-reducible.

b) $LM(f) = -LM(g)$.

c) $f = g = 0$.

(ii) Let $\boldsymbol{G}$ be a signature Gröbner basis up to $\boldsymbol{s} \in T_m \cup \{\infty\}$ with $\{\boldsymbol{f_i} : \boldsymbol{f_i} \leq \boldsymbol{s}, i \in \{1, ..., m\}\} \subseteq \boldsymbol{G}$ and let $\boldsymbol{f}$ be not a syzygy, but Sig-top-irreducible with $Sig(\boldsymbol{f}) \sim \boldsymbol{s}$. Then

$$LM(f) = -LM(f - s).$$

(iii) Let α, β be monomials and let $\boldsymbol{p} = \alpha\boldsymbol{f} - \beta\boldsymbol{g} \in P^m$. Assume that $LM(\alpha f) = LM(\beta g)$ and $\gcd(\alpha, \beta) = 1$. Then $\boldsymbol{p} = Spol(\boldsymbol{f}, \boldsymbol{g})$.

Proof.

(i): Assume $LM(f) \neq -LM(g)$ and $\boldsymbol{f}, \boldsymbol{g}$ are not syzygies. Then we can assume without loss of generality that $LT(f) = LT(f + g)$. As $Sig(\boldsymbol{f}) \leq Sig(\boldsymbol{f} + \boldsymbol{g})$, this implies that $\boldsymbol{f} + \boldsymbol{g}$

is Sig-top-reducible by the top-reducer of $\boldsymbol{f}$. If, without loss of generality, $\boldsymbol{g}$ is a syzygy but $\boldsymbol{f}$ is Sig-top-reducible, then once again $LT(f) = LT(f+g)$, and the statement holds by the same argument already done. If both $\boldsymbol{f}, \boldsymbol{g}$ are syzygies, then $f = g = 0$ holds by definition.

(ii): Since $Sig(\boldsymbol{f} - c\boldsymbol{s}) < \boldsymbol{s}$ for suitable $c \in \mathbb{K}\backslash\{0\}$ and $\boldsymbol{G}$ is a Gröbner basis up to $\boldsymbol{s}$, we know that $\boldsymbol{f} - c\boldsymbol{s} \xrightarrow[\boldsymbol{G}]{*} 0$. This implies that $\boldsymbol{f} - c\boldsymbol{s}$ is Sig-top-reducible or a syzygy. Furthermore, since $\boldsymbol{f_i} \in \boldsymbol{G}$, $c\boldsymbol{s}$ is Sig-top-reducible by $\boldsymbol{G}$. By assumption, $\boldsymbol{f} = (\boldsymbol{f} - c\boldsymbol{s}) + c\boldsymbol{s}$ is Sig-top-irreducible. Using (i), we get that $LM(f) = -LM(f-s)$ or $f = f - s = 0$. Since $f_i \neq 0$ and therefore $s \neq 0$,

$$LM(f) = -LM(f-s)$$

follows.

(iii): Since $LM(\alpha f) = LM(\beta g)$, there exists a monomial d such that

$$d\,\mathrm{lcm}(LT(g), LT(f)) = LM(f)\alpha = LM(g)\beta.$$

As $\boldsymbol{p} = dSpol(\boldsymbol{f}, \boldsymbol{g})$ and $gcd(\alpha, \beta) = 1$, we obtain $d = 1$ and the results follows.

$\square$

Remark 3.2.4. *The statements from Proposition 3.2.3 are implicitly assumed to be fulfilled in [10]. We filled this gap not knowing if the authors left this out on purpose due to space limitations.*

Lemma 3.2.5. *Let $\boldsymbol{f} \neq \boldsymbol{0} \in P^m$ be Sig-top-irreducible and let $\boldsymbol{G} \subseteq P^m$ be a signature Gröbner basis up to $Sig(\boldsymbol{f})$ where $\{\boldsymbol{f_i} : \boldsymbol{f_i} \leq Sig(\boldsymbol{f}), i \in \{1, ..., m\}\} \subseteq \boldsymbol{G}$. Then there exists a regular pair $(\boldsymbol{g_i}, \boldsymbol{g_j})$ with $\boldsymbol{g_i}, \boldsymbol{g_j} \in \boldsymbol{G}$ such that $Sig(Spol(\boldsymbol{g_i}, \boldsymbol{g_j})) \mid Sig(\boldsymbol{f})$.*

Proof. (compare [17, Lemma 9]) Let $Sig(\boldsymbol{f}) \sim \boldsymbol{s} = t\boldsymbol{f_i} \in T_m$. As seen in the proof of Proposition 3.2.3(ii), for suitable $c \in \mathbb{K}$,

$$\boldsymbol{f} - c\boldsymbol{s} \xrightarrow[\boldsymbol{G}]{*} 0.$$

If $f - cs = 0$, $ct\boldsymbol{f_i}$ would be a (singular) Sig-top-reducer of $\boldsymbol{f}$, a contradiction to $\boldsymbol{f}$ being Sig-top-irreducible. Hence, there exists a monomial m and some $\boldsymbol{g} \in \boldsymbol{G}$ such that $LM(mg) = LM(f-s)$ with

$$Sig(m\boldsymbol{g}) \leq Sig(\boldsymbol{f} - \boldsymbol{s}) < Sig(\boldsymbol{f}).$$

As $c\boldsymbol{s}$ and $\boldsymbol{f} - c\boldsymbol{s}$ are both Sig-top-reducible, but $\boldsymbol{f}$ is not, it follows from Proposition 3.2.3(i) that

$$LM(tf_i) = LM(s) = -LM(f-s) = -LM(mg). \tag{3.1}$$

Defining

$$\boldsymbol{p} := \alpha \boldsymbol{f_i} - \beta \boldsymbol{g} \quad \text{with } \alpha := \frac{t}{\gcd(t,m)}, \quad \beta := \frac{m}{\gcd(t,m)},$$

we have $LM(\alpha f_i) = -LM(\beta g)$ and $\gcd(\alpha, \beta) = 1$. Applying Proposition 3.2.3(iii), we see that $\boldsymbol{p} = Spol(\boldsymbol{f_i}, \boldsymbol{g})$. Since $Sig(m\boldsymbol{g}) < Sig(\boldsymbol{f})$, we have

$$\gcd(t, m) Sig(\boldsymbol{p}) = Sig(t\boldsymbol{f_i} - m\boldsymbol{g}) = Sig(t\boldsymbol{f_i}) = \boldsymbol{s},$$

which completes the proof. □

Lemma 3.2.6. *Let $\boldsymbol{G}$ be a signature Gröbner basis up to $\boldsymbol{s}$ and let $(\boldsymbol{g_i}, \boldsymbol{g_j})$ with $\boldsymbol{g_i}, \boldsymbol{g_j} \in \boldsymbol{G}$ be a regular pair such that $Sig(Spol(\boldsymbol{g_i}, \boldsymbol{g_j})) \mid \boldsymbol{s}$. Then there exists a monomial b and $\boldsymbol{g_{i'}}, \boldsymbol{g_{j'}} \in \boldsymbol{G}$ such that $\boldsymbol{q} := Spol(\boldsymbol{g_{i'}}, \boldsymbol{g_{j'}})$ is regular with the following properties:*

(i) $Sig(b\boldsymbol{q}) = \boldsymbol{s}$.

(ii) $b\boldsymbol{q'}$ is regularly Sig-top-irreducible for some $\boldsymbol{q'} \in \overline{\boldsymbol{q}}^{\boldsymbol{G},reg}$.

Proof. (compare [17, Lemma 10]) Let a be the monomial such that $Sig(a\boldsymbol{p}) = \boldsymbol{s}$ and let $\boldsymbol{p'} \in \overline{\boldsymbol{p}}^{\boldsymbol{G},reg}$. If $a\boldsymbol{p'}$ is regularly Sig-top-irreducible, we set $a = b, \boldsymbol{q} = \boldsymbol{p}$ and are done. So from now on, we can assume that $a\boldsymbol{p'}$ is regularly Sig-top-reducible. To prove the rest of the statement, we construct an S-polynomial $\boldsymbol{q} = Spol(\boldsymbol{g_k}, \boldsymbol{g_l})$ for $\boldsymbol{g_k}, \boldsymbol{g_l} \in \boldsymbol{G}$ and a monomial b with the following properties:

(i) $Sig(b\boldsymbol{q}) = \boldsymbol{s}$.

(ii) $LT(ap) > LT(bq)$.

We can find this $b\boldsymbol{q}$ in the following way: If $a \in \mathbb{K}$, then $a\boldsymbol{p'}$ is regularly Sig-top-reducible, but $\boldsymbol{p'}$ Sig-top-irreducible is a contradiction. Hence $a > 1$ and therefore $Sig(\boldsymbol{p}) < \boldsymbol{s}$ follows. Due to the property of $\boldsymbol{G}$ being a signature Gröbner basis up to $\boldsymbol{s}$, we know that

$$\boldsymbol{p} \xrightarrow[\boldsymbol{G}]{*} 0.$$

Observe that since $a\boldsymbol{p'}$ is regularly Sig-top-reducible, $\boldsymbol{p'}$ cannot be a syzygy. Hence, there must exist a **singular** Sig-top-reducer $v\boldsymbol{g_k}$ with v a monomial, $\boldsymbol{g_k} \in \boldsymbol{G}$ such that

$$Sig(v\boldsymbol{g_k}) \sim Sig(\boldsymbol{p'}), \quad LM(vg_k) = LM(p'). \tag{3.2}$$

On the other hand, since $a\boldsymbol{p'}$ is regularly Sig-top-reducible, there exists a monomial u and some $\boldsymbol{g_l} \in \boldsymbol{G}$ such that

$$Sig(u\boldsymbol{g_l}) < Sig(a\boldsymbol{p'}), \quad LM(ug_l) = LM(ap'). \tag{3.3}$$

Combining (3.2) and (3.3), we get that

$$LM(avg_k) = LM(ug_l), \quad Sig(av\boldsymbol{g_k}) > Sig(u\boldsymbol{g_l}).$$

If we define $b := \gcd(av, u)$ and $\boldsymbol{q} := av\boldsymbol{g_k} - u\boldsymbol{g_l}$, we get due to Proposition 3.2.3(iii) that $\boldsymbol{q} = bSpol(\boldsymbol{g_k}, \boldsymbol{g_l})$. Note that

$$\begin{aligned} Sig(\boldsymbol{q}) &= Sig(av\boldsymbol{g_k}) = Sig(a\boldsymbol{p'}) = Sig(a\boldsymbol{p}) = \boldsymbol{s}, \\ LT(b\boldsymbol{q}) &< LT(av\boldsymbol{g_k}) = LT(a\boldsymbol{p'}) \leq LT(a\boldsymbol{p}), \end{aligned}$$

which completes the construction defined above.
If $b\boldsymbol{q'}$ is regularly Sig-top-irreducible for $\boldsymbol{q'} \in \overline{\boldsymbol{q}}^{\boldsymbol{G},reg}$, we are done. Otherwise, we can repeat this construction for $b\boldsymbol{q'}$ instead of $a\boldsymbol{p'}$. Since there can only be a finite descending chain of leading terms, we will eventually obtain an S-polynomial fulfilling the desired properties. □

We are left to state one final lemma before we can finally prove Theorem 3.2.2. Note that this lemma does not only play a role in the current proof, but will be referred to rather often in the following chapters.

Lemma 3.2.7. *Let $\boldsymbol{f}, \boldsymbol{g} \in P^m$ and let $\boldsymbol{G}$ be a signature Gröbner basis up to $Sig(\boldsymbol{f}) = Sig(\boldsymbol{g}) = \boldsymbol{s}$. If $\boldsymbol{f}, \boldsymbol{g}$ are regularly Sig-irreducible, then $f = g$. In particular, if $\boldsymbol{f}, \boldsymbol{g}$ are both regularly Sig-top-irreducible, then $LT(f) = LT(g)$ or $f = g = 0$.*

Proof. (compare [17, Lemma 2]) First note that due to $Sig(\boldsymbol{f}) = Sig(\boldsymbol{g})$, $Sig(\boldsymbol{f} - \boldsymbol{g}) < \boldsymbol{s}$. Since $\boldsymbol{G}$ is a Gröbner basis up to $\boldsymbol{s}$, $\boldsymbol{f} - \boldsymbol{g} \xrightarrow[\boldsymbol{G}]{*} 0$. Assume to the contrary that $f - g \neq 0$, this implies that $\boldsymbol{f} - \boldsymbol{g}$ is Sig-top-reducible. Hence, there exist $t \in T, \boldsymbol{h} \in \boldsymbol{G}$:

$$Sig(t\boldsymbol{h}) \leq Sig(\boldsymbol{f} - \boldsymbol{g}) < \boldsymbol{s}, \quad LT(th) = LT(f - g).$$

Note that $LT(f - g) \in T(f) \cup T(g)$, without loss of generality, $LT(f - g) \in T(f)$. This implies that $\boldsymbol{f}$ is regularly reducible by $\boldsymbol{h}$, a contradiction. The "in particular" statement follows immediately. □

After all these considerations, we can prove the actual result:

Proof of Theorem 3.2.2. (compare [17, Theorem 1]) Assume to the contrary that there exists a $\boldsymbol{f} \in P^m$ such that $Sig(\boldsymbol{f}) < \boldsymbol{s}$, but $\boldsymbol{f}$ cannot be reduced by $\boldsymbol{G}$ to a syzygy. Taking $\boldsymbol{f}$ as the one with the smallest signature among those elements, $\boldsymbol{G}$ is a signature Gröbner basis up to $Sig(\boldsymbol{f})$. We can assume without loss of generality that $\boldsymbol{f}$ is Sig-irreducible, otherwise we could take some $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G}}$ instead. Applying Lemma 3.2.5 and 3.2.6, we obtain the existence of a monomial a and a regular S-polynomial $\boldsymbol{p} = Spol(\boldsymbol{g_i}, \boldsymbol{g_j})$ with $\boldsymbol{g_i}, \boldsymbol{g_j} \in \boldsymbol{G}$ such that the following conditions are fulfilled:

- $Sig(a\boldsymbol{p}) = Sig(\boldsymbol{f})$,
- $a\boldsymbol{p'}$ is regularly Sig-top-irreducible where $\boldsymbol{p'} \in \overline{\boldsymbol{p}}^{\boldsymbol{G},reg}$.

As $\boldsymbol{f}$ is not a syzygy, we can apply Lemma 3.2.7 and obtain by the "in particular" part that $LT(f) = LT(ap')$. This implies that $\boldsymbol{p'}$ is not a syzygy and hence, by assumption must be singularly Sig-top-reducible. It follows that $\boldsymbol{ap'}$ is singularly Sig-top-reducible and hence Sig-top-reducible. Since $\boldsymbol{ap'}$ has the same signature and leading term as $\boldsymbol{f}$, every Sig-top-reducer of $\boldsymbol{ap'}$ is also a Sig-top-reducer of $\boldsymbol{f}$ which makes $\boldsymbol{f}$ Sig-top-reducible, a contradiction. $\square$

3.2.2 First Signature basis algorithm

Theorem 3.2.2 is already enough to state a basic algorithm to compute a signature Gröbner basis:

Algorithm 10: Basic signature algorithm

Input: Input polynomials $F = \{f_1, ..., f_m\}$.

Output: Signature Gröbner basis $\boldsymbol{G}$ of $\langle F \rangle$, in particular $v(\boldsymbol{G})$ gives a Gröbner basis.

Set $\boldsymbol{G} = \{\boldsymbol{f_i}, i \in \{1, ..., m\}\}$;
Set $\boldsymbol{P} = \{Spol(\boldsymbol{f_i}, \boldsymbol{f_j}) = c_1 u \boldsymbol{f_i} - c_2 v \boldsymbol{f_j} : Sig(u\boldsymbol{f_i}) > Sig(v\boldsymbol{f_j})\}$;
while $P \neq \emptyset$ **do**
    Let $\boldsymbol{f}$ be an element of P with minimal signature;
    $\boldsymbol{P} = \boldsymbol{P} \backslash \{\boldsymbol{f}\}$;
    Compute $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},reg}$;
    if $\boldsymbol{f'}$ *is not singularly top-reducible,* $f' \neq 0$ **then**
        $\boldsymbol{P} = \boldsymbol{P} \cup \{Spol(\boldsymbol{f}, \boldsymbol{g}) = u\boldsymbol{f} - v\boldsymbol{g} : Sig(u\boldsymbol{f}) \neq Sig(v\boldsymbol{g}), \boldsymbol{g} \in \boldsymbol{G}\}$;
        $\boldsymbol{G} = \boldsymbol{G} \cup \{LC(f')^{-1}\boldsymbol{f'}\}$;//add the normalized element
    end
end
return $\boldsymbol{G}$;

Remark 3.2.8. *The computation of $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},reg}$ can be done by a slightly changed variant of Algorithm 1, checking the signature condition additionally.*

Proposition 3.2.9. *All elements added to $\boldsymbol{G}$ during Algorithm 10 (except the initial polynomials $\boldsymbol{f_i}$) are added in increasing signature.*

Proof. If a module element $\boldsymbol{f'}$ is added to $\boldsymbol{G}$, no S-polynomial in current P has a lower signature. Note that all regular S-polynomials created by a module element $\boldsymbol{f} \in \boldsymbol{G}$ with $Sig(\boldsymbol{f}) > \boldsymbol{s}$ have a larger signature than $\boldsymbol{s}$. Thus, after adding $\boldsymbol{f'}$ to $\boldsymbol{G}$, all elements in P will have a larger or equal signature than $\boldsymbol{f'}$ throughout the rest of the algorithm. In particular, no element with a lower signature will be added to $\boldsymbol{G}$. $\square$

3.3 Improvements

3.3.1 Syzygy criterion

Up to now, it does not seem too useful to compute the signature Gröbner basis because due to restricted reduction steps, this basis can be larger, and carrying the signatures might cause significant computational overhead. In this section we will see that the additional structure of signatures leads to further criteria that decide whether we have to compute the Sig-reduction of certain S-polynomials. To start the discussion, consider the following lemma:

Lemma 3.3.1. *Let $\boldsymbol{f} \in P^m$ and let $\boldsymbol{G}$ be a signature Gröbner basis up to $Sig(\boldsymbol{f})$. Assume there exists a syzygy $\boldsymbol{h} \in P^m$ such that $Sig(\boldsymbol{h}) \mid Sig(\boldsymbol{f})$. Then $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$.*

Proof. (compare [8, Lemma 6.4]) Observe that $\boldsymbol{h}' := \frac{Sig(\boldsymbol{f})}{Sig(\boldsymbol{h})}\boldsymbol{h}$ is a syzygy. Since we have $Sig(\boldsymbol{f} - \boldsymbol{h}') < Sig(\boldsymbol{f})$ and $\boldsymbol{G}$ is a signature Gröbner basis up to $\boldsymbol{f}$,

$$\boldsymbol{f} - \boldsymbol{h}' \xrightarrow[\boldsymbol{G}]{*} 0$$

follows. As $f - h' = f$ and $Sig(\boldsymbol{f} - \boldsymbol{h}') < Sig(\boldsymbol{f})$, each Sig-reduction step of $\boldsymbol{f} - \boldsymbol{h}'$ is a Sig-reduction step on $\boldsymbol{f}$ as well. Hence, $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$ follows. □

Lemma 3.3.1 leads to the following notion:

Definition 3.3.2 (Known syzygy). *Let $\boldsymbol{s} \in T_m$ and assume that we have already found a syzygy $\boldsymbol{f}$ with $Sig(\boldsymbol{f}) \sim \boldsymbol{s}$. Then we call $\boldsymbol{s}$ and, justified by Lemma 3.3.1, all monomial multiples of $\boldsymbol{s}$ **known syzygy**.*

As a next step, we need to find such syzygy signatures to apply the criterion from Lemma 3.3.1 as often as possible. Obviously, we can find syzygy signatures during the execution of the algorithm: If some S-polynomial $\boldsymbol{p}$ is regularly reduced to a syzygy $\boldsymbol{h}$, we can add $Sig(\boldsymbol{p}) = Sig(\boldsymbol{h})$ to the set of syzygies, which will be denoted in the further discussion by $\boldsymbol{H}$. Comparing for each S-polynomial during the further execution if the criterion from Lemma 3.3.1 holds for a signature in $\boldsymbol{H}$, we can discard S-polynomials meeting that criterion. In that way, we would start the algorithm with an empty set $\boldsymbol{H}$ of syzygy signatures. We can improve this approach further since we already know some syzygies at the start of the algorithm:

Definition 3.3.3 (Principal syzygies). *Recall from Definition 2.3.14 that $Syz(F)$ denotes the set of all syzygies on F and note that $Syz(F)$ is a submodule of P^m. Define*

$$\boldsymbol{s_{i,j}} := f_i \boldsymbol{f_j} - f_j \boldsymbol{f_i}$$

*for $1 \leq i < j \leq m$. Those $\boldsymbol{s_{i,j}} \in Syz(F)$ are called **principal syzygies**. We denote by $PSyz(F)$ the submodule of $Syz(F)$ generated by the principal syzygies.*

Observe that the signatures of those principal syzygies can already be added to $\boldsymbol{H}$ at the start of the algorithm.

3.3.2 Labelled polynomial optimization

A big issue of Algorithm 10 is that saving the module element takes a lot of memory. Luckily, we do not need the whole information stored in the module element $\boldsymbol{f}$, but only its signature $Sig(\boldsymbol{f})$ and its evaluation $v(\boldsymbol{f}) = f$. One can see that by the following observation:

- We only perform regular reductions on $\boldsymbol{f}$, hence its signature remains unchanged during the whole Sig-reduction process.
- We only consider regular S-pairs since singular ones can be discarded instantly.

Furthermore, we can drop the coefficient in the signature without any problems. Hence, the following structure replaces the module element in implemented algorithms:

Definition 3.3.4 (Labelled polynomial). *We call a pair $\mathcal{F} = (\boldsymbol{s}, f)$ with $\boldsymbol{s} \in T^m, f \in P$ a labelled polynomial if there exists a $\boldsymbol{f} \in P^m$:*

(i) $v(\boldsymbol{f}) = f$.

(ii) $MLT(\boldsymbol{f}) = \boldsymbol{s}$.

We define $poly(\mathcal{F}) = f$ and $Sig(\mathcal{F}) = \boldsymbol{s}$.

Remark 3.3.5.

1. *Note that this definition is named and defined quite differently in the literature. We decided to take over the term introduced by [23] which is used in modern research articles by many authors. In [12], this construction is called "Rule", in [10] and [8] "Sig-poly-pair". A labelled polynomial is often defined without properties (i) and (ii) from Definition 3.3.4 and when those properties are fulfilled, it is called admissible.*

2. *During the remaining part of this thesis, we will denote labelled polynomials by calligraphic letters. To shorten the notation, we introduce the following convention: If $\mathcal{F}$ is a labelled polynomial, then $\boldsymbol{f} \in P^m$ is the corresponding module element fulfilling (i) and (ii) from Definition 3.3.4.*

We will use the same notations for labelled polynomials as for module elements. One can always consider $\boldsymbol{f}$ instead of $\mathcal{F}$ to obtain the actual meaning, e.g.

$$Spol(\mathcal{F}, \mathcal{G}) := (MLT(Spol(\boldsymbol{f}, \boldsymbol{g})), v(Spol(\boldsymbol{f}, \boldsymbol{g}))),$$

$$\overline{\mathcal{F}}^{G,reg} := \{(MLT(\boldsymbol{f'}), v(\boldsymbol{f'})) : \boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},reg})\}.$$

We call a set $G = \{\mathcal{G}_1, ..., \mathcal{G}_k\}$ a signature Gröbner basis if $\{\boldsymbol{g_1}, ..., \boldsymbol{g_k}\}$ is such a basis.

3.3.3 Improved algorithm

With the improvements considered in sections 3.3.1 and 3.3.2, we obtain an improved version of Algorithm 10:

Algorithm 11: Improved signature algorithm

Input: Input polynomials $F = (f_1, ..., f_m)$, a term order $<$ and a compatible extension $<$ on T_m.

Output: Signature Gröbner basis G of $\langle F \rangle$, in particular $poly(G)$ gives a Gröbner basis.

Set $G = \{\mathcal{F}_i, i \in \{1, ..., m\}\}$;
Set $P = \{Spol(\mathcal{F}_i, \mathcal{F}_j) = c_1 u \mathcal{F}_i - c_2 v \mathcal{F}_j : Sig(u\mathcal{F}_i) > Sig(v\mathcal{F}_j), i, j \in \{1, ..., m\}\}$;
Set $\boldsymbol{H} = \{Sig(LT(f_i)\boldsymbol{f_j} - LT(f_j)\boldsymbol{f_i}) : i \neq j \in \{1, ..., m\}\}$;
while $P \neq \emptyset$ **do**
    Let $\boldsymbol{f}$ be an element of P with minimal signature;
    $P = P \backslash \{\mathcal{F}\}$;
    if *$Sig(\mathcal{F})$ is not divided by some signature in $\boldsymbol{H}$* **then**
        Compute some $\mathcal{F}' \in \overline{\mathcal{F}}^{G,reg}$;
        if $f' = 0$ **then**
            $\boldsymbol{H} = \boldsymbol{H} \cup \{Sig(\mathcal{F}')\}$;
        end
        else if *$\mathcal{F}'$is not singularly top-reducible* **then**
            $P = P \cup \{Spol(\mathcal{F}', \mathcal{G}) = c_1 u \mathcal{F}' - c_2 v \mathcal{G} : Sig(u\mathcal{F}') \neq Sig(v\mathcal{G}), \mathcal{G} \in G\}$;
            $G = G \cup \{LC(f')^{-1}\mathcal{F}'\}$;//add the normalized element
        end
    end
end
return G;

In the next couple of statements, we will show that this algorithm (and also Algorithm 10) are output-optimal in a certain sense, namely that they return a smallest signature Gröbner basis. To see this, we need to define certain types of signature Gröbner bases, similar to the ones of Definition 2.2.2:

Definition 3.3.6 (Reduced/minimal signature Gröbner basis). *Let $\boldsymbol{G} \subseteq P^m$ be a signature Gröbner basis.*

*(i) $\boldsymbol{G}$ is called a **reduced signature Gröbner basis** if*

$$\forall\, \boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G} : \text{ If } Sig(\boldsymbol{f}) \sim Sig(\boldsymbol{g}), \text{ then } \boldsymbol{f} = \boldsymbol{g}.$$

*(ii) $\boldsymbol{G}$ is called a **minimal signature Gröbner basis** if all $\boldsymbol{f} \in \boldsymbol{G}$ are Sig-top-irreducible with respect to $\boldsymbol{G} \backslash \{\boldsymbol{g}\}$.*

We will show some properties of minimal signature Gröbner bases first and prove that Algorithm 10 and Algorithm 11 return minimal signature Gröbner bases afterwards:

Proposition 3.3.7. *A minimal signature Gröbner basis is a reduced signature Gröbner basis.*

Proof. Let $\boldsymbol{G}$ be a minimal signature Gröbner basis and assume there exist $\boldsymbol{f} \neq \boldsymbol{g} \in \boldsymbol{G}, Sig(\boldsymbol{f}) \sim Sig(\boldsymbol{g})$. Since $\boldsymbol{G}$ is minimal we know that $\boldsymbol{f}, \boldsymbol{g}$ are regularly Sig-top-irreducible. Using Lemma 3.2.7, we obtain $LT(f) = LT(g)$, which implies that $\boldsymbol{f}$ singularly Sig-top-reduces $\boldsymbol{g}$ (and vice-versa), a contradiction. □

Definition 3.3.8. *Let $\boldsymbol{G}, \boldsymbol{G'} \subseteq P^m \backslash \{0\}$. We define ...*

(i) ... $Sig(\boldsymbol{G}) := \{Sig(\boldsymbol{g}) : \boldsymbol{g} \in \boldsymbol{G}\}$.

(ii) ... $Sig(\boldsymbol{G}) \sim Sig(\boldsymbol{G'}) :\Leftrightarrow \begin{cases} \forall \boldsymbol{g} \in \boldsymbol{G}\, \exists\, \boldsymbol{g'} \in \boldsymbol{G'} : Sig(\boldsymbol{g}) \sim Sig(\boldsymbol{g'}) \\ \forall \boldsymbol{g'} \in \boldsymbol{G'}\, \exists\, \boldsymbol{g} \in \boldsymbol{G} : Sig(\boldsymbol{g'}) \sim Sig(\boldsymbol{g}). \end{cases}$

(iii) ... $\boldsymbol{s} \notin Sig(\boldsymbol{G}) :\Leftrightarrow \forall \boldsymbol{g} \in \boldsymbol{G} : \boldsymbol{s} \nsim \boldsymbol{g}$.

The following theorem shows that all minimal signature Gröbner bases contain the same signatures and leading terms:

Theorem 3.3.9. *(compare [10, Theorem 5], stated but not proven there) Let $\boldsymbol{G}, \boldsymbol{G'}$ be minimal signature Gröbner bases. Then $Sig(\boldsymbol{G}) \sim Sig(\boldsymbol{G'})$, $LT(G) = LT(G')$. In particular, $|\boldsymbol{G}| = |\boldsymbol{G'}|$.*

Proof. We prove an even stronger result: For all $\boldsymbol{g} \in \boldsymbol{G}$ there exists a $\boldsymbol{g}' \in \boldsymbol{G'}$:

$$Sig(\boldsymbol{g}) \sim Sig(\boldsymbol{g}'), \quad LT(g) = LT(g').$$

For that, we first show that it suffices to prove that $Sig(\boldsymbol{G}) \sim Sig(\boldsymbol{G'})$: Assume that $Sig(\boldsymbol{G}) \sim Sig(\boldsymbol{G'})$, but there exist $\boldsymbol{g} \in \boldsymbol{G}, \boldsymbol{g}' \in \boldsymbol{G'}$ such that

$$\boldsymbol{s} := Sig(\boldsymbol{g}) \sim Sig(\boldsymbol{g'}), \quad LT(g) \neq LT(g').$$

Let $\boldsymbol{s}$ be the minimal signature with that property and define

$$\boldsymbol{G_s} := \{\boldsymbol{h} \in \boldsymbol{G} : Sig(\boldsymbol{h}) < \boldsymbol{s}\}.$$

Then for all $\boldsymbol{h'} \in \boldsymbol{G'_s}$ there exists $\boldsymbol{h} \in \boldsymbol{G_s}$:

$$Sig(\boldsymbol{h}) \sim Sig(\boldsymbol{h}'), \quad LT(h) = LT(h').$$

Note that $\boldsymbol{g}$ is Sig-top-irreducible with respect to $\boldsymbol{G}$ by minimality of $\boldsymbol{G}$, thus $\boldsymbol{g}$ is Sig-top-irreducible by $\boldsymbol{G_s}$. It follows that $\boldsymbol{g}$ is also Sig-top-irreducible by $\boldsymbol{G'_s}$ and further by $\boldsymbol{G'}$. As $\boldsymbol{g}$ and $\boldsymbol{g'}$ have the same signature and are both Sig-top-irreducible with respect to $\boldsymbol{G}'$, applying Lemma 3.2.7 yields $LT(g) = LT(g')$, a contradiction.

Now we are left to show that $Sig(\boldsymbol{G}) \sim Sig(\boldsymbol{G'})$: Let $\boldsymbol{s}$ be the minimal signature such that $Sig(\boldsymbol{G_s}) \nsim Sig(\boldsymbol{G'_s})$. Then there exists, without loss of generality, $\boldsymbol{g} \in \boldsymbol{G} : Sig(\boldsymbol{g}) = \boldsymbol{s} \notin Sig(\boldsymbol{G'})$. By the same argument as used before, we obtain $LT(\boldsymbol{G_s}) = LT(\boldsymbol{G'_s})$. Since $\boldsymbol{s} \notin Sig(\boldsymbol{G'})$, $\boldsymbol{G'_s}$ is a signature Gröbner basis in $\boldsymbol{s}$ and hence, there exists $\boldsymbol{g'} \in \boldsymbol{G'_s} : LT(g') \mid LT(g)$. But this contradicts the minimality of $\boldsymbol{G}$ since it follows that $\boldsymbol{g}$ is Sig-top-reducible by $\boldsymbol{G'_s}$ and hence by $\boldsymbol{G}\backslash\{\boldsymbol{g}\}$.
To show the "in particular" part, note that by Proposition 3.3.7, $\boldsymbol{G}$ and $\boldsymbol{G'}$ are reduced and therefore,

$$|\boldsymbol{G}| = |Sig(\boldsymbol{G})| = |Sig(\boldsymbol{G'})| = |\boldsymbol{G'}|.$$

□

Lemma 3.3.10. *(stated in [10, p.4], but not proven there)*

(i) Algorithm 10 and Algorithm 11 compute a reduced signature Gröbner basis.

(ii) Algorithm 10 and Algorithm 11 compute a minimal signature Gröbner basis.

Proof.

(i): Assume to the contrary that there exist $\boldsymbol{f} \neq \boldsymbol{g} \in \boldsymbol{G}$ such that $Sig(\boldsymbol{f}) \sim Sig(\boldsymbol{g})$ where, without loss of generality, $\boldsymbol{f}$ was added after $\boldsymbol{g}$. By Proposition 3.2.9 elements are added in increasing signatures, hence, both $\boldsymbol{f}, \boldsymbol{g}$ are still regularly Sig-irreducible at the step where $\boldsymbol{f}$ is computed. This implies that $LT(f) = LT(g)$ by Lemma 3.2.7 and thus, $\boldsymbol{g}$ singularly Sig-top-reduces $\boldsymbol{f}$. Therefore, $\boldsymbol{f}$ will not be added to $\boldsymbol{G}$, a contradiction.

(ii): Assume to the contrary that there exists $\boldsymbol{f} \neq \boldsymbol{g} \in \boldsymbol{G}$ such that $\boldsymbol{g}$ Sig-reduces $\boldsymbol{f}$. By (i), it follows that $Sig(\boldsymbol{g}) < Sig(\boldsymbol{f})$ and hence $\boldsymbol{g}$ is added before $\boldsymbol{f}$. By construction, $\boldsymbol{f}$ is regularly Sig-irreducible at the time it is added to $\boldsymbol{G}$, so it cannot be regularly Sig-reduced by $\boldsymbol{g}$ either. Therefore, $\boldsymbol{g}$ singularly Sig-reduces $\boldsymbol{f}$. But then $\boldsymbol{f}$ would not have been added to $\boldsymbol{G}$, a contradiction.

□

3.4 Rewrite bases

In this section we will mainly follow the ideas of [10] and make use of some aspects considered in [8].

3.4.1 The idea

We already obtained an output-optimal algorithm which discards many S-polynomials in advance. Our target here is to strengthen the criteria of discarding unnecessary S-polynomials as early as possible and to decrease the number of reduction steps taken if a reduction is unavoidable. Note that Lemma 3.2.7 helps us for that purpose: If $Sig(\boldsymbol{f}) \sim Sig(\boldsymbol{g})$, their regular

Sig-reductions with respect to a signature Gröbner basis up to $Sig(\boldsymbol{f})$ coincide. Hence, it suffices to compute exactly one of those. To be precise, even more can be said: Instead of computing the Sig-reduction of an S-polynomial $\boldsymbol{p}$, we can choose any module element $\boldsymbol{h}$ to compute the regular Sig-reduction of $\boldsymbol{h}$ as long as $Sig(\boldsymbol{p}) = Sig(\boldsymbol{h})$. In particular, if $\boldsymbol{p}$ is a minimal S-polynomial left to examine with signature $c\boldsymbol{s}, c \in \mathbb{K}\setminus\{0\}$, we can take any element of the set

$$\{v\boldsymbol{g} : v \in T, \boldsymbol{g} \in \boldsymbol{G}, Sig(v\boldsymbol{g}) \sim \boldsymbol{s}\}$$

and regularly Sig-reduce this one instead. Together with the set $\boldsymbol{H}$ of known syzygies, we can choose any element of

$$\boldsymbol{C_s} := \{v\boldsymbol{g} : v \in T, \boldsymbol{g} \in \boldsymbol{G} \cup \boldsymbol{H}, Sig(v\boldsymbol{g}) \sim \boldsymbol{s}\}.$$

If $\boldsymbol{H} \cap \boldsymbol{C_s} \neq \emptyset$, we can discard all S-polynomials with this signature straight away by 3.3.1. Otherwise, we try to find an element of $\boldsymbol{C_s}$ which tends to have few Sig-reduction steps left to compute and is rather fast to find. The idea is to define a partial order $<$ on $\boldsymbol{G} \cup \boldsymbol{H}$ such that all elements in $\boldsymbol{G}$ are totally ordered and $\boldsymbol{f} \leq \boldsymbol{h}$ whenever $\boldsymbol{h} \in \boldsymbol{H}$. By picking a maximal element with respect to this order, we discard all S-polynomials of this signature when the signature is divisible by some known syzygy. Otherwise, we choose the (unique) maximal element of $\boldsymbol{G} \cap \boldsymbol{C_s}$ with respect to the chosen order.

Remark 3.4.1. *Note that the examined* $\boldsymbol{C_s}$ *are always nonempty: We only consider regular S-polynomials* $\boldsymbol{p} = u\boldsymbol{f} - v\boldsymbol{g}$ *with* $Sig(\boldsymbol{p}) \sim \boldsymbol{s}$*. Since, without loss of generality,* $Sig(\boldsymbol{p}) = Sig(u\boldsymbol{f})$*,* $u\boldsymbol{f} \in \boldsymbol{C_s}$ *follows.*

We are left to define the total order on $\boldsymbol{G}$. It is useful for the further discussion that this order fulfills the following property:

Definition 3.4.2.

(i) Let $<$ *be a total order on* $\boldsymbol{G}$*, fulfilling*

$$\forall \boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G} : Sig(\boldsymbol{f}) \,|\, Sig(\boldsymbol{g}) \Rightarrow \boldsymbol{f} \leq \boldsymbol{g}.$$

Then $<$ *is a* ***rewrite order*** *on* $\boldsymbol{G}$*.*

(ii) Let $\boldsymbol{s}$ *be a signature,* $<$ *a rewrite order on* $\boldsymbol{G}, u \in T$*. Then* $u\boldsymbol{f}$ *is called the* ***canonical rewriter*** *of* $\boldsymbol{s}$ *if* $\boldsymbol{f}$ *is the maximal element of* $\boldsymbol{C_s}$ *and* $Sig(u\boldsymbol{f}) \sim \boldsymbol{s}$*.*

Proposition 3.4.3. *Let* $<$ *be a rewrite order on* $\boldsymbol{G}$*,* $\boldsymbol{f} \in \boldsymbol{G}$ *and* $\boldsymbol{s} := Sig(\boldsymbol{f})$ *not a known syzygy signature. Then* $\boldsymbol{f}$ *is the canonical rewriter of* $\boldsymbol{s}$*.*

Proof. Let $u \in T, \boldsymbol{g} \in \boldsymbol{G} : Sig(u\boldsymbol{g}) \sim Sig(\boldsymbol{f})$. Then $Sig(\boldsymbol{g}) \,|\, Sig(\boldsymbol{f})$ implies by definition that $\boldsymbol{g} \leq \boldsymbol{f}$. Since $\boldsymbol{s}$ is not a syzygy signature, the statement follows. □

To formalize the discussion above, we introduce a new type of basis, namely the so-called rewrite basis:

Definition 3.4.4 (Rewrite basis). *Let $\boldsymbol{s} \in T_m$, $\boldsymbol{G} \subseteq P^m \backslash \{0\}$, $<$ a rewrite order on $\boldsymbol{G}$.*

(i) We call $\boldsymbol{G}$ a ***rewrite basis in*** $\boldsymbol{s}$ *if the canonical rewriter of $\boldsymbol{s}$ is regularly Sig-top-irreducible or $\boldsymbol{s}$ is a syzygy signature.*

(ii) We call $\boldsymbol{G}$ a ***rewrite basis up to*** $\boldsymbol{s}$ *if for all $\boldsymbol{t} \in T_m : \boldsymbol{t} < \boldsymbol{s}$, $\boldsymbol{G}$ is a rewrite basis in $\boldsymbol{t}$.*

Theorem 3.4.5. *Let $\boldsymbol{s} \in T_m$, $\boldsymbol{G} \subseteq P^m \backslash \{0\}$. If $\boldsymbol{G}$ is a rewrite basis up to $\boldsymbol{s}$, $\boldsymbol{G}$ is a signature Gröbner basis up to $\boldsymbol{s}$.*

Proof. (compare [10, Lemma 8]) Assume to the contrary that $\boldsymbol{G}$ is not a signature Gröbner basis. Then there exists $\boldsymbol{f} \in P^m$ which is not Sig-reducible to 0 and $Sig(\boldsymbol{f}) \sim \boldsymbol{t} < \boldsymbol{s}$. Taking $\boldsymbol{f}$ with minimal signature, $\boldsymbol{G}$ is a signature Gröbner basis and a rewrite basis up to $\boldsymbol{t}$. If $\boldsymbol{t}$ is a known syzygy signature, then $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$ follows directly from Lemma 3.3.1, a contradiction. Thus, $\boldsymbol{t}$ is not a known syzygy signature. Let $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{G,reg}$ and let $v\boldsymbol{g}$ with $v \in T, \boldsymbol{g} \in \boldsymbol{G}$ be the canonical rewriter of $\boldsymbol{t}$. Since $\boldsymbol{G}$ is a rewrite basis in $\boldsymbol{t}$, $v\boldsymbol{g}$ is regularly Sig-top-irreducible and so is $\boldsymbol{f'}$ by construction. Note that

$$Sig(\boldsymbol{f'}) = Sig(\boldsymbol{f}) \sim Sig(v\boldsymbol{g})$$

and hence, applying Lemma 3.2.7 yields $LT(vg) = LT(f')$. Therefore, we see that for suitable $c \in \mathbb{K}$ that $cv\boldsymbol{g}$ singularly Sig-top-reduces $\boldsymbol{f'}$. which implies $Sig(\boldsymbol{f'} - cv\boldsymbol{g}) < \boldsymbol{t}$. Since $\boldsymbol{G}$ is a signature Gröbner basis up to $\boldsymbol{t}$, $\boldsymbol{f'} - cv\boldsymbol{g} \xrightarrow[\boldsymbol{G}]{*} 0$ follows. Therefore, $\boldsymbol{f} \xrightarrow[\boldsymbol{G}]{*} 0$, a contradiction. $\square$

Theorem 3.4.5 shows us that finding a correct rewrite basis algorithm leads to a signature Gröbner basis and therefore, to a Gröbner basis. The following statements show us that examining certain S-polynomials is enough to create such a rewrite basis.

Lemma 3.4.6. *Let $\boldsymbol{G}$ be a rewrite basis up to signature $\boldsymbol{s}$ and $\boldsymbol{s}$ no known syzygy signature. Furthermore, let $u\boldsymbol{f}, u \in T, \boldsymbol{f} \in \boldsymbol{G}$ be the canonical rewriter of $\boldsymbol{s}$ and assume there exists a regular top-reducer $\boldsymbol{g}$ of $u\boldsymbol{f}$ such that $LM(cvg) = LM(uf), c \in \mathbb{K}\backslash\{0\}, v \in T$. Then*

$$Spol(\boldsymbol{f}, \boldsymbol{g}) = u\boldsymbol{f} - cv\boldsymbol{g}, \quad Sig(u\boldsymbol{f} - cv\boldsymbol{g}) \sim \boldsymbol{s}.$$

In particular: If $\boldsymbol{G}$ is a rewrite basis up to $\boldsymbol{s}$, but not in $\boldsymbol{s}$, there exists a regular S-polynomial $\boldsymbol{p}$ such that $Sig(\boldsymbol{p}) \sim \boldsymbol{s}$.

Proof. (compare [10, Lemma 9]) Since $LM(cvg) = LM(uf)$, it suffices by Proposition 3.2.3 (iii) to show that $t := \gcd(u, v) = 1$. Assume to the contrary that $t > 1$. Defining $\boldsymbol{s} = r\boldsymbol{f_i}$, observe that $t \,|\, r$ since $u \,|\, r$ and $v \,|\, r$. Since $t > 1$, we have that $\boldsymbol{G}$ is a rewrite basis in $\frac{r}{t}\boldsymbol{f_i}$.

Therefore, the canonical rewriter of $\frac{r}{t}\boldsymbol{f_i}$, denoted by $w\boldsymbol{h}$, is regularly Sig-top-irreducible. Note that

$$Sig(\frac{u}{t}\boldsymbol{f}) = \frac{s}{t}\boldsymbol{f_i} \Rightarrow \boldsymbol{f} \leq \boldsymbol{h}$$
$$Sig(tw\boldsymbol{h}) = \boldsymbol{s} \Rightarrow \boldsymbol{h} \leq \boldsymbol{f}.$$

Since $\boldsymbol{G}$ is totally ordered, it follows that $\boldsymbol{f} = \boldsymbol{h}$ and hence $w = \frac{u}{t}$. But $\frac{v}{t}\boldsymbol{g}$ regularly Sig-top-reduces $\frac{u}{t}\boldsymbol{f} = w\boldsymbol{h}$, a contradiction to $w\boldsymbol{h}$ being regularly Sig-top-irreducible.
The in particular part follows immediately since the canonical rewriter of $\boldsymbol{s}$ is by definition regularly Sig-top-reducible by some element in $C_{\boldsymbol{s}}$. □

Corollary 3.4.7. *(compare [10, Lemma 10])* $\boldsymbol{G}$ *is a rewrite basis up to signature* $\boldsymbol{t}$ *if and only if* $\boldsymbol{G}$ *is a rewrite basis in all signatures* $\boldsymbol{s} < \boldsymbol{t}$ *where* $\boldsymbol{s} = \boldsymbol{f_i}$ *or there exists a regular S-polynomial* $\boldsymbol{p} = Spol(\boldsymbol{f}, \boldsymbol{g})$ *with* $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}$ *such that* $Sig(\boldsymbol{p}) \sim \boldsymbol{s}$.

Proof. The "only if" is obvious by the definition of a rewrite basis. So we are left to show the "if"-direction: Let $\boldsymbol{G}$ be a rewrite basis in all signatures $\boldsymbol{s} < \boldsymbol{t}$ fulfilling the property defined above. Assume to the contrary that $\boldsymbol{G}$ is not a rewrite basis up to $\boldsymbol{t}$. Then there exists a minimal signature $\boldsymbol{t'}$ such that $\boldsymbol{G}$ is a rewrite basis up to $\boldsymbol{t'}$, but not in $\boldsymbol{t'}$. By the "in particular" part of Lemma 3.4.6, there exists an S-polynomial $\boldsymbol{p}$ with $Sig(\boldsymbol{p}) \sim \boldsymbol{t'}$, a contradiction. □

Fortunately, we can use the rewriting argument even for the case when $Spol(\boldsymbol{f}, \boldsymbol{g}) = u\boldsymbol{f} - v\boldsymbol{g}$ with $Sig(u\boldsymbol{f}) > Sig(v\boldsymbol{g})$ when $v\boldsymbol{g}$ is not the canonical rewriter:

Lemma 3.4.8. *Let* $\boldsymbol{G}$ *be a rewrite basis up to* $Sig(\boldsymbol{f})$ *and let* $t \in T(f)$ *be regularly reducible. Then there exist* $\boldsymbol{c'} \in \mathbb{K}\backslash\{0\}, w \in T, \boldsymbol{h} \in \boldsymbol{G}$ *such that* $c'w\boldsymbol{h}$ *regularly reduces* t*, fulfilling the following properties:*

(i) $w\boldsymbol{h}$ *is regularly Sig-top-irreducible.*

(ii) $Sig(w\boldsymbol{h})$ *is not a syzygy signature.*

(iii) $w\boldsymbol{h}$ *is the canonical rewriter of* $Sig(w\boldsymbol{h})$*.*

Proof. (compare [10, Lemma 11]) Since t is regularly Sig-top-reducible, there exist regular top-reducers. Let $cv\boldsymbol{g}, \boldsymbol{c} \in \mathbb{K}\backslash\{0\}, v \in T, \boldsymbol{g} \in \boldsymbol{G}$ be such a top-reducer with minimal signature $\boldsymbol{s}$ among them. We will show that the canonical rewriter of $\boldsymbol{s}$, denoted by $w\boldsymbol{h}$, fulfills all these properties. At first, we show that $c'w\boldsymbol{h}$ for suitable $c' \in \mathbb{K}$ regularly Sig-reduces t: To do that, we prove that $v\boldsymbol{g}$ is regularly Sig-top-irreducible: Assuming the contrary, this implies there exists $v' \in T, \boldsymbol{g'} \in \boldsymbol{G}$ such that

$$LT(v'\boldsymbol{g'}) = LT(v\boldsymbol{g}), \quad Sig(v'\boldsymbol{g'}) < Sig(v\boldsymbol{g}),$$

contradicting the minimality of $v\boldsymbol{g}$. Hence, $v\boldsymbol{g}$ is regular top-irreducible. As $\boldsymbol{G}$ is a rewrite basis up to $\boldsymbol{t}$ and $Sig(w\boldsymbol{h}) \sim \boldsymbol{s} < \boldsymbol{t}$, $w\boldsymbol{h}$ is regularly Sig-top-irreducible. Therefore, we can apply Lemma 3.2.7 and obtain $LT(v\boldsymbol{g}) = LT(w\boldsymbol{h})$, showing that $w\boldsymbol{h}$ regularly Sig-reduces t. We are left to show the three stated properties for $w\boldsymbol{h}$:

(i): Already covered by the discussion above.

(ii): Assume to the contrary that there exists a syzygy $\boldsymbol{f'}$ such that $Sig(\boldsymbol{f'}) = \boldsymbol{s}$. Then

$$LT(wh - f') = LT(wh), \quad Sig(w\boldsymbol{h} - \boldsymbol{f'}) < \boldsymbol{s}.$$

By Theorem 3.4.5, $\boldsymbol{G}$ is a signature Gröbner basis up to $Sig(\boldsymbol{f})$. Thus, there exists a (possibly singular) Sig-top-reducer $\boldsymbol{g'} \in \boldsymbol{G}$ of $w\boldsymbol{h} - \boldsymbol{f}'$. But this implies that $\boldsymbol{g'}$ **regularly** Sig-top-reduces $w\boldsymbol{h}$, a contradiction to (i).

(iii): Follows immediately by the construction and (ii).

$\square$

Summing up all these statements, we obtain a theorem which can be stated in a rather compact form after defining the term "rewriteable":

Definition 3.4.9 (Rewriteable). *Let $\boldsymbol{f} \in P^m \setminus \{0\}, \boldsymbol{G} \subseteq P^m, \boldsymbol{H}$ a set of syzygy signatures. We call $\boldsymbol{f}$* ***rewriteable*** *if $Sig(\boldsymbol{f})$ is divisible by a signature of $\boldsymbol{H}$ or if $\boldsymbol{f}$ is not the canonical rewriter of $Sig(\boldsymbol{f})$ (with respect to $\boldsymbol{G}$).*

Theorem 3.4.10. *Let $\boldsymbol{s} \in T_m$, $\boldsymbol{G} \subseteq P^m$ a rewrite basis up to $\boldsymbol{s}$, but not in $\boldsymbol{s}$, $\boldsymbol{H}$ a set of syzygies. Then there exists a regular S-polynomial $Spol(\boldsymbol{f}, \boldsymbol{g}) = u\boldsymbol{f} - v\boldsymbol{g}$ with $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}$ such that the following conditions hold:*

(i) $v\boldsymbol{g}$ is not rewriteable.

(ii) $u\boldsymbol{f}$ is not rewriteable

(iii) $Sig(Spol(\boldsymbol{f}, \boldsymbol{g})) \sim \boldsymbol{s}$.

Proof. Let $u\boldsymbol{f}$ be the canonical rewriter of $\boldsymbol{s}$. Applying Lemma 3.4.8 with $t := LT(uf)$, we obtain a monomial v and some $\boldsymbol{g} \in \boldsymbol{G}$ such that $v\boldsymbol{g}$ is a regular top-reducer of $u\boldsymbol{f}$, the canonical rewriter of its signature which is not known to be syzygy. Thus, $v\boldsymbol{g}$ is not rewriteable. Since $\boldsymbol{G}$ is not a rewrite basis in $\boldsymbol{s}$, $u\boldsymbol{f}$ is not rewriteable as well. By Lemma 3.4.6, $Spol(\boldsymbol{f}, \boldsymbol{g}) = u\boldsymbol{f} - v\boldsymbol{g}$ and $Sig(Spol(\boldsymbol{f}, \boldsymbol{g})) \sim \boldsymbol{s}$. $\square$

3.4.2 Rewrite basis algorithm

Theorem 3.4.10 leads to an idea how to construct a rewrite basis algorithm. As in Algorithms 10 and 11, we can apply the labelled polynomial optimization from Section 3.3.2, since rewrite orders do not need the whole information from the module element. Note that this new algorithm, in comparison to Algorithms 10 and 11, does not check whether $\boldsymbol{f'}$ is singularly Sig-top-reducible. This is left out intentionally to ensure correctness. Furthermore, we arranged a small improvement in the following pseudocode as well: Instead of adding the initial polynomials $\boldsymbol{f_i}$ directly to $\boldsymbol{G}$, we add them to $\boldsymbol{P}$ in the start and treat them as they were S-polynomials. The algorithm will pick the polynomial with smallest signature first, adding the other initial polynomials $\boldsymbol{f_i}$ at the time no S-polynomial with a smaller signature is left. In that way, we might reduce those initial polynomials even further than we did by the described preprocessing in Remark 3.1.9. For that approach, we have to shift the construction of principal syzygies from initialization to the while-loop, adding syzygy signatures of principal syzygies $\boldsymbol{s}_{i,j}, j \in \{i+1, ..., m\}$ at the time the element $\boldsymbol{f_i}$ is considered.

Algorithm 12: Rewrite basis algorithm

Input: Input polynomials $F = \{f_1, ..., f_m\}$, a rule to define a rewrite order $\leq$ on all sets $\boldsymbol{G} \cup \boldsymbol{H}$ created by the algorithm.

Output: Rewrite basis $\boldsymbol{G}$ of $\langle F \rangle$, in particular $v(\boldsymbol{G})$ is a Gröbner basis.

Set $\boldsymbol{G} = \emptyset$;
Set $\boldsymbol{P} = \{\mathcal{F}_i, i \in \{1, ..., m\}\}$;
Set $\boldsymbol{H} = \emptyset$;
while $\boldsymbol{P} \neq \emptyset$ **do**
    Let $\mathcal{F} = c_1 u \mathcal{G}_1 - c_2 v \mathcal{G}_2$ be an element of $\boldsymbol{P}$ with minimal signature; $\boldsymbol{P} = \boldsymbol{P} \setminus \{\mathcal{F}\}$;
    if $u\mathcal{G}_1$ *and* $v\mathcal{G}_2$ *are not rewriteable* **then**
        $\mathcal{F}' \in \overline{\mathcal{F}}^{G,reg}$;//or $\mathcal{F}' \in \overline{u\mathcal{G}_1}^{G,reg}$
        if $f' = 0$ **then**
            $\boldsymbol{H} = \boldsymbol{H} \cup \{Sig(\mathcal{F}')\}$;
        end
        else
            $\boldsymbol{P} = \boldsymbol{P} \cup \{Spol(\mathcal{F}', \mathcal{G}) = u\mathcal{F}' - v\mathcal{G} : Sig(u\mathcal{F}') \neq Sig(v\mathcal{G}), \mathcal{G} \in \boldsymbol{G}\}$;
            $\boldsymbol{G} = \boldsymbol{G} \cup \{LC(f')^{-1}\mathcal{F}'\}$;
            if $Sig(\mathcal{F}') = \boldsymbol{f_i}$ **then**
                $\boldsymbol{H} = \boldsymbol{H} \cup \{Sig(g\mathcal{F}_i - f_i\mathcal{G}) : \mathcal{G} \in \boldsymbol{G}\}$;
            end
        end
    end
end
return G;

3.4.3 Proof of Correctness

Lemma 3.4.11. *Let $\boldsymbol{G}$ be a rewrite basis up to signature $\boldsymbol{t}$, computed during the algorithm. Furthermore let $u\boldsymbol{f}$ be the canonical rewriter of $\boldsymbol{t}$ where $\boldsymbol{t}$ is not a known syzygy signature. Then Algorithm 12 computes the Sig-reduction of an S-polynomial with signature $\boldsymbol{t}$ if and only if $u\boldsymbol{f}$ is regularly Sig-top-reducible.*

Proof. (compare [10, Lemma 12])

- "If": Follows directly from Theorem 3.4.10.
- "Only if": Assume that $u\boldsymbol{f}$ is not regularly *Sig*-top-reducible. Then there exists no regular $\boldsymbol{p} := Spol(\boldsymbol{f}, \boldsymbol{g})$ with $Sig(\boldsymbol{p}) \sim \boldsymbol{t}, \boldsymbol{g} \in \boldsymbol{G}$. If there is a regular S-polynomial $\boldsymbol{q} := Spol(\boldsymbol{f'}, \boldsymbol{g'})$ with $\boldsymbol{f'}, \boldsymbol{g'} \in \boldsymbol{G}, Sig(\boldsymbol{q}) = u'\boldsymbol{f'} \sim \boldsymbol{t}$, then $\boldsymbol{f'} < \boldsymbol{f}$ and hence, $u'\boldsymbol{f'}$ is rewriteable. Therefore, Algorithm 12 discards $\boldsymbol{q}$, and thus, no S-polynomial with signature $\boldsymbol{t}$ is computed.

□

Theorem 3.4.12. *Algorithm 12 is correct.*

Proof. (compare [10, Theorem 7],) Assume by contradiction that Algorithm 12 returns a set $\boldsymbol{G}$ which is not a rewrite basis in some signature $\boldsymbol{s}$ and let $\boldsymbol{s}$ be minimal with that property. By Lemma 3.4.7, there exists an S-polynomial $\boldsymbol{p}$ such that $Sig(\boldsymbol{p}) \sim \boldsymbol{s}$. Since $\boldsymbol{s}$ is not a known syzygy signature and the canonical rewriter $v\boldsymbol{g}$ of $\boldsymbol{s}$ is regularly *Sig*-top-reducible, we know by Lemma 3.4.11 that the regular reduction of some S-polynomial $\boldsymbol{q}$ with signature $\boldsymbol{s}$ was computed by the algorithm. By assumption, this $\boldsymbol{q'} \in \overline{\boldsymbol{q}}^{\boldsymbol{G},reg}$ is no syzygy (since $\boldsymbol{s}$ is no syzygy signature) and thus, was added to $\boldsymbol{G}$. It follows from Proposition 3.4.3 that $\boldsymbol{q'}$ is the canonical rewriter of $\boldsymbol{s}$. But $\boldsymbol{q'}$ is not regularly *Sig*-reducible, a contradiction. □

Remark 3.4.13.

1. *One can show that Algorithm 12 terminates (see [10, Theorem 20]). We skip the details about this proof for the sake of shortness. Note that termination in those algorithms is far from trivial, many termination "proofs" for signature basis algorithms (e.g. in [15]) turned out to be false.*
2. *A similarly tough question concerns the running time of those algorithms. As estimations for "normal" Gröbner basis algorithms, the running time highly depends on the input set. Generally, one ends up with doubly exponential bounds, but those bounds are often far too pessimistic compared with empirical results. For detailed information about Gröbner bases and complexities, see e.g. [21, Chapter 6].*

The following Lemma states a property of Algorithm 12 when $<_{pot}$ is taken as the order extension. Note that this means advantages as well as disadvantages, as discussed later.

Lemma 3.4.14. *(idea from [23], p.41) Let $\boldsymbol{G_i}$ be the set $\boldsymbol{G}$ at the time some initial polynomial $\boldsymbol{f_i}$ is examined in Algorithm 12 and let $<_{pot}$ be the order extension for an arbitrary term order $<$. Then $\boldsymbol{G_i}$ is a rewrite basis for $\langle f_{i+1}, ..., f_m \rangle$. In particular, $v(\boldsymbol{G_i})$ is a Gröbner basis for $\langle f_{i+1}, ..., f_m \rangle$.*

Proof. Since $<_{pot}$ was chosen as the order extension, the initial polynomials $\boldsymbol{f_{i+1}}, ..., \boldsymbol{f_m}$ have been examined before. Additionally, all S-polynomials created by those polynomials have been treated since they have a lower signature than $\boldsymbol{f_i}$. Applying Theorem 3.4.12, we get that $\boldsymbol{G_i}$ is a rewrite basis for $\langle f_{i+1}, ..., f_m \rangle$. The "in particular" part follows from Theorem 3.4.5 and Proposition 3.1.12. □

3.4.4 Choosing the best rewrite order

So far, we did not discuss any particular rewrite order. We will examine and discuss the properties of the most interesting ones in the next section.

Definition 3.4.15 (Interesting rewrite orders).

(i) Number-order $<_{num}$ (see [26]): This order can only be defined via Algorithm 12, equipping each element in $\boldsymbol{G}$ with a unique number in the following way: When some $\boldsymbol{f}$ is added to $\boldsymbol{G}$, we set $Num(\boldsymbol{f}) := |\boldsymbol{G}|$. In that way, we can define

$$\boldsymbol{f} <_{num} \boldsymbol{g} :\Leftrightarrow Num(\boldsymbol{f}) < Num(\boldsymbol{g}).$$

(ii) Ratio-order $<_{rat}$ (see [10]):

$$\boldsymbol{f} <_{rat} \boldsymbol{g} :\Leftrightarrow \begin{cases} Sig(\boldsymbol{f})LT(g) < Sig(\boldsymbol{g})LT(f) \ \text{or} \\ Sig(\boldsymbol{f})LT(g) = Sig(\boldsymbol{g})LT(f),\, Sig(\boldsymbol{f}) < Sig(\boldsymbol{g}). \end{cases}$$

*(iii) F5-order $<_{F5}$ (see [10]): We define the **index** of a module element $\boldsymbol{f}$ with $Sig(\boldsymbol{f}) = ct\boldsymbol{f_i}$, $c \in \mathbb{K}\backslash\{0\}$, $t \in T$ as $\mathrm{index}\,(\boldsymbol{f}) := i$. For*

$$Sig(\boldsymbol{f}) = ct\boldsymbol{f_i}, \quad Sig(\boldsymbol{g}) = c'u\boldsymbol{g_j},$$

we define

$$\boldsymbol{f} <_{F5} \boldsymbol{g} :\Leftrightarrow \begin{cases} \mathrm{index}(\boldsymbol{f}) > \mathrm{index}(\boldsymbol{g}) \ \text{or} \\ \mathrm{index}(\boldsymbol{f}) = \mathrm{index}(\boldsymbol{g}),\, deg(t) \leq deg(u) \ \text{or} \\ \mathrm{index}(\boldsymbol{f}) = \mathrm{index}(\boldsymbol{g}),\, deg(t) = deg(u),\, \text{arbitrary rule.} \end{cases}$$

Remark 3.4.16. *For shorter notation, we denote the **Sig-lead-ratio** of $\boldsymbol{f} \in \boldsymbol{G}$ by*

$$r_{\boldsymbol{f}} := \frac{Sig(\boldsymbol{f})}{LT(f)}.$$

Using the default way of defining orders on fractions, we set

$$r_{\boldsymbol{f}} < r_{\boldsymbol{g}} :\Leftrightarrow Sig(\boldsymbol{f})LT(g) < Sig(\boldsymbol{g})LT(f).$$

In that way, we can state the ratio order more compactly:

$$\boldsymbol{f} <_{rat} \boldsymbol{g} \Leftrightarrow \begin{cases} r_{\boldsymbol{f}} < r_{\boldsymbol{g}} \text{ or} \\ r_{\boldsymbol{f}} = r_{\boldsymbol{g}},\ Sig(\boldsymbol{f}) < Sig(\boldsymbol{g}). \end{cases}$$

This explains why we call this "ratio order".

Proposition 3.4.17. *Let $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}$, $Sig(u\boldsymbol{f}) \sim Sig(v\boldsymbol{g}), v\boldsymbol{g}$ regularly Sig-top-irreducible. Then $r_{\boldsymbol{f}} \leq r_{\boldsymbol{g}}$.*

Proof. (compare [10, Lemma 14]) For $\boldsymbol{f'} \in \overline{u\boldsymbol{f}}^{\boldsymbol{G},reg}$, we have $LT(uf) \geq LT(f')$. Since $Sig(\boldsymbol{f'}) \sim Sig(v\boldsymbol{g})$ and both elements are regularly Sig-top-irreducible, applying Lemma 3.2.7 yields $LT(vg) = LT(f')$. Since $Sig(u\boldsymbol{f}) \sim Sig(v\boldsymbol{g})$, $r_{\boldsymbol{f}} \leq r_{\boldsymbol{g}}$ follows. □

Lemma 3.4.18. *Let $\boldsymbol{G} \subseteq P^m \backslash \{0\}$ consist of elements with pairwise distinct signatures. Then all three defined orders are rewrite orders.*

Proof. By definition and the property of distinct signatures in $\boldsymbol{G}$, all three orders totally order $\boldsymbol{G}$, hence, it suffices to show that

$$\forall \boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G} : Sig(\boldsymbol{f}) \,|\, Sig(\boldsymbol{g}) \Rightarrow \boldsymbol{f} \leq \boldsymbol{g}. \tag{3.4}$$

(i) $<_{num}$: Let $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}$. Since elements in Algorithm 12 are added in increasing signature to $\boldsymbol{G}$, we have

$$Sig(\boldsymbol{f}) \leq Sig(\boldsymbol{g}) \Leftrightarrow Num(\boldsymbol{f}) < Num(\boldsymbol{g}).$$

As $Sig(\boldsymbol{f}) \,|\, Sig(\boldsymbol{g})$ implies $Sig(\boldsymbol{f}) \leq Sig(\boldsymbol{g})$, (3.4) follows.

(ii) $<_{rat}$ (compare [10, Theorem 13]): Let $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}$, $Sig(\boldsymbol{f}) \,|\, Sig(\boldsymbol{g})$. Then there exists $t \in T : Sig(t\boldsymbol{f}) \sim Sig(\boldsymbol{g})$. We consider two cases:

- If $\boldsymbol{g}$ is an initial polynomial $\boldsymbol{f_i}$, then $Sig(\boldsymbol{f}) \,|\, Sig(\boldsymbol{g})$ implies $Sig(\boldsymbol{f}) \sim Sig(\boldsymbol{g})$. By the property of distinct signatures in $\boldsymbol{G}$, $\boldsymbol{f} = \boldsymbol{g}$ and thus, (3.4) follows trivially.
- If $\boldsymbol{g}$ is no initial polynomial, it was constructed via regular Sig-reductions and is regularly irreducible at the time of its construction. Since all elements added later to $\boldsymbol{G}$ have a larger signature than $\boldsymbol{g}$, it is still regularly irreducible at the end of the algorithm. Applying Proposition 3.4.17 for $u\boldsymbol{f}, 1\boldsymbol{g}$, we obtain $r_{\boldsymbol{f}} \leq r_{\boldsymbol{g}}$. For $r_{\boldsymbol{f}} < r_{\boldsymbol{g}}$, $\boldsymbol{f} \leq_{rat} \boldsymbol{g}$ holds by definition. So we are left to prove the case $r_{\boldsymbol{f}} = r_{\boldsymbol{g}}$: Since $Sig(\boldsymbol{f}) \,|\, Sig(\boldsymbol{g})$ implies $Sig(\boldsymbol{f}) \leq Sig(\boldsymbol{g})$, $\boldsymbol{f} \leq_{rat} \boldsymbol{g}$ follows as well.

(iii) $\leq_{F5}$: Let $Sig(\boldsymbol{f}) = ct\boldsymbol{f_i}, Sig(\boldsymbol{g}) = c'u\boldsymbol{f_j}$. Note that $Sig(\boldsymbol{f}) \mid Sig(\boldsymbol{g})$ implies

$$\operatorname{index} \boldsymbol{f} = i = j = \operatorname{index} \boldsymbol{g}, \quad deg(t) \leq deg(u).$$

Thus, $\boldsymbol{f} \leq_{F5} \boldsymbol{g}$ follows.

□

One might ask why are those specific orders considered? The idea for the number order is to choose the element added the latest since we hope that this element is reduced the farthest. For the ratio-order, see the following result:

Lemma 3.4.19. *Let $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}, \boldsymbol{s} \in T_m$ and let $u, v \in T$ such that $Sig(u\boldsymbol{f}) \sim Sig(v\boldsymbol{f}) \sim \boldsymbol{s}$. If $\boldsymbol{g}$ is the maximal element of $\boldsymbol{C_s}$ with respect to $<_{rat}$, then $LT(vg) \leq LT(uf)$.*

Proof. (compare [8, Remark 7.14]) Note that $\boldsymbol{f} <_{rat} \boldsymbol{g}$ implies

$$LT(g)Sig(\boldsymbol{f}) \leq LT(f)Sig(\boldsymbol{g}).$$

Multiplying the inequality by uv yields

$$LT(vg)\boldsymbol{s} = vLT(g)Sig(u\boldsymbol{f}) \leq uLT(f)Sig(v\boldsymbol{g}) = LT(uf)\boldsymbol{s}.$$

Hence, $LT(vg) \leq LT(uf)$ follows. □

Lemma 3.4.19 shows us that the ratio-order chooses an element $\boldsymbol{f}$ where uf has the lowest leading term among all elements in $\boldsymbol{C_s}$ where $Sig(u\boldsymbol{f}) \sim \boldsymbol{s}$. This seems to be a good heuristic since having a small leading term tends to imply few reduction steps. The order $<_{F5}$ has no certain characterization but is implicitly used in the F5 Algorithm as seen later. With the assumptions made for F5 to work correctly, it behaves similarly to the number order, but not exactly in the same way. The rule to break ties which was left arbitrary in Definition 3.4.15 is implicitly defined by F5 in the code. However, the way this rule is defined does not have an impact on correctness.
To choose the best rewrite order, note that Lemma 3.4.11 shows us that the reduction of an S-polynomial of signature $\boldsymbol{s}$ has to be computed if and only if the canonical rewriter of $\boldsymbol{s}$ is Sig-top-reducible. Note that by Proposition 3.4.3, we maximally compute the reduction of one S-polynomial for each signature, regardless of which rewrite order we choose. Thus, the following theorem shows that $\leq_{rat}$ is optimal in the following sense:

Theorem 3.4.20. *Algorithm 12 applied with rewrite order $\leq_{rat}$ leads to a minimal signature Gröbner basis.*

Proof. (compare [10, Theorem 13]) Let $\boldsymbol{G}$ be the rewrite basis obtained by Algorithm 12 with rewrite order $<_{rat}$. Assume to the contrary that there exists $\boldsymbol{g} \in \boldsymbol{G}$ with signature $\boldsymbol{t}$ such that $\boldsymbol{g}$ is

Sig-top-reducible by $\boldsymbol{G} \setminus \{\boldsymbol{g}\}$. Therefore, $\boldsymbol{g}$ is Sig-top-reducible by $\boldsymbol{G_t} := \{\boldsymbol{g} \in \boldsymbol{G} : Sig(\boldsymbol{g}) < \boldsymbol{t}\}$. By construction, $\boldsymbol{g}$ is regularly irreducible, thus singularly top-reducible. This means that there exist $u > 1 \in T, \boldsymbol{f} \in \boldsymbol{G_t}$:

$$Sig(u\boldsymbol{f}) \sim Sig(\boldsymbol{g}), \quad LT(uf) = LT(g).$$

Since $\boldsymbol{g}$ was added, it holds that $\boldsymbol{g} \in \overline{w\boldsymbol{h}}^{\boldsymbol{G},reg}$ where $w\boldsymbol{h}$ was the canonical rewriter of $\boldsymbol{t}$ at that time. By Lemma 3.4.11, $w\boldsymbol{h}$ is regularly top-reducible and hence,

$$LT(uf) = LT(g) < LT(wh)$$

follows. But $\boldsymbol{f} \in \boldsymbol{G_t}$ and thus, $u\boldsymbol{f}$ would rewrite $w\boldsymbol{h}$, a contradiction to $w\boldsymbol{h}$ being the canonical rewriter. $\square$

By Lemma 3.3.9, all minimal signature Gröbner bases have the same cardinality. Since one can clearly compute a minimal signature Gröber basis out of an arbitrary one by simply deleting not necessary elements, we have that Algorithm 12 with $\leq_{rat}$ as rewrite order computes the smallest signature Gröbner basis among all possible rewrite orders.

Remark 3.4.21. *Other rewrite orders actually lead, in many examples, to larger Gröbner bases. To see such an example where order $<_{F5}$ is used, see [10], Example 19.*

3.5 Regular sequences and F5

3.5.1 Regular sequences

In this section we will state the quite famous result (originally stated in [12] for F5) that Algorithm 12 works specifically well if we use the order extension $<_{pot}$ and the (ordered) input set F, has a specific structure called "regular sequence":

Definition 3.5.1 (Regular sequence). *Let $F = (f_1, ..., f_m)$ be a sequence of polynomials defining a zero-dimensional ideal with $deg(f_i) = d_i$.*

*(i) F is called to be a **d-regular sequence** for some $d \in \mathbb{N}_\infty$ if*

$$\forall i \in \{1, ..., m\}\ \forall g \in P \text{ with } deg(g) < d - d_i :$$
$$\text{If } gf_i \in \langle f_{i+1}, ..., f_m \rangle, \text{ then } g \in \langle f_{i+1}, ..., f_m \rangle.$$

*(ii) F is called to be a **regular sequence** if it is d-regular with $d = \infty$.*

Lemma 3.5.2. *(compare [23, Theorem 3.32 and Corollary 3.33]) Let F be a d-regular sequence, $\boldsymbol{f}$ not an initial polynomial with signature $t\boldsymbol{f_i}$ such that*

$$deg(Sig(\boldsymbol{f})) := deg(t) + deg(f_i) \leq d. \tag{3.5}$$

Then $\boldsymbol{f}$ will not be reduced to 0 by Algorithm 12 when using $<_{pot}$ as order extension.

Proof. Without loss of generality, let $i = 1$. Assume to the contrary that $\boldsymbol{f}$ regularly *Sig*-reduces to a syzygy $\boldsymbol{h} = \sum_{i=1}^{m} h_i \boldsymbol{f_i}$. Note that

$$h_1 f_1 = -\sum_{i=2}^{m} h_i f_i \in \langle f_2, ..., f_m \rangle.$$

Since $LM(h_1)\boldsymbol{f_1} = Sig(\boldsymbol{h}) = Sig(\boldsymbol{f})$, we have $deg(f_1 h_1) \leq d$. By the d-regularity of F, $h_1 \in \langle f_2, ..., f_m \rangle$ follows. In particular,

$$LT(h_1) \in LT(\langle f_2, ..., f_m \rangle).$$

Since $\boldsymbol{f}$ is examined after $\boldsymbol{f_1}$, $v(\boldsymbol{G})$ contains a Gröbner basis of $\langle f_2, ..., f_m \rangle$ by Lemma 3.4.14 at that time. Hence, there exists $\boldsymbol{g} \in \boldsymbol{G}$:

$$\text{index}(\boldsymbol{g}) > 1, \quad LT(g) \,|\, LT(h_1).$$

Note that $LT(g)\boldsymbol{f_1}$ would have been added to $\boldsymbol{H}$ at the time when the initial polynomial $\boldsymbol{f_1}$ was examined, which was before the reduction of $\boldsymbol{f}$. But then, $Sig(\boldsymbol{f})$ would have been a known syzygy signature and hence, $\boldsymbol{f}$ would have been discarded, a contradiction. □

Corollary 3.5.3. *If F is regular and no initial polynomial reduces to zero, no reduction to 0 is computed by Algorithm 12 when using $<_{pot}$ as order extension.*

Proof. Since all module elements $\boldsymbol{f}$ satisfy (3.5) for $d = \infty$, the statement follows directly from Lemma 3.5.2. □

Remark 3.5.4.

1. *An overdetermined system can never be regular. This follows from the algebraic fact that $K[x_1, ..., x_n]$ is a Cohen-Macaulay ring of Krull dimension n. Since in Cohen-Macaulay rings, the Krull dimension equals the longest regular sequence, the result follows. For a complete treatment of these definitions and statements, see [11, in particular Chapter 18]. Therefore, Corollary 3.5.3 does not say anything for overdetermined systems. However, one can state similar results to adapted definitions such that very few zero reductions are actually computed. On the first sight, this seems to suggest that the $<_{pot}$ order extension works well, but in that way we create more polynomials than necessary. Taking other order extensions, we create fewer polynomials, but increase the number of zero reductions again. We will come back to that topic at the end of this chapter.*

2. *One might be tempted to additionally implement Buchberger's criteria from section 2.3.2, but it can be shown (see [7]) that S-polynomials, which can be discarded due to Buchberger's criteria, are already detected by the criteria of Algorithm 12. Hence, implementing them does not decrease the number of examined S-polynomials.*

To round up this chapter, we want to state an alternative formulation of the Syzygy Criterion, when considering regular sequences:

Theorem 3.5.5. *(compare [10, Theorem 17]) Let $\boldsymbol{G}$ be a signature Gröbner basis up to $\boldsymbol{f_i}$ for some $i \in \{1, ..., m\}$ and let the order extension be $<_{top}$. Furthermore, let F be a regular sequence, $t \in T$. Then, the following statements are equivalent:*

(i) $t\boldsymbol{f_i}$ *is a known syzygy signature.*

(ii) $\exists \boldsymbol{g} \in \boldsymbol{G} : \operatorname{index}(\boldsymbol{g}) > i, LT(g) | t.$

To prove that, we need a Lemma which itself is an interesting structural result for regular sequences:

Lemma 3.5.6. *(compare with [23, Theorem 3.4]) Let $F = (f_1, ..., f_m)$. Then the following statements are equivalent:*

(i) F *is a regular sequence.*

(ii) $PSyz(F) = Syz(F)$.

Proof. For this proof, remember the definition of principal syzygies

$$\boldsymbol{s_{i,j}} := f_i \boldsymbol{f_j} - f_j \boldsymbol{f_i}, \quad 1 \leq j < i \leq m.$$

"(i)→(ii)": We show this inductively for $i = m$ to $i = 1$. Obviously,

$$Syz(\{f_m\}) = \{\mathbf{0}\} = PSyz(\{f_m\}),$$

so the induction basis is done. Note that it suffices to show the induction step from 2 to 1. For that, let $\boldsymbol{s} = \sum_{i=1}^{m} s_i \boldsymbol{f_i} \in Syz(\{f_1, ..., f_m\})$. This implies

$$s_1 f_1 = -\sum_{i=2}^{m} s_i f_i$$

and since $\{f_1, ... f_m\}$ is regular, $s_1 \in \langle f_2, ..., f_m \rangle$. Therefore,

$$s_1 = \sum_{i=2}^{m} g_i f_i \text{ for some } g_i \in P$$

and thus,

$$\sum_{i=2}^{m} f_1 g_i f_i = f_1 s_1 = -\sum_{i=2}^{m} s_i f_i.$$

Hence,

$$\boldsymbol{t} := \sum_{i=1}^{m} (f_1 g_i + s_i) \boldsymbol{f_i} \in Syz(\{f_2, ..., f_m\}).$$

By induction hypothesis, $\boldsymbol{t} \in PSyz(\{f_2, ..., f_m\}) \subseteq PSyz(F)$ follows. Since

$$\sum_{i=2}^{m} g_i \boldsymbol{s_{1,i}} \in PSyz(F)$$

and

$$\boldsymbol{t} + \sum_{i=2}^{m} g_i \boldsymbol{s_{1,i}} = \boldsymbol{t} + \sum_{i=2}^{m} g_i f_i \boldsymbol{f_1} - \sum_{i=2}^{m} g_i f_1 \boldsymbol{f_i} = \sum_{i=2}^{m} g_i f_i \boldsymbol{f_1} + \sum_{i=2}^{m} s_i \boldsymbol{f_i} = \boldsymbol{s},$$

we obtain $\boldsymbol{s} \in PSyz(F)$.

"(ii)$\rightarrow$(i)": We have to show that for $g \in P$ such that $gf_k \in \langle f_{k+1}, ..., f_m \rangle$, $g \in \langle f_{k+1}, ..., f_m \rangle$ follows. For that, assume, without loss of generality, $k = 1$. Since $gf_1 \in \langle f_2, ..., f_m \rangle$,

$$gf_1 = \sum_{i=2}^{m} g_i f_i, \quad g_i \in P.$$

Hence,

$$\boldsymbol{s} = g\boldsymbol{f_1} + \sum_{i=2}^{m} g_i \boldsymbol{f_i} \in Syz(F) = PSyz(F)$$

and therefore,

$$\boldsymbol{s} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} c_{i,j} \boldsymbol{s_{i,j}} \text{ with } c_{i,j} \in P.$$

This yields

$$g\boldsymbol{f_1} = \sum_{j=2}^{m} c_{1,j} f_j \boldsymbol{f_1}$$

and accordingly,

$$g = \sum_{j=2}^{m} c_{1,j} f_j \in \langle f_2, ..., f_m \rangle.$$

$\square$

Proof of Theorem 3.5.5. "(i)$\rightarrow$(ii)": Let $\boldsymbol{h} \in Syz(\{f_i, ..., f_m\})$ with $Sig(\boldsymbol{h}) \sim t\boldsymbol{f_i}$. Since F is regular, we know from Lemma 3.5.6 that $\boldsymbol{h} \in PSyz(\{f_i, ..., f_m\})$. Thus, we can write

$$\boldsymbol{h} = \sum_{k=i+1}^{m} \sum_{j=k+1}^{m} p_{j,k}(f_k \boldsymbol{f_j} - f_j \boldsymbol{f_k}), \text{ where } p_{j,k} \in P.$$

Therefore,

$$Sig(\boldsymbol{h}) \sim LT(\sum_{j=i+1}^{m} p_{j,i} f_j)\boldsymbol{f_i} = t\boldsymbol{f_i}.$$

Note that $\sum_{j=i+1}^{m} p_{j,i} f_j \neq 0$ since $\text{index}(\boldsymbol{h}) = i$. By Lemma 3.4.14, $v(\boldsymbol{G})$ is a Gröbner basis for

$\langle f_{i+1}, ..., f_m \rangle$ which implies that for

$$\tilde{g} = \sum_{j=i+1}^{m} p_{j,i} f_j \in \langle f_{i+1}, ..., f_m \rangle,$$

there exists a $\boldsymbol{g} \in \boldsymbol{G}$ such that

$$LT(g) \,|\, LT(\tilde{g}) = t, \quad \mathrm{index}(\boldsymbol{g}) > i.$$

"(ii)→(i)": Conversely, let $\boldsymbol{g} \in \boldsymbol{G}$ such that

$$LT(g) \,|\, t, \quad \mathrm{index}(\boldsymbol{g}) =: j > i.$$

Observe that $\boldsymbol{h} := g\boldsymbol{f_i} - f_i\boldsymbol{g}$ is a syzygy with $Sig(\boldsymbol{h}) = LT(g)\boldsymbol{f_i}$ and this syzygy signature is known since at the iteration we examined $\boldsymbol{f_i}$, $Sig(g\boldsymbol{f_i} - f_i\boldsymbol{g}) = LT(g)\boldsymbol{f_i}$ was added to $\boldsymbol{H}$. □

Remark 3.5.7. *Note that the condition (ii) of Theorem 3.5.5 was used by the original F5 algorithm in [12] which was designed specifically for regular sequences. The definition of "not normalized" in [12] is equivalent to condition (ii) of Theorem 3.5.5 and therefore, to the syzygy criterion of Algorithm 12 applied to regular input sequences. With some small additional considerations, it can be shown that F5 is a special case of Algorithm 12 (see [10]).*

Remark 3.5.8. *Most authors, in particular in [12] and many following research articles, consider only the $<_{pot}$ order extension. For that extension, well-structured results like the one in Corollary 3.5.3 about the number of reductions to zero can be shown. However, this order extension leads to a big issue as well: As we have seen in Lemma 3.4.14, the algorithm incrementally computes signature Gröbner bases $\boldsymbol{G_i}$ for $\langle f_i, ..., f_m \rangle$. In many cases, such intermediate bases are much larger than a signature Gröbner basis of $\langle f_1, ..., f_m \rangle$ needs to be. A proposal to overcome this was F5R in [23] and the further development into F5C in [9] with the following idea: Once we obtain a signature Gröbner basis $\boldsymbol{G_i}$, we extract $v(\boldsymbol{G_i})$ and interreduce this Gröbner basis to obtain a minimal Gröbner basis $\{g_1, ..., g_k\}$ for $\langle f_i, ..., f_m \rangle$. To obtain correct signatures again, we consider $\{f_1, ...f_{i-1}, g_1, ..., g_k\}$ as the new initial polynomials and extend the signature accordingly (the considered module is now P^{i+k-1}). In that way, we reduce the number of examined S-polynomials a lot since we always interreduce the intermediate Gröbner bases. Nevertheless, the algorithm still needs to work with those (reduced) intermediate Gröbner bases, which on its own might be much larger than the final Gröbner basis for $\langle f_1, ..., f_m \rangle$. Thus, the definition of other order extensions was introduced in [18] with some of them working empirically better than $<_{pot}$.*

4 M5GB - a new hybrid approach

In this section we want to combine the ideas about rewrite bases and those of M4GB. The result we aim to achieve is an algorithm which combines the strengths of both algorithms:

- The fast reduction of polynomials due to M4GB structure.
- The well-working criteria from signature algorithms to discard almost all unnecessary S-polynomials.

4.1 Idea explanation

Remember from Remark 3.1.8 that bold small letters, e.g. $\boldsymbol{f}, \boldsymbol{g}$ are module elements with f, g being their images under the evaluation homomorphism $v = v_F$ where $F = \{f_1, ..., f_m\}$ is the set of input polynomials. The overall structure of the algorithm discussed below is mostly the same as in Algorithm 12, only differing in the regular reduction step. For the sake of simplicity, we will state the idea in module elements again as we did for Algorithm 10.

Algorithm 13: M5GB main routine

Input: Input polynomials $F = \{f_1, ..., f_m\}$, a rule for a rewrite order $<$ on all $\boldsymbol{G} \cup \boldsymbol{H}$ created during the algorithm and an order extension $\lessdot$ on T_m.

Output: rewrite basis $\boldsymbol{G}$ of $\langle F \rangle$, in particular $v(\boldsymbol{G})$ is a Gröbner basis

Set $\boldsymbol{G} = \emptyset$;
Set $\boldsymbol{M} = \emptyset$;//usage of $\boldsymbol{M}$ described later
Set $\boldsymbol{P} = \{\boldsymbol{f_i} : i \in \{1, ..., m\}\}$;
Set $\boldsymbol{H} = \emptyset$;
while $\boldsymbol{P} \neq \emptyset$
    Let $\boldsymbol{f} = u\boldsymbol{g}_1 - v\boldsymbol{g}_2$ be a minimal element of $\boldsymbol{P}$ with respect to their signatures.;
    $\boldsymbol{P} = \boldsymbol{P} \backslash \{\boldsymbol{f}\}$;
    if $u\boldsymbol{g}_1$ *and* $v\boldsymbol{g}_2$ *are not rewriteable* **then**
        $\boldsymbol{f}' =$ M5GB-Reduce$(\boldsymbol{f}, \boldsymbol{M}, \boldsymbol{G})$;
        if $f' = 0$ **then**
            $\boldsymbol{H} = \boldsymbol{H} \cup \{Sig(\boldsymbol{f}')\}$;
        end

else

  $\boldsymbol{P} = \boldsymbol{P} \cup \{Spol(\boldsymbol{f'}, \boldsymbol{g}) = u\boldsymbol{f'} - v\boldsymbol{g} : Sig(u\boldsymbol{f'}) \neq Sig(v\boldsymbol{g}), \boldsymbol{g} \in \boldsymbol{G}\}$;

  $\boldsymbol{G} = \boldsymbol{G} \cup \{LC(f')^{-1}\boldsymbol{f'}\}$;//add the normalized element

  if $Sig(\boldsymbol{f'}) = \boldsymbol{f_i}$ **then**

    //$\boldsymbol{f'}$ comes from an initial polynomial $\boldsymbol{f_i}$

    $\boldsymbol{H} = \boldsymbol{H} \cup \{Sig(g\boldsymbol{f_i} - f_i\boldsymbol{g}) : \boldsymbol{g} \in \boldsymbol{G}\}$;

  end

end

end

end

return $\boldsymbol{G}$;

The rest of this section is to discuss the function M5GB-Reduce which should compute $f' \in \overline{f}^{\boldsymbol{G},\boldsymbol{reg}}$. A single reduction step will be similar to M4GB with the following changes: First, we need to change the perspective to modules again, making the sets $\boldsymbol{M}, \boldsymbol{G} \subseteq P^m$. Analogously to M4GB, the set $\boldsymbol{M}$ should contain Sig-tail-irreducible monomial multiples of $\boldsymbol{G}$. $\boldsymbol{M}$ will be augmented when needed analogously to M4GB when a term t is Sig-reducible by $\boldsymbol{G}$, but $t \notin LT(v(\boldsymbol{M}))$ yet. As in Algorithm 12, we want $\boldsymbol{G}$ to remain a rewrite basis and hence a signature Gröbner basis up to the current examined signature. The following problems might occur:

(i) We add a new element $\boldsymbol{f}$ to $\boldsymbol{G}$ with $vLT(f) = t \in T(Tail(g))$ for some $\boldsymbol{g} \in \boldsymbol{M}$. Assume that in a later reduction step, $\boldsymbol{g}$ is chosen to reduce a polynomial $\boldsymbol{h}$ as in M4GB. But since $\boldsymbol{h} - \boldsymbol{g}$ has a regularly reducible term which was not there in $\boldsymbol{h}$, the advantageous property of M4GB is destroyed.

(ii) Assume we already have some $\boldsymbol{f} \in \boldsymbol{G}$ with $vLT(f) = t \in T(Tail(g))$ for some $\boldsymbol{g} \in \boldsymbol{M}$, but $Sig(v\boldsymbol{f}) > Sig(\boldsymbol{g})$. Hence, t was Sig-reducible when $\boldsymbol{g}$ was initially created. Analogously to (i), this reduction becomes regular when the examined signature increases, causing the same problem.

The first point is an issue that also occurs in the M4GB algorithm and is there treated by the introduction of a new generation when an element $\boldsymbol{f}$ is added to $\boldsymbol{G}$. Doing the same for (ii) seems computationally very expensive since we would have to introduce a new generation at each S-polynomial examined and in that way, we almost always need to check whether some $\boldsymbol{g} \in \boldsymbol{M}$ fulfills case (ii), implying the computation of an additional Tail-Reduction. This is the reason why we came up with the idea to additionally add a so-called "signature flag" to each module element. The idea of that flag is to define it as the minimal signature for that (ii) might cause a problem. In that way, if this signature is not yet reached, $\boldsymbol{g} \in \boldsymbol{G}$ is still (Sig-)tail-irreducible and does not need to be updated.

4.2 New definitions

To make these ideas more precise, consider the following definition which generalizes the idea of signature reductions:

Definition 4.2.1 (Reduction up to a signature). *Let* $\boldsymbol{f} \in P^m$, $u \in T, \boldsymbol{s} \in T_m \cup \{\infty\}$. *Furthermore, let* $\boldsymbol{G} \subseteq P^m$ *be a set of monic module elements.*

- *We say* u *is* ***regularly reducible by G and up to s*** *if there exist* $\boldsymbol{g} \in \boldsymbol{G}, v \in T$:

$$vLT(g) = u, \quad Sig(v\boldsymbol{g}) < \boldsymbol{s}.$$

- *We say* $\boldsymbol{f}$ *is* ***regularly reducible by G and up to s*** *if there exists* $t \in T(f)$ *such that* t *is regularly reducible by* $\boldsymbol{G}$ *and up to* $\boldsymbol{s}$. *If* $v\boldsymbol{g}$ *is such a Sig-reducer of* $\boldsymbol{f}$, *we define such a reduction step by*

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{} \boldsymbol{f} - c_t(f)v\boldsymbol{g}$$

and a finite number of such reduction steps (including 0 steps) by

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f'}.$$

- *We define*

$$T_{\boldsymbol{G}}(f, \boldsymbol{s}) := \{t \in T(f) : t \text{ is regularly reducible with respect to } \boldsymbol{G} \text{ and up to } \boldsymbol{s}\}.$$

- $\boldsymbol{f}$ *is called* ***Sig-irreducible with respect to G and up to s*** *if* $T_{\boldsymbol{G}}(f, \boldsymbol{s}) = \emptyset$.
- $\boldsymbol{f'} \in P^m$ *is called to be a* ***normal form of f with respect to G and up to s*** *if*

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f'}$$

and $\boldsymbol{f'}$ *is Sig-irreducible with respect to* $\boldsymbol{G}$ *and up to* $\boldsymbol{s}$. *We define*

$$\overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}} := \{\boldsymbol{f'} \in P^m : \boldsymbol{f'} \text{ is a normal form of } \boldsymbol{f} \text{ with respect to } \boldsymbol{G} \text{ and up to } \boldsymbol{s}\}.$$

- $\boldsymbol{f}$ *is called* ***Sig-tail-irreducible with respect to G and up to s*** *if* $T_{\boldsymbol{G}}(Tail(f), \boldsymbol{s}) = \emptyset$.
- $\boldsymbol{f'}$ *is called to be a* ***tail normal form of f with respect to G and up to s*** *if*

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f'},$$

$LM(f) = LM(f')$ *and* $\boldsymbol{f'}$ *is Sig-tail-irreducible with respect to* $\boldsymbol{G}$ *and up to* $\boldsymbol{s}$. *We define*

$$\overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{s}} := \{\boldsymbol{f'} \in P^m : \boldsymbol{f'} \text{ is a tail normal form of } \boldsymbol{f} \text{ with respect to } \boldsymbol{G} \text{ and up to } \boldsymbol{s}\}.$$

Remark 4.2.2.

1. *If $\boldsymbol{s} \sim Sig(\boldsymbol{f})$, then $\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{} \boldsymbol{f'}$ coincides with $\boldsymbol{f} \xrightarrow[\boldsymbol{G},reg]{} \boldsymbol{f'}$ and therefore, $\overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}} = \overline{\boldsymbol{f}}^{\boldsymbol{G},reg}$.*

2. *If $\boldsymbol{s} = \infty$, then $\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{} \boldsymbol{f'}$ coincides with $f \xrightarrow[v(\boldsymbol{G})]{} f'$ since the signature condition from Definition 4.2.1 is always fulfilled.*

Similar to the often used Lemma 3.2.7, we can state the following properties for normal forms:

Lemma 4.2.3. *Let $\boldsymbol{f} \in P^m, \boldsymbol{G}$ a signature Gröbner basis up to $\boldsymbol{s}$ and $Sig(\boldsymbol{f}) < \boldsymbol{s}$. Then the following two statements hold:*

(i) If $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}}$, then $\boldsymbol{f'}$ is a syzygy.

(ii) If $\boldsymbol{g}, \boldsymbol{g'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}}$, then $g = g'$.

Proof.

(i): Since $Sig(\boldsymbol{f'}) < \boldsymbol{s}$ and $\boldsymbol{G}$ is a signature Gröbner basis up to $\boldsymbol{s}$, we know that $\boldsymbol{f'} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} 0$. By definition, $\boldsymbol{f'}$ is Sig-irreducible with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$, thus $\boldsymbol{f'}$ is already a syzygy itself.

(ii): Note that $LM(g) = LM(g')$ and hence, $LT(g - g') < LT(f)$. Since $\boldsymbol{g}, \boldsymbol{g'}$ are both Sig-irreducible with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$, $\boldsymbol{g} - \boldsymbol{g'}$ is it as well. Furthermore, since $Sig(\boldsymbol{g} - \boldsymbol{g'}) < \boldsymbol{s}$, $\boldsymbol{g} - \boldsymbol{g'} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} 0$. Therefore, $\boldsymbol{g} - \boldsymbol{g'}$ is a syzygy, yielding $g = g'$.

□

The following statements contain both the basic ideas for the M5GB Algorithm as well as the main part for its correctness. We will start with a result which seems trivial, but nevertheless is crucial for further discussion:

Proposition 4.2.4. *Let $\boldsymbol{f} \in P^m, \boldsymbol{s}, \boldsymbol{t} \in T_m, \boldsymbol{s} \geq \boldsymbol{t}$.*

(i) For $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{t}}$, we have $\overline{\boldsymbol{f'}}^{\boldsymbol{G},\boldsymbol{s}} \subseteq \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}}$.

(ii) For $\boldsymbol{g} \in \overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{t}}$, we have $\overline{\boldsymbol{g}}^{Tail,\boldsymbol{G},\boldsymbol{s}} \subseteq \overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$.

Proof. Since $\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f'}$ for (i) and $\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{g}, LM(f) = LM(g)$ for (ii), the statements follow immediately. □

Lemma 4.2.5. *Let $\boldsymbol{f} \in P^m, \boldsymbol{s} \in T_m \cup \{\infty\}$ and let $\boldsymbol{G} \subseteq P^m$ be a set of monic module elements. Fixing for all $t \in T_{\boldsymbol{G}}(f, \boldsymbol{s})$ some $\boldsymbol{m_t} \in \overline{v_t \boldsymbol{g_t}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$ where $\boldsymbol{g_t} \in \boldsymbol{G}, LT(v_t g_t) = t, Sig(v_t \boldsymbol{g_t}) < \boldsymbol{s}$. Then*

$$\boldsymbol{f'} := \boldsymbol{f} - \sum_{t \in T_{\boldsymbol{G}}(f,\boldsymbol{s})} C_t(f) \boldsymbol{m_t} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}}.$$

In particular, if $\boldsymbol{s} \sim Sig(\boldsymbol{f}), \boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},reg}$.

Proof. Since

$$T_{\boldsymbol{G}}(f', \boldsymbol{s}) \subseteq \bigcup_{t \in T(f)} T_{\boldsymbol{G}}(m_t, \boldsymbol{s}) = \emptyset,$$

it follows that $\boldsymbol{f'}$ is regularly Sig-irreducible with respect to $\boldsymbol{G}$. So we are left to show that $\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f'}$: For that reason, let

$$T_{\boldsymbol{G}}(f, \boldsymbol{s}) = \{t_1, ..., t_k\} \text{ with } t_1 \geq t_2 \geq ... \geq t_k.$$

If $T_{\boldsymbol{G}}(f, \boldsymbol{s}) = \emptyset$, then $\boldsymbol{f'} = \boldsymbol{f}$ and the statement follows trivially. Otherwise, we start to reduce the term t_k by its corresponding Sig-reducer $c_{t_k}(f)v_k\boldsymbol{g_k}$. Note that

$$T_{\boldsymbol{G}}(f, \boldsymbol{s}) \cap T_{\boldsymbol{G}}(Tail(v_k g_k), \boldsymbol{s}) = \emptyset$$

since if both sets are nonempty,

$$\min T_{\boldsymbol{G}}(f, \boldsymbol{s}) > \max T(Tail(v_k g_k), \boldsymbol{s}) \geq \max T_{\boldsymbol{G}}(Tail(v_k g_k), \boldsymbol{s}).$$

Thus, all Sig-reductions up to $\boldsymbol{s}$ for $Tail(c_{t_k}(f)v_k\boldsymbol{g_k})$ are Sig-reductions for $\boldsymbol{f} - c_{t_k}(f)v_k\boldsymbol{g_k}$, hence,

$$\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f} - c_{t_k}(f)v_k\boldsymbol{g_k} + \sum_{j=1}^{l_k} v_{k,j}\boldsymbol{g_{k,j}} = \boldsymbol{f} - \boldsymbol{m_{t_k}}.$$

Using that argument iteratively for $t_{k-1}, ..., t_1$, we obtain that $\boldsymbol{f} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{f'}$ and therefore, $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}}$ follows. □

Corollary 4.2.6. *Let $\boldsymbol{f} \in P^m, \boldsymbol{s} \in T_m$ and let $\boldsymbol{G} \subseteq P^m$ be a set of monic module elements. Fixing for all $t \in T_{\boldsymbol{G}}(Tail(f), Sig(\boldsymbol{f}))$ some $\boldsymbol{m_t} \in \overline{v_t\boldsymbol{g_t}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$ where $\boldsymbol{g_t} \in \boldsymbol{G}$, $LT(v_t g_t) = t$, $Sig(v_t\boldsymbol{g_t}) < \boldsymbol{s}$. Then*

$$\boldsymbol{f'} := \boldsymbol{f} - \sum_{t \in T_{\boldsymbol{G}}(Tail(f),\boldsymbol{s})} C_t(f)\boldsymbol{m_t} \in \overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{s}}.$$

Proof. The proof can be made completely analogously to that of Lemma 4.2.5 except using $T_{\boldsymbol{G}}(Tail(f), \boldsymbol{s})$ instead of $T_{\boldsymbol{G}}(f, \boldsymbol{s})$. □

Remark 4.2.7. *Lemma 4.2.5 with the special case of $\boldsymbol{s} = \infty$ (i.e. all reductions are allowed) can be applied to prove Lemma 2.6.5 from section 2.6. Together with Corollary 4.2.6 and Proposition 4.2.4, applied with $\boldsymbol{s} = \infty$, we obtain the missing part of correctness for M4GB.*

Lemma 4.2.5 and Corollary 4.2.6 include the main ideas how to compute the regular reduction step in the M5GB Algorithm: For $t \in T_{\boldsymbol{G}}(f, Sig(\boldsymbol{f}))$, we have $\boldsymbol{m_t} \in \overline{v_t\boldsymbol{g_t}}^{Tail,\boldsymbol{G},Sig(\boldsymbol{f})}$ and apply Lemma 4.2.5 to get some $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},reg}$. To compute $\boldsymbol{m_t}$, we use Corollary 4.2.6 with $\boldsymbol{s} = Sig(\boldsymbol{f})$, $\boldsymbol{f} = v_t\boldsymbol{g_t}$. For that, we apply Corollary 4.2.6 recursively. To increase efficiency, we save at the reduction of some $\boldsymbol{f}$ for given $t \in T$ the element $\boldsymbol{m_t}$ in a new set $\boldsymbol{M} \subseteq P^m$. If we have some $\boldsymbol{g} \in P^m$ with $Sig(\boldsymbol{g}) > Sig(\boldsymbol{f}), t \in T(g)$, we use Proposition 4.2.4 to start with the already partially reduced $\boldsymbol{f'} \in \overline{v_t \boldsymbol{g_t}}^{Tail,\boldsymbol{G},Sig(\boldsymbol{f})}$ for computing some $\boldsymbol{g'} \in \overline{v_t \boldsymbol{g_t}}^{Tail,\boldsymbol{G},Sig(\boldsymbol{g})}$ instead of starting over again with $v_t \boldsymbol{g_t}$. The advantage is that we only have to reduce terms in $T_{\boldsymbol{G}}(g, Sig(\boldsymbol{g})) \setminus T_{\boldsymbol{G}}(g, Sig(\boldsymbol{f}))$ instead of all terms in $T_{\boldsymbol{G}}(g, Sig(\boldsymbol{g}))$.

4.3 Basic Pseudocode

The ideas gathered in Section 4.2 are already enough to state a first M5GB-reduction pseudo code:

Algorithm 14: M5GB-Reduce

Input: $\boldsymbol{f} \in P^m$, $\boldsymbol{G}, \boldsymbol{M} \subseteq P^m$.

Output: Possibly changed set $\boldsymbol{M}$, $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},reg}$.

Set $\boldsymbol{f'} = \boldsymbol{f}$;
for $t \in T_{\boldsymbol{G}}(f, Sig(\boldsymbol{f}))$ **do**
  $//\exists \boldsymbol{g} \in \boldsymbol{G}, v \in T$ such that $LT(vg) = t, Sig(v\boldsymbol{g}) < Sig(\boldsymbol{f})$
  if $\exists \boldsymbol{m} \in \boldsymbol{M} : LT(m) = t$ **then**
    $\boldsymbol{m'} =$ M5GB-Tail-Reduce$(\boldsymbol{m}, Sig(\boldsymbol{f}), \boldsymbol{G}, \boldsymbol{M})$;
    $\boldsymbol{M} = \boldsymbol{M} \setminus \{\boldsymbol{m}\}$;
  end
  else
    $\boldsymbol{m'} =$ M5GB-Tail-Reduce$(v\boldsymbol{g}, Sig(\boldsymbol{f}), \boldsymbol{G}, \boldsymbol{M})$;
  end
  $\boldsymbol{f'} = \boldsymbol{f'} - c_t(f)\boldsymbol{m'}$;
  $\boldsymbol{M} = \boldsymbol{M} \cup \{\boldsymbol{m'}\}$;
end
return $\boldsymbol{f'}$;

Algorithm 15: M5GB-Tail-Reduce

Input: $\boldsymbol{f} \in P^m$, $\boldsymbol{s} \in T_m, \boldsymbol{G}, \boldsymbol{M} \subseteq P^m$.

Output: Possibly changed set $\boldsymbol{M}$, $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$.

Set $\boldsymbol{f'} = \boldsymbol{f}$;
for $t \in T_{\boldsymbol{G}}(Tail(f), \boldsymbol{s})$
  $//\exists \boldsymbol{g} \in \boldsymbol{G}, v \in T$ such that $LT(vg) = t, Sig(v\boldsymbol{g}) < Sig(\boldsymbol{f})$
  if $\exists \boldsymbol{m} \in M : LT(m) = t$ **then**
    $\boldsymbol{m'} =$ M5GB-Tail-Reduce$(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{G}, \boldsymbol{M})$;
    $\boldsymbol{M} = \boldsymbol{M} \setminus \{\boldsymbol{m}\}$;

```
    else
        m' = M5GB-Tail-Reduce(vg, s, G, M);
    end
    f' = f' − c_t(f)m';
    M = M ∪ {m'};
end
return f';
```

Theorem 4.3.1. *The basic M5GB Algorithm is correct.*

Proof. Since the main layout is the same as the one for Algorithm 12, it suffices to show that M5GB-Reduce is correct. To see that, note that M5GB-reduction is called for module elements in increasing signature and hence, correctness follows from Lemma 4.2.3, Proposition 4.2.4, Lemma 4.2.5 and Corollary 4.2.6. Since we decrease the leading term of the polynomial in each call, the recursion depth is finite, hence the routine eventually terminates. □

4.4 Improvements

In this section, we will concentrate on mathematical improvements and not on implementation details which are shortly discussed later. There are many ways how we can improve this basic algorithm, namely:

1. Implementing generations and signature flags.
2. Fast checks for irreducibility.
3. Doing a labelled polynomial optimization as in Section 3.3.2.
4. Checking divisibility by $\boldsymbol{M}$ instead of $\boldsymbol{G}$.

4.4.1 Generations and signature flags

A major problem of the basic M5GB-reduction is that we call M5GB-Tail-Reduce at all times, even when some $\boldsymbol{m} \in \boldsymbol{M}$ is found which is Sig-tail-irreducible up to the current signature. To avoid those useless comparisons, we introduce generations like in M4GB and signature flags as two fast checks whether some $\boldsymbol{m} \in \boldsymbol{M}$ is still Sig-tail-irreducible. Assume that some element $\boldsymbol{m}$ is added to $\boldsymbol{M}$ when some element $\boldsymbol{f}$ is M5GB-reduced. Then, $\boldsymbol{m}$ is by construction Sig-tail-irreducible with respect to $\boldsymbol{G}_{Sig(\boldsymbol{f})}$ and up to $Sig(\boldsymbol{f})$. When $\boldsymbol{m}$ is called later when reducing some $\boldsymbol{g}$ with respect to $\boldsymbol{G}_{Sig(\boldsymbol{g})}$ where $Sig(\boldsymbol{g}) > Sig(\boldsymbol{f})$, there are, as mentioned in Section 4.1, two possibilities for $\boldsymbol{m}$ being not Sig-tail-irreducible with respect to $\boldsymbol{G}_{Sig(\boldsymbol{g})}$ and up to $Sig(\boldsymbol{g})$:

(i) $\exists t \in T(Tail(m)), \boldsymbol{h} \in \boldsymbol{G}_{Sig(\boldsymbol{f})}, v \in T, Sig(\boldsymbol{f}) \leq Sig(v\boldsymbol{h}) < Sig(\boldsymbol{g}) : LT(vh) = t.$

(ii) $\exists t \in T(Tail(m)), \boldsymbol{h} \in \boldsymbol{G}_{Sig(\boldsymbol{g})} \backslash \boldsymbol{G}_{Sig(\boldsymbol{f})}, v \in T, Sig(v\boldsymbol{h}) < Sig(\boldsymbol{g}) : LT(vh) = t.$

To treat (i), we introduce the signature flag:

Definition 4.4.1 (Signature flag). *Let $\boldsymbol{m}$ be constructed when some element $\boldsymbol{f}$ is M5GB-reduced. We define*

$$Flag(\boldsymbol{m}) := \inf\{\boldsymbol{s} \in T_m \cup \{\infty\} : \boldsymbol{m} \text{ is Sig-tail-reducible with respect to } \boldsymbol{G}_{Sig(\boldsymbol{f})} \text{ and up to } \boldsymbol{s}\}.$$

Remark 4.4.2. *Note that this set is always finite. Hence, if the set is non-empty, we can define it as minimum as well. For the case of an empty set, we apply the definition of the infimum to obtain the signature ∞.*

If we have $Flag(\boldsymbol{m}) = \boldsymbol{s}$ and $Sig(\boldsymbol{g}) \leq \boldsymbol{s}$, then (i) is not possible. The idea to compute this signature flag is straightforward: During M5GB-Tail-reduce, we check for each term $t \in T(Tail(m)) \backslash T_{\boldsymbol{G}_{Sig(\boldsymbol{f})}}(Tail(m), \boldsymbol{s})$ if there exist $\boldsymbol{h} \in \boldsymbol{G}, v \in T$ such that

$$LT(vh) = t, \quad Sig(v\boldsymbol{h}) > Sig(\boldsymbol{f}). \tag{4.1}$$

We obtain $Flag(\boldsymbol{m})$ as the minimal signature of those $v\boldsymbol{h}$ fulfilling the properties in (4.1).

Remark 4.4.3. *Note that*

$$\inf\{Sig(v\boldsymbol{h}) > Sig(\boldsymbol{f}) : \boldsymbol{h} \in \boldsymbol{G}_{Sig(\boldsymbol{f})}, v \in T : LT(vh) \in T(Tail(m)) \backslash T_{\boldsymbol{G}_{Sig(\boldsymbol{f})}}(Tail(m), \boldsymbol{s})\} =$$
$$\inf\{\boldsymbol{s} \in T_m : \boldsymbol{m} \text{ is Sig-tail-reducible with respect to } \boldsymbol{G}_{Sig(\boldsymbol{f})} \text{ and up to } \boldsymbol{s}\}$$

if

$$\{\boldsymbol{h} \in \boldsymbol{G}_{Sig(\boldsymbol{f})}, v \in T : LT(vh) \in T(Tail(m)) \backslash T_{\boldsymbol{G}_{Sig(\boldsymbol{f})}}(Tail(m), \boldsymbol{s}), Sig(v\boldsymbol{h}) > Sig(\boldsymbol{f})\}$$

is nonempty. If this set is empty, m is irreducible and hence,

$$\inf\{\boldsymbol{s} \in T_m : \boldsymbol{m} \text{ is Sig-tail-reducible with respect to } \boldsymbol{G}_{Sig(\boldsymbol{f})} \text{ and up to } \boldsymbol{s}\} = \infty.$$

To treat (ii), we introduce generations with the same idea already used in [20] for M4GB:

Definition 4.4.4 (Generation). *Let $\boldsymbol{m}$ be constructed when some element $\boldsymbol{f}$ is M5GB-reduced. We define*

$$Gen(\boldsymbol{m}) := |\boldsymbol{G}_{Sig(\boldsymbol{f})}|.$$

If we have $Gen(\boldsymbol{m}) = k$ and $|\boldsymbol{G}_{Sig(\boldsymbol{g})}| = k$, then (ii) is not possible. To sum up, assume we want to reduce some $\boldsymbol{g}$ by some $\boldsymbol{m}$ where $Flag(\boldsymbol{m}) \geq Sig(\boldsymbol{g})$ and $Gen(\boldsymbol{m}) = |\boldsymbol{G}_{Sig(\boldsymbol{g})}|$. Then we know that $\boldsymbol{m}$ is Sig-tail-irreducible up to $Sig(\boldsymbol{g})$ and hence, can be directly used to reduce $\boldsymbol{g}$. Let us consider now what we do if one of those cases is not fulfilled. If $Gen(\boldsymbol{m}) = |\boldsymbol{G}_{Sig(\boldsymbol{f})}| < |\boldsymbol{G}_{Sig(\boldsymbol{g})}|$, we check if

$$\{\boldsymbol{h} \in \boldsymbol{G}_{Sig(\boldsymbol{g})} \backslash \boldsymbol{G}_{Sig(\boldsymbol{f})} : \exists t \in T(Tail(g)), v \in T : LT(vh) = t\}$$

is nonempty. If this is the case, we call Tail-M5GB-reduce on all those $v\boldsymbol{h}$ and reduce $\boldsymbol{m}$ by that $\overline{v\boldsymbol{h}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$. Note that we need to update the signature flag of this reduced element as well, in the same way as discussed above. However, it is sufficient to look through $\boldsymbol{G}_{Sig(\boldsymbol{g})} \backslash \boldsymbol{G}_{Sig(\boldsymbol{f})}$. To sum up, see the improved M5GB-Tail-reduce routine:

Algorithm 16: Flag-Generation M5GB-Tail-Reduce

Input: $\boldsymbol{f} \in P^m$, $\boldsymbol{s} \in T_m, \boldsymbol{G}, \boldsymbol{M} \subseteq P^m$ where $\boldsymbol{M}, \boldsymbol{G}$ are equipped with signature flags and generations.

Output: Possibly changed set $\boldsymbol{M}$, $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$.

Set $\boldsymbol{f'} = \boldsymbol{f}$;
for $t \in T_{\boldsymbol{G}}(Tail(f), \boldsymbol{s})$ **do**
    $//\exists \boldsymbol{g} \in \boldsymbol{G}, v \in T$ such that $LT(vg) = t, Sig(v\boldsymbol{g}) < Sig(\boldsymbol{f})$
    if $\exists \boldsymbol{m} \in \boldsymbol{M} : LT(m) = t$ **then**
        $\boldsymbol{m'} =$ Update$(\boldsymbol{m}, \boldsymbol{sG}, \boldsymbol{M})$;
        $\boldsymbol{M} = \boldsymbol{M} \backslash \{\boldsymbol{m}\}$;
    end
    else
        $\boldsymbol{m'} =$Flag-Generation M5GB-Tail-Reduce$(v\boldsymbol{g}, \boldsymbol{s}, \boldsymbol{G}, \boldsymbol{M})$;
        $Gen(\boldsymbol{m'}) = |\boldsymbol{G}|$;
    end
    $\boldsymbol{f'} = \boldsymbol{f'} - c_t(f)\boldsymbol{m'}$;
    $\boldsymbol{M} = \boldsymbol{M} \cup \{\boldsymbol{m'}\}$;
end
CreateFlag$(\boldsymbol{f'}, \boldsymbol{G})$;
$Gen(\boldsymbol{f'}) = |\boldsymbol{G}|$;
return $\boldsymbol{f'}$;

Algorithm 17: Update

Input: $\boldsymbol{f} \in P^m, \boldsymbol{s} \in T_m, \boldsymbol{G}, \boldsymbol{M} \subseteq P^m$ where $\boldsymbol{M}, \boldsymbol{G}$ are equipped with generations and signature flags.

Output: Possibly changed set $\boldsymbol{M}$, $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$.

if $Gen(\boldsymbol{f}) = |\boldsymbol{G}|$ **then**
    if $Flag(\boldsymbol{f}) \geq \boldsymbol{s}$ **then**
        return $\boldsymbol{f}$;
    end

---

else
    return Flag-Generation M5GB-Tail-Reduce($\boldsymbol{f}, \boldsymbol{s}, \boldsymbol{G}, \boldsymbol{M}$);
end
end
else
    for $t \in T_{\boldsymbol{G}\setminus\boldsymbol{G}_{\boldsymbol{f}}}(Tail(f), \boldsymbol{s})$ **do**
        $//\exists \boldsymbol{g} \in \boldsymbol{G}\setminus\boldsymbol{G}_{Sig(\boldsymbol{f})}, v \in T$ such that $LT(vg) = t, Sig(v\boldsymbol{g}) < Sig(\boldsymbol{f})$
        if $\exists \boldsymbol{m} \in \boldsymbol{M} : LT(m) = t$ **then**
            $\boldsymbol{m'}$ =Update($\boldsymbol{m}, \boldsymbol{sG}, \boldsymbol{M}$);
            $\boldsymbol{M} = \boldsymbol{M}\setminus\{\boldsymbol{m}\}$;
        end
        else
            $\boldsymbol{m'}$ =Flag-Generation M5GB-Tail-Reduce($v\boldsymbol{g}, \boldsymbol{s}, \boldsymbol{G}, \boldsymbol{M}$);
            $Gen(\boldsymbol{m'}) = |\boldsymbol{G}|$;
        end
        $\boldsymbol{f'} = \boldsymbol{f'} - c_t(f)\boldsymbol{m'}$;
        $\boldsymbol{M} = \boldsymbol{M} \cup \{\boldsymbol{m'}\}$;
    end
    if $Flag(\boldsymbol{f}) \geq \boldsymbol{s}$ **then**
        $Flag(\boldsymbol{f'}) = \min\{Flag(\boldsymbol{f}), CreateFlag(\boldsymbol{f'}, \boldsymbol{G}_{Sig(\boldsymbol{f})})\}$;
        return $\boldsymbol{f'}$;
    end
    else
        return Flag-Generation M5GB-Tail-Reduce($\boldsymbol{f'}, \boldsymbol{s}, \boldsymbol{G}, \boldsymbol{M}$);
    end
end

Algorithm 18: CreateFlag

Input: $\boldsymbol{f} \in P^m, \boldsymbol{s} \in T_m, \boldsymbol{G} \subseteq P^m$.

Output: Signature $\boldsymbol{s} \in T_m \cup \{\infty\}$ such that $\boldsymbol{f}$ is Sig-tail-irreducible with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$

if $\{\boldsymbol{g} \in \boldsymbol{G} \mid \exists v \in T, t \in T(Tail(f)) : LT(vg) = t\} \neq \emptyset$ **then**
    return $\min\{Sig(v\boldsymbol{g}) \mid v \in T, \boldsymbol{g} \in \boldsymbol{G}, t \in T(Tail(f)) : LT(vg) = t\}$;
end
else
    return ∞;
end

Theorem 4.4.5. *Let $\boldsymbol{m}$ be Sig-tail-irreducible with respect to $\boldsymbol{G}_{Sig(\boldsymbol{f})}$ and up to $Sig(\boldsymbol{f})$ where $Gen(\boldsymbol{m}) = |\boldsymbol{G}_{Sig(\boldsymbol{f})}|$. Then $\boldsymbol{m'} := Update(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{G}, \boldsymbol{M}) \in \overline{\boldsymbol{m}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$.*

Proof. If $Gen(\boldsymbol{m}) = |\boldsymbol{G}|$ and $Flag(\boldsymbol{m}) \geq \boldsymbol{s}$, the result follows immediately by the discussion above. If $Gen(\boldsymbol{m}) = |\boldsymbol{G}|$ but $Flag(\boldsymbol{m}) < \boldsymbol{s}$, we call a normal M5GB-reduction on $\boldsymbol{m}$ and hence, correctness follows as well. If $Gen(\boldsymbol{m}) \neq |\boldsymbol{G}|$, the candidates which might reduce $\boldsymbol{m}$ are in $\boldsymbol{G} \backslash \boldsymbol{G}_{Sig(\boldsymbol{f})}$. Since we do the same steps as in M5GB-tail-reduce, the returned element is Sig-tail-irreducible with respect to $\boldsymbol{G} \backslash \boldsymbol{G}_{Sig(\boldsymbol{f})}$ and up to $\boldsymbol{s}$. Furthermore, $\boldsymbol{m}$ is Sig-tail-irreducible with respect to $\boldsymbol{G}$ and up to $Sig(\boldsymbol{f})$. If the old flag is greater or equal to $\boldsymbol{s}$, it is Sig-tail-irreducible with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$. Hence, updating the flag suffices. Otherwise, we call a normal M5GB-reduction on the result again, and thus, correctness follows. $\square$

4.4.2 Fast irreducibility checks

Note that an implementation of the M5GB Algorithm so far would spend a lot of time to check whether some $t \in T(f)$ is Sig-reducible up to some signature $\boldsymbol{s}$ or not. If we find some element in $\boldsymbol{M}$ with leading term t, we can speed this up, but otherwise, we need to check for Sig-reducibility in $\boldsymbol{G}$ which is costly. Hence, our idea is to save those currently Sig-irreducible terms in a set $Irr \subseteq T$ if once characterized as Sig-irreducible. Similar to Sig-tail-irreducibility for some element in $\boldsymbol{m} \in \boldsymbol{M}$, those (at the moment) Sig-irreducible terms might become Sig-reducible again during the algorithm. Equipping all terms in Irr with generation and flag, applying the same updates as described in Section 4.4.1, solves this problem. The advantages are the following:

1. In the process of $M5GB$-reducing some $\boldsymbol{f}$, we can immediately skip the examination of all $t \in T(f) \cap Irr$ as long as the elements in Irr are updated. This update only checks for Sig-reducibility by $\boldsymbol{g} \in \boldsymbol{G} \backslash \boldsymbol{G}_{\boldsymbol{s}}$, where $Gen(t) = |\boldsymbol{G}_{\boldsymbol{s}}|$, instead of Sig-reducibility by the whole set $\boldsymbol{G}$. This saves a lot of time since we do not have to check again whether some $\boldsymbol{g} \in \boldsymbol{G}_{\boldsymbol{s}}$ Sig-reduces t. Obviously, the same improvement works for $M5GB$-Tail-reduce and $t \in T(Tail(f)) \cap Irr$.
2. We can update $Flag(\boldsymbol{m})$ for $\boldsymbol{m} \in \boldsymbol{M}$ even faster: This can be done since

$$Flag(\boldsymbol{m}) = \min\{Flag(t) : t \in T(Tail(m))\}$$

follows from its definition.

Remark 4.4.6. *Improvement 4.4.2 consumes additional storage, but reduces the computational effort significantly.*

4.4.3 Labelled polynomial optimization

Similar to the discussion in Section 3.3.2, we do not want to save the whole module element $\boldsymbol{f}$ explicitly, but $\mathcal{F} = (Sig(\boldsymbol{f}), v(\boldsymbol{f}))$. We have already seen in 3.3.2 how we can adapt this for the overall routine without losing any needed property. This worked well in Algorithm 12 because we only did regular reductions there and hence, the signature stayed the same during the reduction step. Since we do the same for elements in $\boldsymbol{G}$, we can guarantee this for M5GB-reduction too. The problem arises when we consider M5GB-Tail-reduce: Assume we reduce some element $\boldsymbol{f} \in P^m$ up to some signature $\boldsymbol{s}$, which is larger than $Sig(\boldsymbol{f})$. Then, we get $\boldsymbol{f'} \in \overline{\boldsymbol{f}}^{\boldsymbol{G},\boldsymbol{s}}$ which might have $Sig(\boldsymbol{f'}) > Sig(\boldsymbol{f})$. The first idea is to do the following: If we want to reduce $\mathcal{F}$ by some $v\mathcal{G}$, we define

$$Sig(\mathcal{F} - v\mathcal{G}) := \max\{Sig(\mathcal{F}), Sig(v\mathcal{G})\}.$$

This works well unless the reduction is singular. In that case, this computation might not be longer correct, because by the definition above, we possibly end up with $Sig(\boldsymbol{f} - v\boldsymbol{g}) < Sig(\mathcal{F} - v\mathcal{G})$. The fact we can guarantee is that $Sig(\mathcal{F} - v\mathcal{G})$ is an upper bound for $Sig(\boldsymbol{f} - v\boldsymbol{g})$ in that way. In particular, if $\mathcal{F}' = M5GB\text{-}Tail\text{-}reduce(\mathcal{F}, G, \boldsymbol{s})$, we have $Sig(\boldsymbol{f'}) \leq Sig(\mathcal{F}') < \boldsymbol{s}$. Such a $\mathcal{F}'$ does no longer fulfill the properties of a labelled polynomial as defined in Definition 3.3.4, leading to the following weaker definition:

Definition 4.4.7 (Over-labelled polynomials). *We call a pair $\mathcal{F} = (\boldsymbol{s}, f)$ with $\boldsymbol{s} \in T_m$, $f \in P$ an over-labelled polynomial if there exists $\boldsymbol{f} \in P^m$:*

(i) $v(\boldsymbol{f}) = f$.

(ii) $MLT(\boldsymbol{f}) \leq \boldsymbol{s}$.

We set $poly(\mathcal{F}) := f$ and $Sig(\mathcal{F}) := \boldsymbol{s}$. To define this upper bound described above by $Sig^+(\boldsymbol{f})$, we apply following rules:

(i) For initial polynomials $f_1, ..., f_m$, we set for all $i \in \{1, ..., m\}$

$$Sig^+(\boldsymbol{f_i}) := Sig(\boldsymbol{f_i}).$$

(ii) For $c \in \mathbb{K}, u \in T, \boldsymbol{f} \in P^m$, we define

$$Sig^+(cu\boldsymbol{f}) := cuSig^+(\boldsymbol{f}).$$

(iii) For $\boldsymbol{f}, \boldsymbol{g} \in P^m$, we define

$$Sig^+(\boldsymbol{f} + \boldsymbol{g}) = \max\{Sig^+(\boldsymbol{f}), Sig^+(\boldsymbol{g})\}.$$

If $Sig^+(\boldsymbol{f}) \sim Sig^+(\boldsymbol{g})$, we can choose $Sig^+(\boldsymbol{f})$ or $Sig^+(\boldsymbol{g})$ arbitrarily.

Remark 4.4.8. *It is straightforward to see that $\mathcal{F} := (Sig^+(\boldsymbol{f}), f)$ is an over-labelled polynomial. This is the structure over-labelled polynomials will always have in the M5GB algorithm.*

At the first sight, working with over-labelled polynomials might seem to destroy the correctness, but in fact, it does not. To see that, observe that $\boldsymbol{G}$ still consists of labelled polynomials and only $\boldsymbol{M}$ has over-labelled ones. For M5GB-reduce, this means that we might not be allowed to reduce some $\mathcal{F} \in \boldsymbol{G}$ by some $\mathcal{M} \in \boldsymbol{M}$, although $Sig(\boldsymbol{f}) > Sig(\boldsymbol{m})$, but $Sig(\mathcal{F}) \leq Sig(\mathcal{M})$. Nevertheless, by the construction of M, there exist $\boldsymbol{g} \in \boldsymbol{G}, v \in T$ such that

$$LT(vg) = LT(m), \quad Sig(v\boldsymbol{g}) < Sig(\boldsymbol{f}).$$

Hence, a Sig-reducer for $\mathcal{F}$ exists if and only if $\boldsymbol{f}$ is Sig-reducible. In particular, the element $\mathcal{F}'$ computed by labelled polynomial reduction is regularly irreducible as its module equivalent. Observe that the same argumentation can be applied analogously for M5GB-Tail-reduce. This might lead to a small decrease in straightforward reductions and, therefore, to more computational effort. Nevertheless, the advantage of less storage and faster additions of (over-)labelled polynomials in comparison to module elements outweigh this by far. Note that working with over-labelled polynomials helps for the following improvement as well.

4.4.4 Check divisibility by M

Consider the following scenario: We want to M5GB-reduce some $t \in T_{\boldsymbol{G}}(f, Sig(\boldsymbol{f}))$ for $\boldsymbol{f}$, but there exists no $\boldsymbol{m} \in \boldsymbol{M}$ such that $LT(m) = t$. In the basic version of the algorithm, we would call Tail-M5GB-reduce on $v\boldsymbol{g}$ where $\boldsymbol{g} \in \boldsymbol{G}, v \in T, LT(vg) = t$. But assume there exists $\boldsymbol{m} \in \boldsymbol{M}, u \in T : LT(um) = t$. Since $\boldsymbol{m}$ was at some time Sig-tail-irreducible, it seems to be a better choice to call Tail-M5GB-reduce on $u\boldsymbol{m}$ because one expects less Sig-reducible terms in $u\boldsymbol{m}$ than in some $v\boldsymbol{g}, v \in T, \boldsymbol{g} \in \boldsymbol{G}$. Note that this idea was already considered for M4GB by [20]. The following lemma tells us when we are allowed to take such an $\boldsymbol{m} \in \boldsymbol{M}$ without threatening correctness:

Lemma 4.4.9. *Let $\boldsymbol{m} \in \overline{v\boldsymbol{g}}^{Tail,\boldsymbol{G},\boldsymbol{t}}$ where $v \in T, \boldsymbol{g} \in \boldsymbol{G}, \boldsymbol{t} \in T_m$. Define $\boldsymbol{r} := Sig^+(\boldsymbol{m})$ and let $u \in T, \boldsymbol{s} \in T_m$ such that $\boldsymbol{s} \geq u\boldsymbol{r}$. Then*

$$\overline{u\boldsymbol{m}}^{Tail,\boldsymbol{G},\boldsymbol{s}} \subseteq \overline{uv\boldsymbol{g}}^{Tail,\boldsymbol{G},\boldsymbol{s}}.$$

Proof. By definition of $\boldsymbol{r}$, we have that $\boldsymbol{m} \in \overline{v\boldsymbol{g}}^{Tail,\boldsymbol{G},\boldsymbol{r}}$. Moreover, note that

$$v\boldsymbol{g} \xrightarrow[\boldsymbol{G},\boldsymbol{r}]{*} \boldsymbol{m}, LT(vg) = LT(m) \Rightarrow uv\boldsymbol{g} \xrightarrow[\boldsymbol{G},u\boldsymbol{r}]{*} u\boldsymbol{m}, LT(uvg) = LT(um).$$

Since $\boldsymbol{s} \geq u\boldsymbol{r}$, this implies

$$uv\boldsymbol{g} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} u\boldsymbol{m}, \quad LT(uvg) = LT(um).$$

Taking an arbitrary $\boldsymbol{m'} \in \overline{\boldsymbol{um}}^{Tail,\boldsymbol{G},\boldsymbol{s}}$, we get

$$u v \boldsymbol{g} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} u \boldsymbol{m} \xrightarrow[\boldsymbol{G},\boldsymbol{s}]{*} \boldsymbol{m'}, \quad LM(uv\boldsymbol{g}) = LM(\overline{\boldsymbol{um}}^{Tail,\boldsymbol{G},\boldsymbol{s}}),$$

which proves the statement. □

Remark 4.4.10. *At the first sight, it seems unnecessary to take $Sig^+(\boldsymbol{m})$ instead of $Sig(\boldsymbol{m})$, but if $Sig(\boldsymbol{m}) < \boldsymbol{t} < Sig(v\boldsymbol{g})$, the statement does no longer hold. Since by construction $\boldsymbol{r} \leq \boldsymbol{t}$, we could take $\boldsymbol{t}$ instead of $\boldsymbol{r}$. But as this $\boldsymbol{r}$ might be a strict lower bound, we proved a stronger statement which enables us to apply the improvement discussed above more often.*

If there is more than one element in $\boldsymbol{M}$ with that property, we look for an element that seems to be the most promising. For that, we can order the elements in $\boldsymbol{M}$ by different heuristics. Some suggestions are:

- Take the $\boldsymbol{m}$ with the largest generation.
- Take the $\boldsymbol{m}$ with the largest signature flag.
- Take the $\boldsymbol{m}$ with the largest leading term.
- Take the $\boldsymbol{m}$ with the smallest number of terms.
- Take the $\boldsymbol{m}$ with the smallest remainder (in notation from above, smallest u).

Approaches (i),(ii) and (iv) seek $\boldsymbol{m}$ itself to contain the least Sig-reducible terms and hence, $u\boldsymbol{m}$ tends to have few Sig-reducible terms. (iii) and (v) minimize u, hoping to obtain few Sig-reducible terms for $u\boldsymbol{m}$ directly. To decide which heuristic is the best one or if simply taking the first element appearing, skipping this comparing process, is faster, needs to be tested in actual implementations.

Remark 4.4.11. *Note that if some $t \in T$ is Sig-reducible by $\boldsymbol{M}$ (with signatures $Sig(\mathcal{M})$) and up to signature $\boldsymbol{s}$, it is Sig-reducible by $\boldsymbol{G}$ up to signature $\boldsymbol{s}$ as well. Hence, one can check first if this t is reducible at all by looking for a Sig-reducer in $\boldsymbol{G}$ before selecting a Sig-reducer in $\boldsymbol{M}$. For some irreducible t (which is not yet marked as irreducible), this is an improvement since $\boldsymbol{G}$ tends to be much smaller than $\boldsymbol{M}$. To further speed up this process, create a map where each $\boldsymbol{m} \in \boldsymbol{M}$ is mapped to the $\boldsymbol{g} \in \boldsymbol{G}$ which originally created $\boldsymbol{m}$. This helps to find those candidates in $\boldsymbol{M}$ faster since $LT(g)|LT(m)$ still holds.*

4.4.5 Taking the best element with current signature

Lemma 3.2.7 gives us the following optimization choice: Given an S-polynomial $\boldsymbol{p}$ of minimal signature $\boldsymbol{s}$, we could choose any module element $\boldsymbol{f}$ with the same signature and reduce this one instead. In Section 3.4, we exploited this to search in

$$\{v\boldsymbol{g} : v \in T, \boldsymbol{g} \in \boldsymbol{G} : Sig(v\boldsymbol{g}) \sim \boldsymbol{s}\}$$

for a representative that is easy to reduce by rewrite orders induced by heuristics similar to those in Section 4.4.4. Note that we have more freedom for this heuristics than for the rewrite order since we do not have to fulfill

$$\forall \boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G} : Sig(\boldsymbol{f}) \mid Sig(\boldsymbol{g}) \Rightarrow \boldsymbol{f} \leq \boldsymbol{g}.$$

We can extend the search for such an element further to $\boldsymbol{M}$ and go through

$$\{v\boldsymbol{m} : v \in T, \boldsymbol{m} \in \boldsymbol{M} : Sig(v\boldsymbol{m}) \sim \boldsymbol{s}\} \tag{4.2}$$

to find the best representative. Since the elements in $\boldsymbol{M}$ tend to be far reduced, this might reduce the number of reduction steps even further, although checking for such an element might cost too much time.

Remark 4.4.12. *Using the labelled polynomial optimization, M contains over-labelled polynomials, thus, taking those elements in general destroys the correctness of the algorithm. One could check at the construction/update of some $\boldsymbol{m} \in \boldsymbol{M}$ whether the element is indeed a labelled polynomial. This can be done by checking at each reduction step whether it is singular. If not, $\boldsymbol{m}$ is still a labelled polynomial. In our implementation, we examined the whole set $\boldsymbol{M}$ and decided to ignore this fact. In all tested cases, we ended up with a correct Gröbner basis output, although this can not be guaranteed for all inputs.*

4.5 Implementation and results

We did a small test implementation of our proposed M5GB Algorithm in C++ which can be found on [3]. Due to the restricted time scope of this thesis, we concentrated on correctness and implementation of the improvements, but not on computation time efficiency due to storage management and optimized sub-routines. We decided to implement the algorithm with term order $<_{grevlex}$ and order extension $\boldsymbol{<}_w$ as defined in 3.1.2. It was chosen in that way since $<_{grevlex}$ seems to be the best choice for generic Gröbner basis computations in basically all modern algorithms and $\boldsymbol{<}_w$ was empirically tested to be the best in [18]. Actually, we compared it with extension $\boldsymbol{<}_{pot}$, but this led to a much worse result. Considering the rewrite order, we both implemented ratio $<_{rat}$ and $<_{num}$ as they are the best rewrite orders known so far. Since M4GB is one of the most modern algorithms and similar to M5GB, we compared our implementation with the original implementation from [2] and chose some small examples from the Fukuoka MQ challenges [1]. For the implementation of the M5GB Algorithm, we considered all five improvements discussed in the last section. As already mentioned, we did not concentrate on time efficiency, hence, we counted operation steps rather than actual time consumption. To compare, we concentrated on three main counters:

1. The number of polynomials which are (full-)reduced. In M5GB this equals the number of critical pairs which are indeed reduced, in M4GB this equals twice the number of critical

pairs examined since uf and vg are separately reduced for $spol(f, g) = uf - vg$.

2. The number of reduction steps in Full-reduce. This counts all steps in full-reductions where some term is reduced.

3. The number of overall reduction steps: This counts all reduction steps combined, i.e. reduction steps in Full-reduce, reduction steps in Tail-reduce, reduction of the basis before and after the algorithm, updating polynomials throughout the algorithm, ...

We mainly considered the MQ challenges of Type I and Type IV, meaning we concentrated on the case $p = 2$. We did this on purpose since signature algorithms tend to work quite well on small fields due to the additional syzygies coming from the field equations. At first, we tested the different heuristics described above against each other to get appropriate options to compete with M4GB. Note that we used the rewrite order $<_{rat}$ for the first tests. For documentation in this thesis, we chose the following ideals from [1] as benchmarks:

(i) $n = 15$, Type I, Seed 0.

(ii) $n = 10$, Type IV, Seed 0.

In the following lines, we will refer quite often to two certain improvements described above. We will call them shortly "divisibility-improvement" for the one considered in Section 4.4.4, respectively "best-element-improvement" for the one considered in Section 4.4.5. We considered the following heuristics at the same time for those improvements:

- Take the module element with the largest generation.
- Take the module element with the largest signature flag.
- Take the module element with the largest leading term.
- Take the module element with the minimal number of terms.
- Take the module element with the smallest remainder.

This led for $n = 15$, Type I, Seed 0 to the following table (green indicates the best value in each counter):

Heuristic	critical pairs	full reductions	total reductions	zero reductions
No optimization	1135	75274	238666	854
Largest generation	1135	75021	239073	854
Largest signature flag	1135	75137	241545	854
Largest leading term	1134	75277	246210	853
Smallest leading term	1132	75028	230315	851
Minimal number of terms	1135	42617	349891	854
Smallest remainder	1134	42694	422661	853

Remark 4.5.1.

1. *The divisibility-improvement decreases the number of total reductions, the best-element-improvement decreases the number of full reductions (and hence, indirectly affects the number of total reductions).*

2. *It makes no sense to use the largest flag heuristic or largest leading term for the best-element-improvement, hence, those heuristics are applied only on the divisibility-improvement in the table above.*

3. *Largest leading term and smallest remainder naturally coincide for the divisibility-improvement.*

This test (repeated with similar results for other inputs) suggests that minimal number of terms and smallest remainder tend to be the most promising heuristics for the divisibility-improvement, but are bad for the best-element-improvement. Hence, we tested other heuristics for the best-element-improvement again, keeping the heuristic for the divisibility-improvement fixed. The best options we could find was minimal number of terms combined with smallest leading term. We tested this setting with rewrite order $<_{num}$ and got slightly worse values (as before, tested with several inputs with similar results):

Rewrite order	critical pairs	full reductions	total reductions	zero reductions
$<_{rat}$	1135	42789	340712	854
$<_{num}$	1136	42820	340751	855

To compete against M4GB, we considered smallest leading term in both improvements to minimize the number of total reductions and minimum number of terms combined with smallest leading term to minimize the number of full reductions. For the sake of shortness, we denote these two variants from now on simply by "Smallest LT" and "Number of Terms". We can see that M4GB outperforms those variants for $n = 15$, Type I, Seed 0, by far:

Rewrite order	critical pairs	full reductions	total reductions	zero reductions
Number of terms	1135	42789	340712	854
Smallest LT	1132	75028	230315	851
M4GB	818	30924	155275	No information

This is at least partly due to the property that we chose a much overdetermined system ($n = 15, m = 30$, with added field equations even $m = 45$). In such a system, most reductions computed are still unnecessary since such a system is very far away from being regular, but there is no possibility known to determine these zero reductions in advance. Note that we empirically observed the following for this Type I equations: Once the first zero reduction occurs, every reduction turns out to be a zero reduction. Therefore, we tested the program to stop computing new S-polynomials as soon as the first zero reduction occurs. For all tested inputs of Type I, we obtained with that "cut" algorithm the desired Gröbner basis. This algorithm, which is of course no longer guaranteed to be correct in general, leads to nice numbers:

Algorithm	critical pairs	full reductions	total reductions	zero reductions
Cut Number of terms	281	11311	159749	1
Cut Smallest LT	281	18224	113160	1
M4GB	818	30924	155275	No information

We tested this approach on a slightly larger example as well, getting a worse result than for M4GB again: For $n = 20, m = 40$, Seed 0:

Algorithm	critical pairs	full reductions	total reductions	zero reductions
Cut Number of terms	1307	126218	3075908	1
Cut Smallest LT	1307	338013	2480574	1
M4GB	972	152596	990054	No information

Since the first zero reduction will definitely not work for all ideals, we propose to test every $k \in \mathbb{N}$ zero reduction steps or every $l \in \mathbb{N}$ reduction steps whether the current $v(\boldsymbol{G})$ already is a Gröbner basis. In these overdetermined cases of Type I, a reduced Gröbner basis has n elements with degree 1 for all tested inputs. Hence, reducing $v(\boldsymbol{G})$ (without signatures) and checking if we obtain such n elements, might cut down the computation time a lot. Another possibility would be to change to a non-signature algorithm as soon as a certain signature bound is reached. Then we can interreduce $v(\boldsymbol{G})$ and if we are not yet done, we can reduce the remaining S-polynomials quite fast since the elements in the reduced Gröbner basis candidate tend to be of low degree.
For underdetermined systems, we expected the original M5GB algorithm to work better since only few zero reductions should be computed. For that reason, we tested Type IV, Seed 0, $n = 10$, which is an underdetermined system with $m = 7$. In the following table, "with M" denotes the idea of considering the set defined in (4.2). Indeed, M5GB seems to be competitive for underdetermined systems:

Algorithm	critical pairs	full reductions	total reductions	zero reductions
Number of terms/smallest LT	232	5260	100413	74
With M	208	3963	92971	74
Smallest LT/Smallest LT	200	7320	33572	41
M4GB	351	5817	35768	No information

To make sure this was not a coincidence, we tested the other seeds of the same type as well, showing that M5GB performs great in those settings:
Seed1:

Algorithm	critical pairs	full reductions	total reductions	zero reductions
Number of terms/smallest LT	243	5554	98943	90
With M	220	4273	93251	74
Smallest LT/Smallest LT	235	8051	36830	41
M4GB	395	6255	37385	No information

Seed2:

Algorithm	critical pairs	full reductions	total reductions	zero reductions
Number of terms/smallest LT	264	5546	75095	90
with M	248	4481	102959	91
Smallest LT/Smallest LT	259	8918	40563	41
M4GB	324	5995	35875	No information

Seed3:

Algorithm	critical pairs	full reductions	total reductions	zero reductions
Number of terms/smallest LT	212	4478	84420	90
with M	191	3701	78015	34
Smallest LT/Smallest LT	222	7999	40471	65
M4GB	393	6126	36405	No information

4.5.1 Implementation details and possible improvements

As pointed out earlier, our implementation is not time-optimized in many different areas. We left out the following aspects completely which should be possible to implement quite easily:

1. The Update function: Using a complete tail-reduction on some $\boldsymbol{m} \in \boldsymbol{M}$ when either the signature flag condition or the generation condition is not fulfilled leads to lots of unnecessary computations. We described in Section 4.4.1 how one can overcome this. Nevertheless, since this optimization does not decrease the number of reduction steps itself, it was left out by our implementation.

2. Suitable data structures: We did not optimize the used data structures. Therefore, it should be easy to save a lot of time there. To speed up the process of searching for a divisor in $\boldsymbol{G}$, one could save already computed multiples or implement some fast check whether some $\boldsymbol{g}$ does not divide some module element $\boldsymbol{f}$. Similarly for $\boldsymbol{M}$, as discussed earlier, one could create a map that maps each $\boldsymbol{m} \in \boldsymbol{M}$ to the element $\boldsymbol{g} \in \boldsymbol{G}$ which was used to create $\boldsymbol{m}$. Testing for divisibility with that $\boldsymbol{m}$ is only needed to be computed when that corresponding $LT(g)$ divides the considered term as well.

3. Fast field operations: We did not optimize the operations done on single polynomials, one needs to optimize the addition of polynomials, the multiplication of terms (see below) and the computation of field inverses.

4. The encoding of terms: This encoding should be optimized to do the following operations for $t, u \in T$ very quickly:

   - Check if $t < u$.
   - Check if $t \mid u$.
   - Compute $t \cdot u$.

   The data structure we chose to do that is a large lookup table where each term t, up to some degree d as degree bound, is encoded once as an exponent vector and once as an integer k such that t is the k^{th} smallest term with respect to $<_{grevlex}$. The idea is to check divisibility via the exponent vector, the comparison operator via the integer representation. For the multiplication of terms, we constructed a separate multiplication table where for each term t encoded as an integer, we saved the n different integers corresponding to $t \cdot x_i, i \in \{1, ..., n\}$. In that way, the multiplication of $t \cdot u$ takes exactly $deg(u)$ table lookups. Unfortunately, the construction of such a table is quite time and storage consuming, since these tables have size $\mathcal{O}(n \cdot \binom{n+d}{n})$. For large n, this might be an issue and worth to overthink.

4.6 Conclusion and future work

In this thesis, we started with an application from the cryptographic area that is using Gröbner bases to emphasize their importance. Then we revised some characteristic facts about those bases and defined a new Gröbner basis equivalence (Lemma 2.1.5). We revised, in a short way, some commonly used Gröbner basis algorithms, namely F4 and FGLM. Additionally, we described a modification of M4GB which is easy to understand, but contains all the main ideas of this algorithm.

In the following course of this thesis, we summed up, in a mathematically precise manner, the theory of signature and rewrite Gröbner basis algorithms and revised that F5 can be seen as a special case of the latter. Doing that, we closed some small proof gaps (e.g. Proposition 3.2.3, Theorem 3.3.9, Lemma 3.3.10 and most of Lemma 3.4.18). In addition, we proved some statements more rigorously than the original authors do, e.g. Theorem 3.4.20 and found different proofs of already shown statements (e.g. Lemma 3.5.2, Theorem 3.5.5).

Lastly, we proposed with M5GB a new hybrid approach between rewrite basis algorithms and M4GB. For that, we generalized the theory of rewrite bases and signature reductions, came up with many new statements and proved them. We showed the correctness of this new algorithm and in this context, main parts of the correctness proof of M4GB, which are missing in the original paper. We implemented this new algorithm and compared it with M4GB. It turned

out that the algorithm can most likely compete with M4GB when implemented more efficiently, especially for underdetermined systems and small prime fields. For largely overdetermined systems, we proposed a heuristic to adapt M5GB to work well.
To determine the practicability of M5GB, one needs a more efficient implementation, considering the suggestions from Section 4.5.1 as well as additional computer science tools to improve the actual running time. From the mathematical point of view, one could look for new criteria to detect reductions to zero. This would be highly advantageous since using the most suitable order extension and overdetermined systems still leads to many unnecessary reductions.

Bibliography

[1] Fukuoka MQ challenge. https://www.mqchallenge.org/. Accessed: 2020-06-29.

[2] Github - M4GB Implementation. https://github.com/cr-marcstevens/m4gb/. Accessed: 2020-08-27.

[3] Github - M5GB basic Implementation. https://github.com/manschga/M5GB. Accessed: 2020-08-31.

[4] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 01 / 1965.

[5] David Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. Springer New York, 03 / 1998.

[6] David A. Cox. *Ideals, varieties and algorithms*. Springer International PU, Second edition, 2016.

[7] Christian Eder. Predicting zero reductions in Gröbner basis computations. In *Proceedings of the 2014 Symposium on Symbolic-Numeric Computation*, page 109–110, New York, NY, USA, 2014. Association for Computing Machinery.

[8] Christian Eder and Jean-Charles Faugère. A survey on signature-based algorithms for computing Gröbner bases. *Journal of Symbolic Computation*, 80:719 – 784, 2017.

[9] Christian Eder and John Perry. F5C: A variant of Faugère's F5 algorithm with reduced Gröbner bases. *Journal of Symbolic Computation*, 45(12):1442 – 1458, 2010.

[10] Christian Eder and Bjarke Roune. Signature rewriting in Gröbner basis computation. *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 331–338, 06 / 2013.

[11] David Eisenbud. *Commutative Algebra - with a view toward Algebraic Geometry*. Springer Science & Business Media, Berlin Heidelberg, 2013.

[12] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). *Proceedings of the 2002 international symposium on Symbolic and algebraic computation - ISSAC 02*, 2002.

[13] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.*, 16:329–344, 10 / 1993.

[14] Jean-Charles Faugère and Ludovic Perret. On the security of UOV. *IACR Cryptology ePrint Archive*, 2009:483, 01 / 2009.

[15] Jean-Charles Faugére. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[16] Alessandro Giovini, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. "One Sugar cube, Please" or selection strategies in the Buchberger Algorithm. In *ISSAC '91: Proceedings of the 1991 international symposium on Symbolic and algebraic computation*, pages 49–54, 01 / 1991.

[17] Bjarke Hammersholt Roune and Michael Stillman. Practical Groebner Basis Computation. *arXiv e-prints*, page arXiv:1206.6940, 06 / 2012.

[18] Amir Hashemi and Gwénolé Ars. Extended F5 criteria. *Journal of Symbolic Computation*, 45(12):1330 – 1340, 2010. MEGA'2009.

[19] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592, pages 206–222, 04 / 1999.

[20] Rusydi H. Makarim and Marc Stevens. M4GB: An efficient Gröbner-Basis Algorithm. In *ISSAC '17: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 293–300, 07 / 2017.

[21] Ernst W. Mayr. Some complexity results for polynomial ideals. *Journal of Complexity*, 13(3):303 – 325, 1997.

[22] A.J.M Segers. Algebraic attacks from a Gröbner basis perspective. Master's thesis, University of Technology Eindhoven, 01 / 2004.

[23] Till Stegers. Faugere's F5 algorithm revisited. Master's thesis, University of Technology Darmstadt, 09 / 2005.

[24] Alan Szepieniec. *Mathematical and Provable Security Aspects of Post-Quantum Cryptography.* PhD thesis, KU Leuven, 12 / 2018.

[25] Volker Weispfenning Thomas Becker. *Gröbner Bases, A Computational Approach to Commutative Algebra.* Springer, New York, NY, 1993.

[26] Dingkang Wang. The F5 algorithm in Buchberger's style. *Journal of Systems Science and Complexity*, 24, 06 / 2010.