JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Clemens Hofstadler

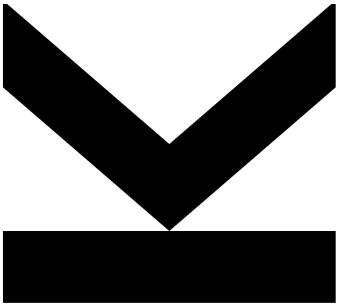
Submitted at
Institute for Algebra

Supervisor
Priv.-Doz. Dr. Georg Regensburger

Co-Supervisor
Dr. Clemens Raab

March 2020

Certifying operator identities and ideal membership of noncommutative polynomials



Master Thesis
to obtain the academic degree of
Diplom-Ingenieur
in the Master's Program
Computer Mathematics

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

Typically, operator identities are proven by a formal computation using known identities. In principle, one has to make sure that every step of such a computation respects the restrictions imposed by the domains and codomains of the operators involved. In this thesis, we present a recently developed framework that allows us to prove operator identities in a fully algebraic fashion without having to inspect every part of the computation. To this end, we model operator identities by noncommutative polynomials and encode their domains and codomains in a labelled quiver. It turns out that, if the polynomial f associated to a claimed operator identity as well as all polynomials in the set F corresponding to assumed identities are compatible with such a quiver, proving the claim reduces to showing that f is contained in the ideal generated by F. The compatibility condition on f and the polynomials in F ensures that these polynomials respect the restrictions imposed by the domains and codomains of the operators involved and, provided that the ideal membership can be verified, in fact implies the claimed identity for all settings in which it can be phrased. To show ideal membership, the theory of Gröbner bases can be utilised. In contrast to the commutative case, where Gröbner bases can be used to decide this problem, ideal membership in the noncommutative case is undecidable in general. Nevertheless, there exist generalisations of Buchberger's algorithm and of Faugère's F4 algorithm to the noncommutative setting, that often allow us to verify ideal membership in practice.

To be able to state these algorithms, we recall the main results of the theory of noncommutative Gröbner bases. To this end, we adopt the point of view of Bergman and consider polynomial reduction as a linear map. We also discuss several optimisation strategies to improve the performance of these algorithms in practice, such as deletion criteria and redundant generator reduction. Additionally, we extend both algorithms to keep track of cofactors during the computation. Provided that we can verify the membership of a given polynomial in an ideal, this enables us to produce a certificate for this ideal membership, which can then be checked independently. Finally, we introduce the software package `OperatorGB`, which is available for MATHEMATICA and SAGEMATH and, besides automated compatibility checks, implements certified ideal membership of noncommutative polynomials via (partial) Gröbner bases computations. We also provide insight into some implementation details of this package and compare it to other packages that provide functionality for noncommutative Gröbner bases computations.

Zusammenfassung

Operatoridentitäten werden üblicherweise ausgehend von bekannten Identitäten mit Hilfe einer formalen Rechnung bewiesen. Theoretisch muss dabei sichergestellt werden, dass in jedem Schritt dieser Rechnung Einschränkungen, die sich durch die Definitions- und Wertebereiche der involvierten Operatoren ergeben, berücksichtigt werden. In dieser Arbeit wird eine kürzlich entwickelte Theorie vorgestellt, die es erlaubt, Operatoridentitäten rein algebraisch zu beweisen, ohne dabei jeden Teil der Rechnung inspizieren zu müssen. Dazu werden Operatoridentitäten mit Hilfe von nichtkommutativen Polynomen modelliert und die Definitions- und Wertebereiche der auftretenden Operatoren in Form eines gerichteten Multigraphen kodiert. Vorausgesetzt, dass das Polynom f, welches einer zu beweisenden Operatoridentität entspricht, sowie die Polynome der Menge F, welche den übersetzten Annahmen entsprechen, mit solch einem Graphen kompatibel sind, reduziert sich damit der Beweis der entsprechenden Aussage über Operatoren auf das Verifizieren, dass f in dem von F erzeugten Ideal enthalten ist. Die Kompatibilitätsanforderung an f und die Polynome in F stellt sicher, dass diese Polynome den Restriktionen entsprechen, die sich durch die Definitions- und Wertebereiche der involvierten Operatoren ergeben und beweist, wenn zusätzlich die Idealzugehörigkeit verifiziert werden kann, die behauptete Operatoridentität sogar für alle Situationen, in denen solch eine Aussage formuliert werden kann. Um Idealzugehörigkeit zu beweisen, kann die Theorie der Gröbnerbasen verwendet werden. Im Gegensatz zum kommutativen Fall, wo dieses Problem mit Hilfe von Gröbnerbasen entschieden werden kann, ist das Idealzugehörigkeitsproblem im Kontext von nichtkommutativen Polynomen im Allgemeinen unentscheidbar. Trotzdem gibt es Verallgemeinerungen des Buchberger Algorithmus sowie von Faugères F4 Algorithmus für den nichtkommutativen Fall, welche es in der Praxis oft erlauben Idealzugehörigkeit zu verifizieren.

Um diese Algorithmen formulieren zu können, werden die wichtigsten Resultate der Theorie nichtkommutativer Gröbnerbasen wiedergegeben. Dazu wird die Sichtweise von Bergman übernommen und polynomielle Reduktion als lineare Abbildung betrachtet. Weiters werden verschiedene Optimierungsstrategien diskutiert, welche die Laufzeit dieser Algorithmen in der Praxis verringern sollen. Außerdem werden beide Algorithmen so erweitert, dass während der Berechnung auch Kofaktoren mitprotokolliert werden können. Dies erlaubt es, vorausgesetzt, dass die Zugehörigkeit eines gegebenen Polynoms in einem Ideal gezeigt werden kann, ein Zertifikat für diese Idealzugehörigkeit zu produzieren, welches dann unabhängig überprüft werden kann. Abschließend wird das Softwarepaket `OperatorGB` vorgestellt, welches für MATHEMATICA sowie SAGEMATH verfügbar ist und neben automatisierten Kompatibilitätstests Funktionalität bietet, um Zertifikate für Idealzugehörigkeit nichtkommutativer Polynome mittels (partieller) Gröbnerbasen zu berechnen. Es werden auch Einblicke in einige Details der Implementierung der Software gegeben und das Paket wird mit anderen Paketen verglichen, welche ebenfalls Funktionalität zur Berechnung nichtkommutativer Gröbnerbasen bieten.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Priv.-Doz. Dr. Georg Regensburger and my co-supervisor Dr. Clemens Raab, who not only sparked my interest in the topics of noncommutative Gröbner bases and formal proofs with their enthusiastic manner but who also continuously supported and guided me with their sheer endless knowledge on these topics. I am very grateful for the numerous valuable discussions we had, both scientific and nonscientific, and for all the time they spent proofreading this thesis. Furthermore, I sincerely thank them for providing me the opportunity to join them on a research stay as well as for enabling me to become part of several publications.

Additionally, I would like to thank Cyrille Chenavier for proofreading parts of this thesis and his valuable remarks.

Finally, I want to express my sincerest thanks to my parents for supporting me throughout my whole life and to my girlfriend Kathi and my friends for their understanding and unconditional support over the last years.

This work was supported by the Austrian Science Fund (FWF): P 27229 and P 32301.

Clemens Hofstadler
Linz, March 2020

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.
Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

.....	.....
Ort, Datum	Clemens Hofstadler

Contents

Chapter 1

Introduction

Linear operators appear in many different areas of mathematics, either in form of matrices over finite-dimensional spaces or as operators over infinite-dimensional spaces. Often, one wants to use known properties of certain operators to infer new properties. Typically, these properties are expressible in terms of identities consisting of the operators involved and the arithmetic operations addition, composition and scaling. Then, proving that a claimed identity holds is usually done by a formal computation using the assumed identities. During such a computation one has to assure that all appearing expressions respect the restrictions imposed by the domains and codomains of the operators involved (respectively, respect the formats of the corresponding matrices). In other words, all intermediate sums and products must be valid operators (respectively matrices). Making sure that this is indeed the case can be quite tedious, especially when the computations involve many steps. So, additional properties, which only hold for a certain class of linear operators, are often used in order to keep the computations short. Examples of such properties are the existence of a finite rank or determinantal properties, which are useful to argue about matrices but do not necessarily carry over to operators over infinite-dimensional vector spaces. Hence, a proof using such particular properties is then only valid for a certain class of linear operators and an analogous statement for a larger class of operators requires a new proof.

In this thesis, we present the algebraic framework developed in [RRH19], which allows us to state a simple procedure, that on the one hand overcomes the problem of constant compatibility checks and on the other hand produces universally valid proofs. In particular, this framework enables us to ignore compatibility checks during the computation as long as the identities that we start with as well as the claimed identities only consist of valid operator expressions. Moreover, it ensures that, by a single formal computation, we can prove the claimed identities for all situations in which the assumptions and claims can be formulated (e.g., matrices of different sizes, bounded linear operators on Hilbert spaces, etc.). To this end, we model operator identities by noncommutative polynomials in some set X. Each indeterminate in X corresponds to a basic operator and composition of operators translates into multiplication of polynomials, where indeterminates still commute with coefficients but not with each other. Note that we have to use noncommutative polynomials as the composition of linear operators is also noncommutative in general. Additionally, the domains and codomains of the operators are encoded in a directed

multigraph Q, where edges have labels in X. The vertices of Q correspond to functional spaces and its edges correspond to basic operators between these spaces. We refer to such a graph as a labelled quiver. Each path in Q induces a monomial that corresponds to a valid operator. Consequently, polynomials whose monomials all share paths in Q with the same start and end can be translated into linear operators and are called compatible with Q. If the possibilities for start and end of corresponding paths are uniform for all its monomials, a polynomial is said to be uniformly compatible.

The algebraic framework presented, requires the polynomial f associated to a claimed identity to be compatible with the quiver Q encoding the domains and codomains of the operators involved and the polynomials in the set F, which correspond to the assumed identities, to be uniformly compatible with Q. If this is the case, proving that the claimed operator identity follows from the assumptions reduces to showing that f is contained in the ideal generated by F. Once this is shown, the claimed identity holds for all settings into which the polynomial f and the polynomials in F can be translated, using what is called a consistent representation of Q.

To verify ideal membership of noncommutative polynomials, we can utilise the theory of Gröbner bases. For commutative polynomials over a field, the notion of Gröbner bases was first introduced by Buchberger in 1965 [Buc65]. About a decade later, Bergman provided the main theoretical results to adapt these concepts to the noncommutative case [Ber78] by using the abstract concept of reduction systems, where he considers polynomial reduction as a linear map. In the commutative case, Gröbner bases allow us to decide ideal membership and can be computed for example by an algorithm proposed by Buchberger, now known as Buchberger's algorithm [Buc65], or by Faugère's F4 algorithm [Fau99].

Both of these algorithms have been generalised to the setting of noncommutative polynomials. However, in contrast to the commutative case, the noncommutative polynomial ring (in more than one variable) is not Noetherian, which renders the ideal membership problem of noncommutative polynomials undecidable in general. This is also why Gröbner bases computations in this ring need not terminate and we have to content ourselves with enumeration procedures. Nevertheless, in practice, we can often use the noncommutative analogues of the Buchberger algorithm and of the F4 algorithm to certify ideal membership. In this case, we can also obtain an explicit representation of the polynomial in question in terms of the generators of the ideal. This representation, which we call a cofactor representation, then serves as a certificate for the ideal membership and can be checked independently.

In the package `OperatorGB` [HRR19], we have implemented routines to efficiently certify ideal membership of noncommutative polynomials based on cofactor representations computed via the noncommutative analogues of the Buchberger algorithm and of the F4 algorithm. We note that at the time of writing this thesis, this is the only publicly available package that provides this functionality for arbitrary finitely generated ideals of noncommutative polynomials. Versions of the package for MATHEMATICA and SAGEMATH along with documentation and examples can be obtained at

`http://gregensburger.com/softw/OperatorGB`.

In particular, we also provide example files for most of the computations done in this thesis.

Furthermore, the package can also check compatibility of polynomials with quivers and almost fully automatically certify operator identities.

The goal of this thesis is to provide a self-contained reference work for the user of the software that contains all theoretical results needed to fully understand what the different methods of the package do as well as details about the implementation and a comparison of the package to similar software. Aside from the software package, the main contribution of this thesis to the topic of noncommutative Gröbner bases is a detailed discussion on how to trace cofactors in the noncommutative versions of the Buchberger algorithm and of the F4 algorithm and on how to use these cofactors to efficiently certify ideal membership. Additionally, we propose simple criteria to efficiently apply the chain criterion during noncommutative Gröbner bases computations. It is also worth mentioning that we present the theory of Gröbner bases adopting the point of view of Bergman and consider polynomial reduction as a linear map.

To keep this thesis self-contained, we recall the most important notions that are relevant for the subsequent chapters in Chapter 2, including the free monoid, the noncommutative polynomial ring and (two-sided) ideals. We also introduce the notion of cofactor representations.

Following upon that, we discuss Gröbner bases of noncommutative polynomial ideals over a field K in Chapter 3. To this end, we first have to impose an ordering on monomials. In Section 3.1, we give an overview over several such monomial orderings. For the convenience of the reader, we also recall some basic terminology from the field of abstract rewriting, which then helps us to define a reduction relation on polynomials. Using this reduction relation, we define Gröbner bases in the noncommutative polynomial ring and give several equivalent characterisations. Furthermore, we also discuss the notion of reduced Gröbner bases.

In Chapter 4, we dedicate ourselves to the computation of Gröbner bases in the noncommutative polynomial ring. In particular, we start by describing an interreduction procedure that enables us to detect and delete redundant generators of a finite system of generators of an ideal. After that, we state Bergman's famous Diamond Lemma [Ber78], which provides an algorithmic test to determine whether a given set of noncommutative polynomials is a Gröbner basis. As already mentioned, the ring of noncommutative polynomials is not Noetherian. Due to this fact, some ideals have infinite Gröbner bases, and consequently, we cannot expect to obtain terminating algorithms for computing Gröbner bases in this ring. Nevertheless, we shall present versions of the Buchberger algorithm and of the F4 algorithm to enumerate noncommutative Gröbner bases. Additionally, we extend both algorithms to keep track of cofactors during the computation. Provided that we can verify the membership of a given polynomial in an ideal, this enables us to produce a certificate for this ideal membership. At the end of Chapter 4, we present some basic optimisation strategies for Gröbner basis computations including deletion criteria as well as strategies to remove redundant generators of a Gröbner basis. Furthermore, we also discuss an algorithm proposed by Faugère and Lachartre [FL10] to efficiently compute the reduced row echelon form of matrices appearing during the execution of F4.

In Chapter 5, we summarise the algebraic framework developed in [RRH19], together with a simple three step procedure for proving operator identities in a fully algebraic fashion. We then illustratively apply this procedure to prove the uniqueness of the Moore-Penrose inverse of a matrix.

Chapter 6 is dedicated to the `OperatorGB` package. After discussing the main functionality

provided by the package and its usage, we give some insights concerning the design choices made when implementing the package and explain some optimisations, which helped to obtain reasonably efficient implementations of the algorithms. Finally, we compare our implementations to other well-known packages that provide functionality for computing noncommutative Gröbner bases.

Chapter 2

Noncommutative polynomials

To keep this thesis self-contained, we recall some important notions in this chapter, which are relevant for the remainder of this work. In particular, we define the *noncommutative polynomial ring*, which is the main algebraic structure we will be working with throughout this thesis, and its elements, the *noncommutative polynomials*. To this end, we first define the notion of the *free monoid* and then, based upon that, so-called *monoid rings*.

Definition 2.1. Let X be a set. A *word* w over X is a finite sequence of the form $w = x_1 \dots x_m$ where $m \in \mathbb{N}$ and $x_1, \dots, x_m \in X$. The quantity m is called the *length* of w and abbreviated by $|w|$. If $|w| = 0$, w is called the *empty word* and denoted by $w = 1$. The set of all words over X is denoted by $\langle X \rangle$. We define a multiplication on $\langle X \rangle$ as follows: for two words $w, w' \in \langle X \rangle$ with $w = x_1 \dots x_m$ and $w' = x'_1 \dots x'_n$, the product ww' is given by the concatenation of w and w', i.e. $ww' = x_1 \dots x_m x'_1 \dots x'_n$. Equipped with this multiplication, the set $\langle X \rangle$ becomes a monoid; the so-called *free monoid* generated by X.

Remark. If $|X| = 1$, i.e. $X = \{x\}$, then $\langle X \rangle = \{x^i \mid i \geq 0\}$ consists of all nonnegative powers of x and since $x^i x^j = x^{i+j} = x^{j+i} = x^j x^i$ for all $i, j \geq 0$, we end up with a commutative monoid. If $|X| > 1$, then $\langle X \rangle$ is not commutative anymore.

If the elements of X are explicitly given, say $X = \{x_1, \dots, x_n\}$, we typically omit the set parentheses in $\langle \{x_1, \dots, x_n\} \rangle$ and simply write $\langle x_1, \dots, x_n \rangle$.

Example 2.2. Let $X = \{x, y, z\}$. Then, the elements of $\langle X \rangle$ are of the form

$$1, x, y, z, xx, yy, zz, xy, yx, xz, zx, yz, zy, xxx, \dots$$

We also need the concept of divisibility in $\langle X \rangle$. So, we define what it means for a word w to be divisible by another word w'.

Definition 2.3. Let X be a set and $w, w' \in \langle X \rangle$. We say that w' *divides* w or w' is a *subword* of w if there exist $a, b \in \langle X \rangle$ such that $aw'b = w$. In this case, we write $w' \mid w$.

Example 2.4. Let $X = \{x, y, z\}$ and $w = xyxz$, $w' = yx \in \langle X \rangle$. Then, w' is a subword of w since $aw'b = w$ with $a = x$ and $b = z$. Also $w'' = xz \in \langle X \rangle$ is a subword of w. Here, the right factor b equals the empy word, i.e. $b = 1$.

Remark. Note that, unlike to the commutative case, a word w' can divide a word w in several ways. This means that the factors a and b are not necessarily unique. For example, consider the words $w = xyxyx$ and $w' = xy$ in $\langle x, y\rangle$. Then, clearly $w' \mid w$, but we have $w'xyx = xyw'x = w$.

The words in the free monoid $\langle X\rangle$ serve as the monomials in the noncommutative polynomial ring. What remains now, is that we have to construct a ring around the word monoid and another commutative ring. This can be done by considering the so-called *monoid ring*, which can be defined for an arbitrary monoid M. Note that by a ring we always mean a ring with unity.

Definition 2.5. Let R be a commutative ring and M be a monoid. Consider the set

$$RM \coloneqq \{ \sum_{m\in M} c_m m \mid c_m \in R \text{ such that } c_m = 0 \text{ for almost all } m\}.$$

This set, together with an addition defined by

$$\sum_{m\in M} c_m m + \sum_{m\in M} c'_m m \coloneqq \sum_{m\in M} (c_m + c'_m)m$$

and a multiplication defined by

$$\sum_{k\in M} c_k k \cdot \sum_{l\in M} c_l l \coloneqq \sum_{m\in M} \Big(\sum_{kl=m} c_k c_l \Big) m,$$

becomes a ring, which is called the *monoid ring* of M over R.

Now, we can put these pieces together to define the main algebraic structure we will be working with throughout this thesis, the *noncommutative polynomial ring*, together with its elements, the *noncommutative polynomials*. To this end, we fix an arbitrary commutative ring R and a nonempty set X for the rest of this chapter.

Definition 2.6. We refer to the monoid ring of $\langle X\rangle$ over R, denoted by $R\langle X\rangle$, as the *ring of noncommutative polynomials* in the indeterminates X with coefficients in R. In other contexts $R\langle X\rangle$ is also called the *free (associative) algebra* or the *free monoid ring* generated by X over R.

Remark. If $X = \{x\}$, we have already seen that $\langle X\rangle$ is a commutative monoid consisting of all nonnegative powers of x. In this case, $R\langle X\rangle$ becomes the usual univariate polynomial ring $R[x]$. If $|X| > 1$, then $R\langle X\rangle$ and $R[X]$ do not coincide anymore.

Definition 2.7. An element $f = \sum_{m\in\langle X\rangle} c_m m \in R\langle X\rangle$ is called a *(noncommutative) polynomial*. The element $c_m \in R$ is called the *coefficient* of f in m and denoted by $\operatorname{coeff}(f, m)$. The set $\operatorname{supp}(f) \coloneqq \{m \in \langle X\rangle \mid c_m \neq 0\}$ is called the *support* of f. Furthermore, we often refer to $m \in \langle X\rangle$ in this context as a *monomial* rather than as a word.

Remark. Note that indeterminates commute with coefficients in $R\langle X\rangle$ but not with each other.

Example 2.8. Let $R = \mathbb{Z}$ and $X = \{x, y\}$. We consider $R\langle X\rangle = \mathbb{Z}\langle x, y\rangle$. For $f_1 = xy + x$, $f_2 = xy - 2y \in \mathbb{Z}\langle x, y\rangle$ we can compute

$$\begin{aligned} f_1 + f_2 &= 2xy + x - 2y, \\ f_1 f_2 &= (xy + x)(xy - 2y) = xyxy + xxy - 2xyy - 2xy, \\ f_2 f_1 &= (xy - 2y)(xy + x) = xyxy + xyx - 2yxy - 2yx. \end{aligned}$$

Note that $f_1 f_2 \neq f_2 f_1$.

Often, we want to talk about all monomials appearing in a set of polynomials. To this end, it is convenient to extend the definition of the support from single polynomials to subsets of $R\langle X\rangle$.

Definition 2.9. Let $F \subseteq R\langle X\rangle$. We denote the set of all monomials appearing in any $f \in F$ by $\operatorname{supp}(F)$, i.e. $\operatorname{supp}(F) \coloneqq \bigcup_{f\in F} \operatorname{supp}(f)$.

Later, we will be dealing with sets of polynomials encoding certain identities and usually we are not only interested in the identities at hand but also in all their consequences. We can translate this into our setting of polynomials by considering *(two-sided) ideals* of $R\langle X\rangle$.

Definition 2.10. Let $(R', +, \cdot)$ be a (not necessarily commutative) ring. A subset $I \subseteq R'$ is called a *(two-sided) ideal* of R' if it satisfies the following two conditions.

1. $(I, +)$ is a subgroup of $(R', +)$.
2. For all $r, r' \in R'$ and all $f \in I$, also $rfr' \in I$.

Example 2.11. Every ring R' has two trivial ideals, namely $\{0\}$ and R' itself.

Usually, when we are given a set of polynomials $F \subseteq R\langle X\rangle$, we are looking for the smallest ideal of $R\langle X\rangle$ (in terms of set inclusion) containing F. This ideal is called the *ideal generated by* F and denoted by (F). It is easy to see that (F) is given by

$$(F) = \{\sum_{i=1}^{n} a_i f_i b_i \mid f_i \in F,\ a_i, b_i \in R\langle X\rangle,\ n \in \mathbb{N}\}.$$

The set F is called a *system of generators* of (F). An ideal I is said to be *finitely generated* if there exists a finite system of generators F such that $I = (F)$. We agree upon the convention to write $(f_1, \ldots, f_n)$ instead of $(\{f_1, \ldots, f_n\})$ if the elements of $F = \{f_1, \ldots, f_n\}$ are explicitly given.

Note that, if we consider the usual commutative polynomial ring $R[X]$ in finitely many variables over a Noetherian ring R, Hilbert's basis theorem states that $R[X]$ is Noetherian as well. This means that every ascending chain of ideals in $R[X]$ eventually becomes stationary, or equivalently, that every ideal in $R[X]$ is finitely generated. In the noncommutative case however, Hilbert's basis theorem fails if $|X| > 1$ (if $|X| = 1$ we have already seen that we end up with the usual univariate polynomial ring). Hence, there exist ideals in $R\langle X\rangle$ which are not finitely generated. One prominent example is the following: let R be any Noetherian ring (or even a field) and $X = \{x, y\}$. The ideal generated by the set $\{xy^i x \mid i \in \mathbb{N}\} \subseteq R\langle x, y\rangle$ has no finite system of generators. This means that $R\langle X\rangle$ is not Noetherian, which causes problems when it comes to computing Gröbner bases in this ring. We discuss this in more detail in Chapter 3 and 4.

To end this chapter, we introduce the notion of *cofactor representations*.

Definition 2.12. Let $F \subseteq R\langle X\rangle$. For an element $f \in (F)$, we call a representation of the form

$$f = \sum_{i=1}^{n} a_i f_i b_i, \tag{1}$$

with $f_i \in F$, $a_i, b_i \in R\langle X\rangle$ and $n \in \mathbb{N}$, a *cofactor representation* of f with respect to F. We refer to the elements a_i and b_i as the *cofactors* of f with respect to the representation (1).

Note that in general the cofactors a_i and b_i from several summands with the same f_i cannot be collected on both sides of the f_i simultaneously. However, collecting cofactors only on the right hand side of summands with the same f_i is possible if the left hand side cofactors are all equal. For example, we can do the following simplification

$$\sum_{j=1}^{k} a_i f_i b_j = a_i f_i \left(\sum_{j=1}^{k} b_j \right).$$

In a similar fashion, we can collect the cofactors on the left hand side if the cofactors on the right hand side of the summands are equal.

Remark. Given a cofactor representation of the form (1), we can expand the cofactors a_i and b_i into monomials in $\langle X\rangle$. Hence, we can write every $f \in (F)$ as

$$f = \sum_{i=1}^{m} c_i v_i f_i w_i,$$

with $c_i \in R$, $v_i, w_i \in \langle X\rangle$ and $f_i \in F$.

Example 2.13. Consider the ideal $(f_1, f_2) \subseteq \mathbb{Z}\langle x, y\rangle$ generated by $f_1 = yx - xy$ and $f_2 = xy - x$ and let $f = xyyx - xx \in (f_1, f_2)$. A cofactor representation of f is given by

$$f = xyf_1 + xyf_2 + f_2x.$$

Note that the two summands xyf_2 and f_2x cannot be merged together. This example also shows that a cofactor representation is not necessarily unique, since we also have

$$f = f_2yx + f_2x. \tag{2}$$

This is why we usually speak of *a* cofactor representation of f instead of *the* cofactor representation of f. Note that in (2) we can collect the cofactors on the right hand side of the summands to obtain $f = f_2(yx + x)$.

Given a polynomial $f \in R\langle X\rangle$ and a set $F \subseteq R\langle X\rangle$, one of the main goals of this thesis is to derive procedures to determine whether $f \in (F)$ or $f \notin (F)$. Although, in general, this so-called *ideal membership problem* is undecidable in $R\langle X\rangle$, in practice it turns out that we are often able to at least give an affirmative answer to this question, i.e. to verify that indeed $f \in (F)$. In this case, we can also compute a cofactor representation of f with respect to F. This cofactor representation then serves as a certificate for the ideal membership and can be checked independently. The main tool needed for these computations are (partial) Gröbner bases, which we discuss in the next chapter. However, for sake of simplicity, we only consider Gröbner bases in the noncommutative polynomial ring $K\langle X\rangle$ over a field K. So, for the rest of this thesis, we let K be an arbitrary field.

Chapter 3

Gröbner bases in $K\langle X\rangle$

In this chapter, we summarise the main results of the theory of Gröbner bases of ideals in noncommutative polynomial rings over a field K. Historically, first the commutative case was considered, where Buchberger was the first to characterise Gröbner bases as we know them today. In his famous PhD thesis [Buc65], written in 1965, he also provides an algorithm, now called Buchberger's algorithm, for computing Gröbner bases in the commutative polynomial ring.

The main theoretical results needed to adapt these concepts to the noncommutative case are due to Bergman [Ber78], who uses the abstract concept of reduction systems to generalise the ideas from the commutative setting, and were developed independent of Buchberger's theory about a decade later. Around the same time, also Bokut' [Bok76] proved statements that are essentially equivalent to the ones by Bergman. The first algorithmic approach to compute noncommutative Gröbner bases was proposed by Mora [Mor86] a few years later. In 1994, Mora also managed to unify the theory of Gröbner bases for commutative and noncommutative polynomial rings via a generalisation of the Gaussian elimination algorithm [Mor94]. For recent surveys on the theory of noncommutative Gröbner bases we refer to [Mor16, Nor01, Xiu12].

In this thesis, we define Gröbner bases via a certain reduction relation on polynomials. This reduction relation replaces a "larger" term in a polynomial by "smaller" terms and a Gröbner basis is a system of generators of a polynomial ideal for which this reduction relation has certain properties. In particular, we follow Bergman's approach and consider polynomial reduction as a linear map. The advantage of this approach is that it can be generalised to other settings, for example as done by Hossein Poor et al. [HRR18, Hos18], where, based on certain reduction systems, an analogous procedure of Buchberger's algorithm in tensor rings is presented. For a generalisation of Gröbner bases using a rewriting theory approach, we refer to [GHM19].

Before we can define the reduction relation on polynomials, we first have to clarify what we mean by "larger" and "smaller" terms of a polynomial. To this end, we recall the concept of a monomial ordering on $\langle X\rangle$. This is done in Section 3.1, where we also give several examples of such orderings. In Section 3.2.1, we recap some basic notions of abstract rewriting. Based upon those, we state the reduction relation in Section 3.2.2 and discuss some of its properties. Using this reduction relation, we can then define and characterise Gröbner bases in $K\langle X\rangle$ in full detail in Section 3.3.

For the rest of this thesis, we assume that the set X of indeterminates is finite.

3.1 Monomial orderings

Recall that a binary relation $\preceq$ on a set S is a subset of $S \times S$. We follow the usual convention to write $a \preceq b$ instead of $(a, b) \in \preceq$.

Definition 3.1. A *total ordering* $\preceq$ on a set S is a binary relation on S that satisfies the following properties for all $a, b, c \in S$.

1. If $a \preceq b$ and $b \preceq a$, then $a = b$. (*antisymmetry*)
2. If $a \preceq b$ and $b \preceq c$, then $a \preceq c$. (*transitivity*)
3. $a \preceq b$ or $b \preceq a$. (*connexity*)

Remark. The connexity property is sometimes also called *completeness* because it ensures that every pair of elements in S is comparable under this relation. It also implies reflexivity, i.e. $a \preceq a$ for all $a \in S$.

However, for our purpose, a total ordering is not sufficient because the properties of a total ordering alone cannot guarantee that the reduction relation that we will define later will always terminate. Informally speaking, we might end up in an infinite sequence of replacing larger terms by smaller terms without ever reaching some kind of minimal term. We will soon see an example of a total ordering where this can happen. So, what we are seeking for is a so-called *well-ordering*. If we additionally require that the ordering respects the multiplication in $\langle X \rangle$, then we obtain a *monomial ordering*.

Definition 3.2. Let $\preceq$ be a total ordering on $\langle X \rangle$. Then, $\preceq$ is called a *monomial ordering* or an *admissible ordering* on $\langle X \rangle$ if it satisfies the following two conditions:

1. $w \preceq w'$ implies $awb \preceq aw'b$ for all $a, b, w, w' \in \langle X \rangle$. (*compatibility with multiplication*)
2. Every nonempty subset of $\langle X \rangle$ has a least element. (*well-ordering*)

Remark. The property of being a well-ordering is equivalent to the fact that every descending chain $w_1 \succeq w_2 \succeq \dots$ of words in $\langle X \rangle$ eventually becomes stationary. This is the crucial property that will ensure that our reduction relation will always terminate.

We can also use the property of being a well-ordering together with the compatibility of a monomial ordering with the multiplication in $\langle X \rangle$ to derive the following proposition.

Proposition 3.3. *Let $\preceq$ be a monomial ordering on $\langle X \rangle$. Then, 1 is the least element of $\langle X \rangle$ with respect to $\preceq$.*

Proof. Assume that there exists $w \in \langle X \rangle$ such that $1 \succ w$. Then, the first condition of Definition 3.2 yields $w^n = w^n \cdot 1 \succ w^n \cdot w = w^{n+1}$ for all $n \in \mathbb{N}$, which implies the existence of an infinite strictly decreasing sequence $1 \succ w \succ w^2 \succ \dots$, but this is a contradiction to $\preceq$ being a well-ordering. □

The first ordering that we consider is the *lexicographic ordering*, which we denote by $\preceq_{lex}$ and which is probably best known from how the words in a dictionary are ordered. This is why it is sometimes also referred to as the *dictionary ordering*. We adapt this ordering to the free monoid $\langle X \rangle$ as follows.

Definition 3.4. Let $X = \{x_1, x_2, \ldots, x_n\}$. We order the elements of X as follows: $x_1 \prec_{lex} x_2 \prec_{lex} \cdots \prec_{lex} x_n$. Then, we have $w \preceq_{lex} w'$ for two monomials $w = x_{i_1} \ldots x_{i_k}, w' = x_{j_1} \ldots x_{j_l} \in \langle X \rangle$ if

1. there exists an index $1 \leq s \leq \min\{k, l\}$ such that $x_{i_m} = x_{j_m}$ for all $1 \leq m < s$ and $x_{i_s} \prec_{lex} x_{j_s}$, or
2. $x_{i_m} = x_{j_m}$ for all $1 \leq m \leq \min\{k, l\}$ and $k \leq l$.

Remark. By reordering the elements in $X = \{x_1, \ldots, x_n\}$, one can obtain different orderings. Formally, this means that for a permutation π of $\{1, \ldots, n\}$, we consider $\preceq_{lex,\pi}$ where the indeterminates are ordered as $x_{\pi(1)} \prec_{lex,\pi} x_{\pi(2)} \prec_{lex,\pi} \cdots \prec_{lex,\pi} x_{\pi(n)}$ and where we have $w \preceq_{lex,\pi} w'$ for two monomials $w = x_{i_1} \ldots x_{i_k}, w' = x_{j_1} \ldots x_{j_l} \in \langle X \rangle$ if

1. there exists an index $1 \leq s \leq \min\{k, l\}$ such that $x_{i_m} = x_{j_m}$ for all $1 \leq m < s$ and $x_{\pi(i_s)} \prec_{lex} x_{\pi(j_s)}$, or
2. $x_{i_m} = x_{j_m}$ for all $1 \leq m \leq \min\{k, l\}$ and $k \leq l$.

Then, the lexicographic ordering as defined above can be obtained as a special case by setting $\pi = \mathrm{id}$. To keep the notation uncluttered, we will only use the "usual" lexicographic ordering $\preceq_{lex}$ for all future definitions and examples in this thesis.

Example 3.5. Let $X = \{x, y, z\}$ and order the indeterminates as $x \prec_{lex} y \prec_{lex} z$. Then, we have $xyx \prec_{lex} xyz$ by the first condition of Definition 3.4 and $xy \prec_{lex} xyx$ by the second condition of Definition 3.4.

The lexicographic ordering is a total ordering but it is not a monomial ordering since it is not a well-ordering. This can easily be seen by taking $X = \{x, y\}$ and letting $x \prec_{lex} y$. Then, $y \succ_{lex} xy \succ_{lex} xxy \succ_{lex} \ldots$ is an infinite strictly decreasing chain in $\langle X \rangle$. Additionally, the lexicographic ordering is also not compatible with the multiplication in $\langle X \rangle$. For example, we have $x \prec_{lex} xx$ but $xy \succ_{lex} xxy$. Hence, this ordering will not be useful for our polynomial reduction relation.

Remark. In the commutative case, the lexicographic ordering is a monomial ordering, which can be obtained from Definition 3.4 by writing commutative monomials $x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ in the form

$$\underbrace{x_n \ldots x_n}_{\alpha_n\text{-times}} \underbrace{x_{n-1} \ldots x_{n-1}}_{\alpha_{n-1}\text{-times}} \cdots \underbrace{x_1 \ldots x_1}_{\alpha_1\text{-times}}.$$

Although the lexicographic ordering on $\langle X \rangle$ is not a monomial ordering, it can be used as a building block for other monomial orderings by considering so-called *induced monomial orderings*.

Definition 3.6. Let S be a semigroup equipped with a well-ordering $\leq$ that is compatible with the multiplication in S and let $\varphi : \langle X \rangle \to S$ be a semigroup homomorphism. Then, the *induced monomial ordering* $\preceq$ on $\langle X \rangle$ is defined by $w \preceq w'$ for $w, w' \in \langle X \rangle$ if

1. $\varphi(w) < \varphi(w')$, or
2. $\varphi(w) = \varphi(w')$ and $w \preceq_{lex} w'$.

For example, for $S = \mathbb{N}$ with the usual ordering and $\varphi(w) = |w|$ for $w \in \langle X \rangle$, the induced ordering first compares the monomials in $\langle X \rangle$ by their length and then uses the lexicographic ordering as a tie-breaker for words of the same length. In this way, we obtain the *degree lexicographic ordering*, which is also called the *graded lexicographic ordering* and which we will abbreviate by $\preceq_{deglex}$.

Example 3.7. Let $X = \{x, y, z\}$ and order the indeterminates as $x \prec_{lex} y \prec_{lex} z$. Using the lexicographic ordering, we had $xy \succ_{lex} xxy$, which caused some problems. Now, we have $xy \prec_{deglex} xxy$, since $\varphi(xy) = 2 < 3 = \varphi(xxy)$. We also have $xxy \prec_{deglex} xyx$, since $\varphi(xxy) = 3 = \varphi(xyx)$ and $xxy \prec_{lex} xyx$.

Sometimes, we want to value the appearance of certain indeterminates in a monomial more than the appearance of others. This leads to the following generalisation of the degree lexicographic ordering, called the *weighted degree lexicographic ordering* or *weighted graded lexicographic ordering* and abbreviated by $\preceq_{wlex,\omega}$. But before we can state this monomial ordering, we first have to define the *weighted degree* of a word in $\langle X \rangle$.

Definition 3.8. Let $X = \{x_1, \ldots, x_n\}$ and let $w = x_{i_1} \ldots x_{i_k} \in \langle X \rangle$. Furthermore, let $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n_{>0}$; we call ω a *weight tuple* of X. The *weighted degree* of w with respect to ω, denoted by $W_\omega(w)$, is given by

$$W_\omega(w) \coloneqq \sum_{j=1}^{k} \omega_{i_j}.$$

For a weight tuple ω of X the weighted degree lexicographic ordering $\preceq_{wlex,\omega}$ is then the ordering induced by $\varphi(w) = W_\omega(w)$ for $w \in \langle X \rangle$. The degree lexicographic ordering can be obtained as a special case of the weighted degree lexicographic ordering if we take $\omega = (1, \ldots, 1)$.

Remark. Note that the weight tuple ω used in the weighted degree lexicographic ordering must consist of strictly positive numbers. It is not sufficient to demand nonnegative numbers as the following example shows. Let $X = \{x, y\}$ and order the indeterminates as $x \prec_{lex} y$. Furthermore, let $\omega = (0, 1)$. Then, $W_\omega(x^n y) = 1$ for all $n \geq 0$ and $x^{n+1} y \prec_{lex} x^n y$ since $x \prec_{lex} y$, which yields that also $x^{n+1} y \prec_{wlex,\omega} x^n y$ for all $n \geq 0$. Hence, we end up with the infinite strictly decreasing chain $y \succ_{wlex,\omega} xy \succ_{wlex,\omega} xxy \succ_{wlex,\omega} \ldots$ of monomials in $\langle X \rangle$. This shows that the weighted degree lexicographic ordering is not a well-ordering in this case and consequently also not a monomial ordering.

Example 3.9. Let $X = \{x, y, z\}$ and order the indeterminates as $x \prec_{lex} y \prec_{lex} z$. Additionally, we also consider the weight tuple $\omega = (1, 2, 3) \in \mathbb{R}^3_{>0}$. Then, we have $xx \prec_{wlex,\omega} z$, since $\varphi(xx) = 2 < 3 = \varphi(z)$. We also have $xy \prec_{wlex,\omega} z$, since $\varphi(xy) = 3 = \varphi(z)$ and $xy \prec_{lex} z$.

The last monomial ordering that we consider in this thesis is the so-called *multigraded lexicographic ordering* which we denote by $\preceq_{mlex,\mathcal{S}}$. To be able to introduce this ordering, we need the notion of the length of a monomial *with respect to a set* $S \subseteq X$.

Definition 3.10. Let $X = \{x_1, \ldots, x_n\}$ and $S \subseteq X$. Furthermore, let $w = x_{i_1} \ldots x_{i_k} \in \langle X \rangle$. The *length of* w *with respect to* S, denoted by $|w|_S$, is given by

$$|w|_S \coloneqq |\{j \mid x_{i_j} \in S,\, 1 \leq j \leq k\}|.$$

Given a partition $\mathcal{S} = \{S_1, \ldots, S_k\}$ of X, we define $\varphi(w) = (|w|_{S_1}, \ldots, |w|_{S_k})$ for $w \in \langle X \rangle$. This is a semigroup homomorphism from $\langle X \rangle$ to $\mathbb{N}^k$, which induces the multigraded lexicographic ordering $\preceq_{mlex,\mathcal{S}}$ on $\langle X \rangle$. Note that for this purpose, we have to extend the usual ordering on $\mathbb{N}$ to $\mathbb{N}^k$ by defining $a \leq b$ for $a = (a_1, \ldots, a_k), b = (b_1, \ldots, b_k) \in \mathbb{N}^k$ if $a_i = b_i$ for all $1 \leq i \leq k$ or if $a_i < b_i$ for the smallest index $1 \leq i \leq k$ where a_i and b_i differ.

Remark. Typically, one partitions the set X into two parts, i.e $S_1 = S$, $S_2 = X \setminus S$. Such an instance of the multigraded lexicographic ordering is a so-called *elimination ordering* because the reduction relation that we will define later tends to remove (or eliminate) terms containing many indeterminates from S_2 and replace them by terms involving fewer or even no indeterminates from S_2. Also, note that if $\mathcal{S} = \{X\}$ we end up with the degree lexicographic ordering $\preceq_{deglex}$.

Example 3.11. Let $X = \{x, y, z\}$ and order the indeterminates as $x \prec_{lex} y \prec_{lex} z$. Additionally, let $\mathcal{S} = \{S_1, S_2\}$ with $S_1 = \{x\}$ and $S_2 = \{y, z\}$. Then, we have $yzy \prec_{mlex,S} x$, since $\varphi(yzy) = (0, 3) < (1, 0) = \varphi(x)$. We also have $xy \prec_{mlex,S} xyy$, since $\varphi(xy) = (1, 1) < (1, 2) = \varphi(xyy)$. Furthermore, $xyy \prec_{mlex,S} xyz$, since $\varphi(xyy) = (1, 2) = \varphi(xyz)$ and $xyy \prec_{lex} xyz$.

Remark. Further monomial orderings can be obtained by combining the ones defined above. For example, for a partition $\mathcal{S} = \{S_1, \ldots, S_k\}$ of X and weight tuples ω_i of S_i one can define a weighted multigraded lexicographic ordering as the monomial ordering induced by $\varphi(w) = (W_{\omega_1}(w), \ldots, W_{\omega_k}(w))$. It is also possible to use φ to map noncommutative monomials to commutative monomials, which are then compared by a commutative monomial ordering.

Since many of the following definitions strongly depend on the monomial ordering used, we fix an arbitrary monomial ordering $\preceq$ on $\langle X \rangle$ for the rest of this section. If we use a particular ordering in an example, we denote it by the corresponding subscript.

We can now define what we mean by a "large" term of a polynomial.

Definition 3.12. Let $f \in K\langle X \rangle \setminus \{0\}$ and let $m = \max_{\preceq} \operatorname{supp}(f)$. Then, m is called the *leading monomial* of f, denoted by $\operatorname{lm}(f)$. The coefficient of f in m is called the *leading coefficient* of f and abbreviated as $\operatorname{lc}(f)$. If $\operatorname{lc}(f) = 1$, then f is said to be *monic*. Furthermore, we refer to the product of the leading coefficient with the leading monomial as the *leading term* $\operatorname{lt}(f)$ of f, i.e. $\operatorname{lt}(f) \coloneqq \operatorname{lc}(f) \cdot \operatorname{lm}(f)$

Remark. Note that the leading monomial/coefficient/term of the zero polynomial remain undefined.

Example 3.13. The leading term strongly depends on the monomial order chosen. For example, let $K = \mathbb{Q}$, $X = \{x, y, z\}$ and order the indeterminates as $x \prec_{lex} y \prec_{lex} z$. Furthermore, let

$f = z + xy - xxx \in \mathbb{Q}\langle x, y, z\rangle$. If we use the lexicographic ordering, we get $\operatorname{lt}(f) = z$. In this case f is monic since the coefficient of f in z is 1. However, if we use the degree lexicographic ordering, we have $\operatorname{lt}(f) = -xxx$ and f is not monic anymore since $\operatorname{lc}(f) = -1$. If we use a multigraded lexicographic ordering with $\mathcal{S} = \{\{y\}, \{x, z\}\}$, we get $\operatorname{lt}(f) = xy$.

Remark. So, to be precise we should only speak of a leading monomial/coefficient/term with respect to a certain monomial ordering $\preceq$. In the following, we will omit this additional information when it is clear from the context which monomial ordering is used.

In the following proposition, we list some basic facts about leading monomials which follow basically directly from the definition.

Proposition 3.14. *Let $f, g \in K\langle X\rangle \setminus \{0\}$.*

1. *If $f + g \neq 0$, then $\operatorname{lm}(f + g) \preceq \max_{\preceq}\{\operatorname{lm}(f), \operatorname{lm}(g)\}$. In particular, we have $\operatorname{lm}(f + g) = \max_{\preceq}\{\operatorname{lm}(f), \operatorname{lm}(g)\}$ if and only if $\operatorname{lt}(f) \neq -\operatorname{lt}(g)$.*
2. *We have $\operatorname{lm}(fg) = \operatorname{lm}(f)\operatorname{lm}(g)$ and $\operatorname{lc}(fg) = \operatorname{lc}(f)\operatorname{lc}(g)$. Therefore, also $\operatorname{lt}(fg) = \operatorname{lt}(f)\operatorname{lt}(g)$.*
3. *For all $w, w' \in \langle X\rangle$, we have $\operatorname{lm}(wfw') = w\operatorname{lm}(f)w'$ and $\operatorname{lc}(wfw') = \operatorname{lc}(f)$. Hence, $\operatorname{lt}(wfw') = w\operatorname{lt}(f)w'$.*

The "smaller" terms of a polynomial are all its terms except the leading term. We call these terms the *tail* of the polynomial.

Definition 3.15. Let $f \in K\langle X\rangle \setminus \{0\}$. The *tail* of f, denoted by $\operatorname{tail}(f)$, is defined as $\operatorname{tail}(f) \coloneqq f - \operatorname{lt}(f)$.

Remark. We agree upon the same convention as for the leading term to omit the information with respect to which monomial ordering the tail is computed whenever it is clear from the context which monomial ordering is used.

We often talk about sets of polynomials instead of single polynomials. This is why it is convenient to extend the definitions of leading monomial/coefficient/term and tail to sets of polynomials.

Definition 3.16. Let $F \subseteq K\langle X\rangle$. We define the following sets.

1. $\operatorname{lc}(F) \coloneqq \{\operatorname{lc}(f) \mid f \in F \setminus \{0\}\} \subseteq K$,
2. $\operatorname{lm}(F) \coloneqq \{\operatorname{lm}(f) \mid f \in F \setminus \{0\}\} \subseteq \langle X\rangle$,
3. $\operatorname{lt}(F) \coloneqq \{\operatorname{lt}(f) \mid f \in F \setminus \{0\}\} \subseteq K\langle X\rangle$, and
4. $\operatorname{tail}(F) \coloneqq \{\operatorname{tail}(f) \mid f \in F \setminus \{0\}\} \subseteq K\langle X\rangle$.

Remark. If F is an ideal, then $\operatorname{lc}(F) \cup \{0\}$ is an ideal in K. All other sets fail to be ideals in general.

A monomial ordering $\preceq$ allows us to introduce a strict partial ordering $\lll$ on polynomials.

Definition 3.17. Let $f, g \in K\langle X\rangle$. Then, $f \lessdot g$ if

1. $f = 0$ and $g \neq 0$, or
2. $f, g \neq 0$ and $\mathrm{lm}(f) \prec \mathrm{lm}(g)$, or
3. $f, g \neq 0$ and $\mathrm{lm}(f) = \mathrm{lm}(g)$ and $\mathrm{tail}(f) \lessdot \mathrm{tail}(g)$.

Remark. We note that the ordering $\lessdot$ coincides with the multiset order $<_{mul}$ as discussed in [BN98, Section 2.5], when $<_{mul}$ is induced by $\prec$ and restricted to finite sets.

An important property of this ordering is that it is also well-ordering, i.e. there are no infinite sequences $f_1 \gtrdot f_2 \gtrdot \ldots$ of elements $f_1, f_2, \ldots \in K\langle X\rangle$.

Proposition 3.18. *The partial ordering $\lessdot$ is a well-ordering.*

Proof. Assume that $\lessdot$ is not a well-ordering and choose an infinite sequence

$$f_1 \gtrdot f_2 \gtrdot \ldots$$

of elements $f_1, f_2, \ldots \in K\langle X\rangle$ such that $\mathrm{lm}(f_1)$ is minimal with respect to $\preceq$ among all $f \in K\langle X\rangle$ starting an infinite sequence. Choosing this f_1 is possible since $\preceq$ is a well-ordering. We note that $f_i \neq 0$ for all $i \in \mathbb{N}$ because otherwise the sequence would be finite as 0 is minimal with respect to $\lessdot$. Hence, we can consider $t_i = \mathrm{lm}(f_i)$, which produces the following infinite descending chain in $\langle X\rangle$

$$t_1 \succeq t_2 \succeq \ldots.$$

Because $\preceq$ is a well-ordering, this sequence stabilises at some point $n \in \mathbb{N}$, i.e. $\mathrm{lm}(f_n) = \mathrm{lm}(f_{n+1}) = \ldots$. But if the leading terms of all f_i are equal for $i \geq n$, we must have

$$\mathrm{tail}(f_n) \gtrdot \mathrm{tail}(f_{n+1}) \gtrdot \ldots,$$

which is still an infinite sequence. But we have $\mathrm{lm}(\mathrm{tail}(f_n)) \prec \mathrm{lm}(f_1)$, which is a contradiction to the minimality of $\mathrm{lm}(f_1)$. □

3.2 The reduction relation

To be able to define and characterise the reduction relation on polynomials that will eventually lead to the definition of Gröbner bases, we require some basic notions and results from the field of abstract rewriting. To keep this thesis self-contained, we give a short survey of these results in Section 3.2.1. For further details we refer to [BN98, Section 2.1], which was also our main reference for this section. In Section 3.2.2, we then define the reduction relation on polynomials, discuss some of its properties and link it to the ideal membership problem.

3.2.1 Abstract rewriting

Abstract rewriting describes the concept of manipulating some object (e.g. a term) by the repeated application of simplification rules. From the mathematical point of view, this process can be described by a binary relation $\to$ on a set A. As for orderings, we follow the convention to write $x \to y$ instead of $(x, y) \in \to$.

Definition 3.19. Let A be a set and $\to \subseteq A \times A$ be a binary relation on A. We call the pair $(A, \to)$ an *abstract reduction system*. The relation $\to$ is called a *reduction relation* or simply a *reduction*.

Remark. The term *reduction* comes from the fact that in many applications a certain quantity is reduced with each application of a simplification rule.

The repeated application of such reductions can be described as the composition of relations. Recall that the composition of two relations $R \subseteq A \times B$ and $S \subseteq B \times C$ is defined by

$$R \circ S \coloneqq \{(x, z) \in A \times C \mid \exists y \in B : (x, y) \in R \wedge (y, z) \in S\}.$$

Based on this definition, we introduce some basic notions of the composition of a reduction with itself. To this end, we fix an arbitrary abstract reduction system $(A, \to)$ for the rest of this section.

Definition 3.20. We define the following notions.

$$\begin{aligned}
&\overset{0}{\to} \coloneqq \{(x, x) \mid x \in A\} && (\textit{identity}) \\
&\overset{i+1}{\to} \coloneqq \overset{i}{\to} \circ \to && ((i+1)\textit{-fold composition, } i \geq 0) \\
&\overset{=}{\to} \coloneqq \to \cup \overset{0}{\to} && (\textit{reflexive closure}) \\
&\overset{+}{\to} \coloneqq \bigcup_{i>0} \overset{i}{\to} && (\textit{transitive closure}) \\
&\overset{*}{\to} \coloneqq \overset{+}{\to} \cup \overset{=}{\to} && (\textit{reflexive transitive closure}) \\
&\leftarrow \coloneqq \{(y, x) \mid x \to y\} && (\textit{inverse}) \\
&\leftrightarrow \coloneqq \to \cup \leftarrow && (\textit{symmetric closure}) \\
&\overset{*}{\leftrightarrow} \coloneqq (\leftrightarrow)^* && (\textit{reflexive transitive symmetric closure})
\end{aligned}$$

Remark. Notations like $\leftarrow$ or $\leftrightarrow$ only make sense for arrow-like symbols. In case of an arbitrary relation $R \subseteq A \times A$ one would probably write R^{-1} for the inverse relation for example. However, we will only be dealing with arrow-like symbols in this thesis. So, we will not worry too much about this.

The following definitions extend the notation from above.

Definition 3.21. Let $x, y, z \in A$. Then,

1. x is *reducible* if there exists a y such that $x \to y$.

2. x is *in normal form* or *irreducible* if it is not reducible.

3. y is *a normal form* of x if $x \xrightarrow{*} y$ and y is in normal form.

4. x and y are *joinable* if there exists a z such that $x \xrightarrow{*} z \xleftarrow{*} y$. In this case, we write $x \downarrow y$.

To illustrate these notions we consider and slightly extend Example 2.1.2 from [BN98].

Example 3.22. Let $A = \mathbb{N} \setminus \{1\}$ and $\to = \{(m, n) \mid m > n \ \wedge \ n \mid m\}$. Then,

1. m is in normal form if and only if m is prime.

2. p is a normal form of m if and only if p is a prime factor of m. Consequently, all elements of A have a normal form, which however is not always unique. In fact, the only elements that have a unique normal form are prime powers. If we let $A = \mathbb{N}$, then every element of A has the same unique normal form, namely 1. Hence, in this case, all elements of A are joinable.

3. $m \downarrow n$ if and only if m and n are not relatively prime.

4. $\xrightarrow{+} \ = \to$ since $>$ and $\mid$ are already transitive.

5. $\leftarrow = \{(n, m) \mid \exists k \in \mathbb{N} \setminus \{1\} : m = kn\}$. Considering this relation, no element has a normal form.

6. $\overset{*}{\leftrightarrow} = A \times A$.

Example 3.23. Let $A = \langle a, b \rangle$ and $\to = \{(ubav, uabv) \mid u, v \in A\}$. Then,

1. w is in normal form if and only if w is sorted, i.e. of the form $a^m b^n$ for some $m, n \geq 0$.

2. Every w has a unique normal form, the result of sorting w.

3. $w_1 \downarrow w_2$ if and only if $w_1 \overset{*}{\leftrightarrow} w_2$ if and only if w_1 and w_2 contain the same number of a's and b's.

One important application of reduction systems and the one that we are interested in is checking the equivalence of two elements x and y of A with respect to the reduction relation $\to$, i.e. to check whether $x \overset{*}{\leftrightarrow} y$ holds. This problem is called the *word problem*. One approach to solve it is to reduce x and y to normal forms x' and y', respectively, and then check whether x' and y' are syntactically equal. However, as easy as this method sounds in theory, in practice two problems may arise which make the word problem undecidable in general. First of all, not every element of A needs to have a normal form as we could end up in an infinite sequence of reductions. This can be seen when considering the inverse relation from Example 3.22. And moreover, assuming that every element has a normal form, certain elements might still have several different normal forms. As an example, consider the relation from Example 3.22. In order to ensure existence and uniqueness of normal forms, we recall some important properties of reductions.

Definition 3.24. The reduction relation $\to$ is said to be

1. *Church-Rosser* if $x \overset{*}{\leftrightarrow} y$ implies $x \downarrow y$.
2. *confluent* if $y_1 \overset{*}{\leftarrow} x \overset{*}{\to} y_2$ implies $y_1 \downarrow y_2$.
3. *locally confluent* if $y_1 \leftarrow x \to y_2$ implies $y_1 \downarrow y_2$.
4. *normalising* if every element of A has a normal form.
5. *terminating* if there is no infinite descending chain $a_0 \to a_1 \to \dots$.

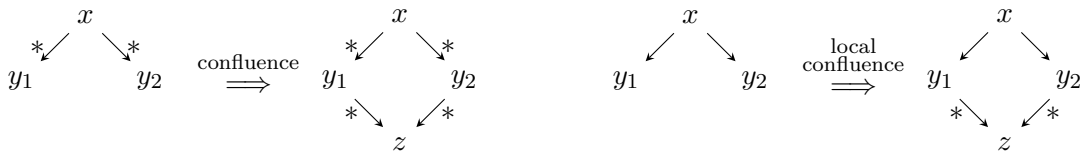


Figure 3.1: confluence and local confluence

Example 3.25. The reductions in Example 3.22 and 3.23 are both terminating but only the second one is Church-Rosser and confluent. The inverse relation in Example 3.22 is not even terminating.

So, to ensure the existence of normal forms for every element of A, the reduction relation has to be normalising. To prove that a concrete reduction relation is normalising is usually a nontrivial task. However, we can see that termination implies normalisation and in the case of our polynomial reduction relation, termination will be fairly easy to prove. We refer to Section 3.2.2 for more details. Concerning the uniqueness of normal forms, we gather some well known facts, cf. [BN98, Theorem 2.1.5, Lemma 2.7.2]

Theorem 3.26. *Let $\to$ be a reduction relation. Then, $\to$ is Church-Rosser if and only if $\to$ is confluent. Furthermore, if $\to$ is terminating, then $\to$ is confluent if and only if $\to$ is locally confluent.*

Proof. See [Win96, Theorem 8.1.2]. □

Remark. The second part of Theorem 3.26 is called *Newman's lemma.*

The following corollary is an immediate consequence of Theorem 3.26.

Corollary 3.27. *Let $\to$ be confluent and $x \overset{*}{\leftrightarrow} y$. Then,*

1. *$x \overset{*}{\to} y$ if y is in normal form, and*
2. *$x = y$ if both x and y are in normal form.*

Hence, we see that the confluence of $\to$ implies that every element of A has at most one normal form. Combining this fact with the property of normalisation we get that every element of A has a unique normal form. In fact, the converse is also true.

Proposition 3.28. *The reduction relation $\to$ is normalising and confluent if and only if every element of A has a unique normal form.*

Proof. The left-to-right implication follows immediately from Corollary 3.27 and the definition of normalising. For the other direction assume that every element of A has a unique normal form under $\to$. This obviously implies that $\to$ is normalising. To show confluence, let $x, y_1, y_2 \in A$ be such that $y_1 \overset{*}{\leftarrow} x \overset{*}{\to} y_2$. Since every element of A has a unique normal form, so do y_1 and y_2. Denote these normal forms by n_1 and n_2, respectively. As $x \overset{*}{\to} y_1$ and $x \overset{*}{\to} y_2$ by assumption, n_1 and n_2 are also normal forms of x. But x has to have a unique normal form, so $n_1 = n_2$. This implies $y_1 \downarrow y_2$. □

In case of our polynomial reduction relation, we have already anticipated that proving normalisation is easy. So, it remains to find confluent polynomial reduction systems in order to ensure the existence of unique normal forms. This fact will lead to the definition of Gröbner bases.

3.2.2 Polynomial reduction

We have finally gathered all tools to be able to discuss the central notion of the theory of Gröbner bases, the (noncommutative) polynomial reduction relation. Following the nomenclature in [Ber78], we first define a *reduction system.*

Definition 3.29. We call a pair $r = (w_r, f_r) \in \langle X \rangle \times K\langle X \rangle$ a *reduction rule.* For every two words $a, b \in \langle X \rangle$ such a reduction rule r induces a K-vector space endomorphism

$$h_{a,r,b} : K\langle X \rangle \to K\langle X \rangle,$$

defined by mapping the basis element $aw_rb \in \langle X \rangle$ to $af_rb \in K\langle X \rangle$ and fixing all other basis elements in $\langle X \rangle$. We refer to $h_{a,r,b}$ as a *reduction.* Furthermore, we call a set S of reduction rules a *reduction system.*

In the following, we introduce some basic terminology for such a reduction system S. To this end, we denote by $r = (w_r, f_r) \in S$ a reduction rule, by $a, b \in \langle X \rangle$ two words and by $f, g \in K\langle X \rangle$ two polynomials.

We say that a reduction $h_{a,r,b}$ acts *trivially* on f if $\mathrm{coeff}(f, aw_rb) = 0$, i.e. $h_{a,r,b}(f) = f$. Moreover, r *reduces* f to g if $h_{a,r,b}$ acts nontrivially on f and $h_{a,r,b}(f) = g$ for some a, b. By a slight abuse of notation we call the words a and b *cofactors (of the reduction).* If r reduces f to g using the cofactors a, b we write $f \to_{a,r,b} g$ or simply $f \to_r g$ if we do not care about the particular cofactors.

A reduction system S induces a reduction relation $\to_S$ on $K\langle X \rangle$ by defining $f \to_S g$ for $f, g \in K\langle X \rangle$ if there exists a reduction rule $r \in S$ such that r reduces f to g. For a fixed reduction system S, we say f can be reduced to g by S, denoted by $f \overset{*}{\to}_S g$, if either $f = g$ or there exist $r_1, \ldots, r_n \in S$ and $f_1, \ldots, f_{n-1} \in K\langle X \rangle$ such that

$$f \to_{r_1} f_1 \to_{r_2} \cdots \to_{r_{n-1}} f_{n-1} \to_{r_n} g.$$

Remark. Note that the reduction sequence has to be finite.

In terms of the terminology of Section 3.2.1 we can consider $(K\langle X\rangle, \to_S)$ as an abstract reduction system induced by S. Furthermore, $\overset{*}{\to}_S$ corresponds to the reflexive transitive closure of the reduction relation $\to_S$.

According to Definition 3.29, any pair (w, f) consisting of a monomial $w \in \langle X\rangle$ and a polynomial $f \in K\langle X\rangle$ can de used to define a reduction rule. This general setting is advantageous in many situations, for example consider the concept of *rewriting* in [RRH19, CHRR20]. We, however, are only interested in certain combinations of monomials w and polynomials f. To make this precise, we introduce the notion of an *induced reduction system*.

Definition 3.30. Let $F \subseteq K\langle X\rangle$ and let $\preceq$ be a monomial ordering on $\langle X\rangle$. Then,

$$S_{F,\preceq} \coloneqq \left\{ \left(\operatorname{lm}(f), -\frac{1}{\operatorname{lc}(f)} \operatorname{tail}(f) \right) \in \langle X\rangle \times K\langle X\rangle \mid f \in F \setminus \{0\} \right\}$$

is called the by F and $\preceq$ *induced reduction system*. We denote the reduction relation induced by $S_{F,\preceq}$ by $\to_{F,\preceq}$.

Remark. When working with the reduction system $S_{F,\preceq}$ induced by a set of polynomials $F \subseteq K\langle X\rangle$ and a monomial ordering $\preceq$, we identify a reduction rule $r = (\operatorname{lm}(f), -\frac{1}{\operatorname{lc}(f)} \operatorname{tail}(f)) \in S_{F,\preceq}$ with the polynomial $f \in F$ and write $h_{a,f,b}(g) = g'$ instead of $h_{a,r,b}(g) = g'$ and $g \to_{a,f,b} g'$ instead of $g \to_{a,r,b} g'$, respectively. Furthermore, whenever the monomial ordering $\preceq$ used is clear from the context, we simply write S_F instead of $S_{F,\preceq}$ and $\to_F$ instead of $\to_{F,\preceq}$.

An induced reduction system replaces the leading term of a polynomial by its tail, and therefore, replaces a "larger" term by "smaller" terms. The following theorem tells us that in this way we always obtain a terminating reduction relation. For the rest of this thesis, we fix a monomial ordering $\preceq$ on $\langle X\rangle$ with respect to which all further reduction systems will be computed unless explicitly stated otherwise.

Theorem 3.31. *Let S_F be the reduction system induced by some set $F \subseteq K\langle X\rangle$. Then, $\to_F$ is terminating.*

Proof. Assume that $\to_F$ is not terminating and choose an infinite sequence

$$f_1 \to_F f_2 \to_F \dots$$

of elements $f_1, f_2, \dots \in K\langle X\rangle$. Since a reduction step only introduces terms which are strictly smaller than the term that is replaced, we obtain the following infinite sequence

$$f_1 \gg f_2 \gg \dots$$

in $K\langle X\rangle$, which is a contradiction to Proposition 3.18. □

In order to obtain a terminating reduction relation, the requirement in Definition 3.30 that $\preceq$ is a monomial ordering cannot be weakened as the following example shows.

Example 3.32. Let K be an arbitrary field, $X = \{x, y\}$ and sort the indeterminates as $x \prec_{lex} y$. Using the set $F = \{x^i y - x^{i+1} y \mid i \geq 0\} \subseteq K\langle x, y\rangle$ and $\preceq_{lex}$, we construct a reduction system $S_{F,\preceq_{lex}}$ according to Definition 3.30 and show that the induced reduction relation is not terminating. Recall that $\preceq_{lex}$ is a total ordering on $\langle X\rangle$ but not a monomial ordering. We have already seen that $x^i y \succ_{lex} x^{i+1} y$. So, following Definition 3.30, we compute

$$S_{F,\preceq_{lex}} = \{(x^i y, x^{i+1} y) \mid i \geq 0\}.$$

Hence, the induced reduction relation $\to_{F,\preceq_{lex}}$ allows us to replace any term of the form $x^i y$ by $x^{i+1} y$ for all $i \geq 0$. Using this fact, we can construct the following infinite sequence

$$y \ \to_{y-xy} \ xy \ \to_{xy-xxy} \ xxy \ \to_{xxy-xxxy} \ \ldots,$$

which shows that $\to_{F,\preceq_{lex}}$ is not terminating.

From now on, we shall only work with reductions systems S_F with are induced by a monomial ordering. Consequently, termination of the reduction relation $\to_F$ will always be guaranteed. In the following, we also state some useful properties this reduction relation.

Proposition 3.33. *Let $F \subseteq K\langle X\rangle$, $c \in K \setminus \{0\}$, $m, m' \in \langle X\rangle$ and $g, g', p \in K\langle X\rangle$.*

1. *If $g \to_F g'$, then $cmgm' \to_F cmg'm'$.*
2. *If $g \to_F g'$, then $g + p \downarrow_F g' + p$.*

Proof. 1. If $g \to_F g'$, then by definition there exist $f \in F$ and $a, b \in \langle X\rangle$ such that $h_{a,f,b}(g) = g'$. But then, $h_{ma,f,bm'}(cmgm') = cmh_{a,f,b}(g)m' = cmg'm'$.

2. Let $f \in F$ and $a, b \in \langle X\rangle$ be such that $g \to_{a,f,b} g'$. This means that $h_{a,f,b}(g) = g'$. Furthermore, it follows from the definition of an admissible reduction system that $h_{a,f,b}$ acts trivially on g'. This yields

$$h_{a,f,b}(g + p) = h_{a,f,b}(g) + h_{a,f,b}(p) = g' + h_{a,f,b}(p) = h_{a,f,b}(g') + h_{a,f,b}(p) = h_{a,f,b}(g' + p),$$

where the first and last equality follow from the linearity of $h_{a,f,b}$. This shows that $g + p \downarrow_F g' + p$. □

Having in mind that we are actually interested in certifying ideal membership, we will now link the polynomial reduction relation with the ideal membership problem via the so-called *ideal congruence*.

Definition 3.34. Let $I \subseteq K\langle X\rangle$ be an ideal and $f, g \in K\langle X\rangle$. We say f and g are *congruent modulo I*, denoted by $f \equiv_I g$, if $f - g \in I$.

Remark. Note that $f \in I$ if and only if $f \equiv_I 0$.

The following theorem establishes the crucial relation.

Theorem 3.35. *Let $F \subseteq K\langle X\rangle$. The reflexive transitive symmetric closure of $\to_F$ equals the ideal congruence modulo (F), i.e. $\overset{*}{\leftrightarrow}_F = \equiv_{(F)}$.*

Proof. To prove the first inclusion "$\subseteq$" we note that both, $\overset{*}{\leftrightarrow}_F$ and $\equiv_{(F)}$, are equivalence relations. Additionally, $\overset{*}{\leftrightarrow}_F$ is the smallest equivalence relation containing $\rightarrow_F$. Hence, $\rightarrow_F \subseteq \equiv_{(F)}$ implies $\overset{*}{\leftrightarrow}_F \subseteq \equiv_{(F)}$. So, we show $\rightarrow_F \subseteq \equiv_{(F)}$. To this end, let $g, g' \in K\langle X\rangle$ be such that $g \rightarrow_F g'$. This means that there exist $f \in F$ and $a, b \in \langle X\rangle$ such that $h_{a,f,b}(g) = g'$. Since, $h_{a,f,b}$ acts trivially on g', we get that $h_{a,f,b}(g) = h_{a,f,b}(g')$ and consequently $h_{a,f,b}(g - g') = 0$. But this means that $g - g' = afb \in (F)$, i.e. $g \equiv_{(F)} h$.

For the second inclusion "$\supseteq$" let $g, g' \in K\langle X\rangle$ be such that $g \equiv_{(F)} g'$, i.e. $g = g' + \sum_{i=1}^{n} c_i a_i f_i b_i$ with $c_i \in K$, $a_i, b_i \in \langle X\rangle$ and $f_i \in F$. We show that $g \overset{*}{\leftrightarrow}_F g'$ holds for the case $n = 1$. The general statement for $n \geq 1$ then follows by induction on n. It is clear that $f_1 \rightarrow_F 0$. Consequently, by Proposition 3.33.1. also $c_1 a_1 f_1 b_1 \rightarrow_F 0$ holds. Then, Proposition 3.33.2 tells us that $g = g' + c_1 a_1 f_1 b_1 \downarrow_F g' + 0 = g'$ and therefore $g \overset{*}{\leftrightarrow}_F g'$. □

Hence, given a system of generators $F \subseteq K\langle X\rangle$ of an ideal I and a polynomial $f \in K\langle X\rangle$, we can verify the ideal membership of f in I by checking whether $f \overset{*}{\rightarrow}_F 0$ holds. Knowing that according to Theorem 3.31 the reduction relation $\rightarrow_F$ is terminating, this approach seems to be very promising. However, by taking a look at the following example we notice a big problem.

Example 3.36. Let $F = \{f_1, f_2, f_3\} \subseteq \mathbb{Q}\langle x, y\rangle$ with $f_1 = yx - xy - y$, $f_2 = xy - x$ and $f_3 = xxy$. Furthermore, let $f = xyx - x \in \mathbb{Q}\langle x, y\rangle$. If we compute all leading monomials/coefficients and tails with respect to $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y$, then S_F is given by

$$S_F = \{(yx, xy + y), (xy, x), (xxy, 0)\}.$$

The first reduction rule $(yx, xy + y)$ tells us that we can replace any occurrence of the term yx by the polynomial $xy + y$. We see that $yx \mid xyx \in \operatorname{supp}(f)$. So, we can apply this reduction rule to reduce f using the cofactors x and 1 to obtain $x(xy + y) - x = xxy + xy - x$. Now, we see that the second reduction rule is applicable. This reduces $xxy + xy - x$ to $xxy + x - x = xxy$. Finally, we can apply the third reduction rule to obtain 0. In short, this can be written as

$$f \ \rightarrow_{x,f_1,1} \ xxy + xy - x \ \rightarrow_{1,f_2,1} \ xxy \ \rightarrow_{1,f_3,1} \ 0,$$

or even shorter as $f \overset{*}{\rightarrow}_F 0$, which shows that $f \in (F)$. However, if we use the second reduction rule (xy, x) instead of the third to reduce xxy in the last step, we obtain

$$xxy \rightarrow_{x,f_2,1} xx.$$

Since xx is irreducible under $\rightarrow_F$ we have computed a different normal form of f which does not help us in deciding whether $f \in (F)$.

This shows that we cannot guarantee that the normal forms are unique. This is due to the fact that for an arbitrary set $F \subseteq K\langle X\rangle$ the reduction relation $\rightarrow_F$ is not necessarily confluent. Such distinguished sets for which $\rightarrow_F$ is confluent are called *Gröbner bases* of the ideal (F).

3.3 Characterisation of Gröbner bases

Definition 3.37. Let $I \in K\langle X\rangle$ be an ideal and $G \subseteq I \setminus \{0\}$ such that $(G) = I$. Then, G is called a *Gröbner basis* of I if and only if $\to_G$ is confluent.

Our characterisation of Gröbner bases is by far not the only one possible. In the following, we list some equivalent definitions.

Theorem 3.38. *Let $I \subseteq K\langle X\rangle$ be an ideal and $G \subseteq I \setminus \{0\}$ such that $(G) = I$. Then, the following conditions are equivalent.*

1. *G is a Gröbner basis of I.*
2. *$f \overset{*}{\to}_G 0$ for all $f \in I$.*
3. *$\mathrm{lm}(I) = \{w\mathrm{lm}(g)w' \mid g \in G,\ w, w' \in \langle X\rangle\}$.*
4. *$(\mathrm{lm}(I)) = (\mathrm{lm}(G))$.*
5. *Every $f \in I \setminus \{0\}$ can be written as a bounded linear combination of elements in G, i.e. there exist $g_i \in G$ and $a_i, b_i \in K\langle X\rangle$ such that $f = \sum_{i=1}^{n} a_i g_i b_i$ with $\mathrm{lm}(a_i g_i b_i) \preceq \mathrm{lm}(f)$ for $1 \leq i \leq n$.*

Proof. "1. $\Leftrightarrow$ 2.": To show 1. $\Rightarrow$ 2. suppose that G is a Gröbner basis and let $f \in I$. Since $f \equiv_I 0$, we have $f \overset{*}{\leftrightarrow}_G 0$ by Theorem 3.35. According to Theorem 3.26, G has the Church-Rosser property. This implies $f \downarrow_G 0$. Finally, Corollary 3.27 gives us $f \overset{*}{\to}_G 0$, since 0 is irreducible. Conversely, to prove 2. $\Rightarrow$ 1., we assume that $f \overset{*}{\to}_G 0$ holds for all $f \in I$ and let $g, g_1, g_2 \in K\langle X\rangle$ be such that $g_1 \ {}_G\overset{*}{\leftarrow} g \overset{*}{\to}_G g_2$. Then, clearly $g_1 \overset{*}{\leftrightarrow}_G g_2$ holds, and therefore, we have $g_1 \equiv_I g_2$ by Theorem 3.26. Consequently, $g_1 - g_2$ lies in I and so $g_1 - g_2 \overset{*}{\to}_G 0$ by our assumption. To finish the proof, we can use Proposition 3.33 to deduce

$$(g_1 - g_2) + g_2 = g_1 \downarrow_G g_2 = 0 + g_2,$$

which shows that $\to_G$ is confluent

"2. $\Leftrightarrow$ 3.": We denote $J = \{w\,\mathrm{lm}(g)w' \mid g \in G,\ w, w' \in \langle X\rangle\}$. First, we show 2. $\Rightarrow$ 3.. To this end, assume that $f \overset{*}{\to}_G 0$ holds for all $f \in I$. Since $G \subseteq I$, also $wgw' \in I$ for all $g \in G$ and $w, w' \in \langle X\rangle$. Using Proposition 3.14.3, it follows that $w\,\mathrm{lm}(g)w' = \mathrm{lm}(wgw') \in \mathrm{lm}(I)$. Hence, $J \subseteq \mathrm{lm}(I)$. For the other inclusion, let $t \in \mathrm{lm}(I)$. Then, there exists $f \in I$ such that $\mathrm{lm}(f) = t$. By assumption, we can reduce f to 0. Since a reduction step only introduces terms which are strictly smaller than the term that is replaced, it is not possible that the leading term of f cancels after applying a reduction rule. Hence, there must exist a $g \in G$ which reduces the leading term of f. But this means that $\mathrm{lm}(g) \mid \mathrm{lm}(f) = t$ and consequently $t \in J$. Now, to show 3. $\Rightarrow$ 2. suppose that $\mathrm{lm}(I) = J$ holds and assume that not all $f \in I$ can be reduced to 0. Let $\tilde{f} \in I$ be such an f and such that $\mathrm{lm}(\tilde{f})$ is minimal. By assumption, there exists $g \in G$ such that $\mathrm{lm}(g) \mid \mathrm{lm}(\tilde{f})$. Hence, we can use this g to reduce $\tilde{f}$ to h and by the properties of our reduction relation we have $\mathrm{lm}(h) \prec \mathrm{lm}(\tilde{f})$. Moreover, if $\tilde{f}$ cannot be reduced to 0, neither can h. But this is a contradiction to the minimality of $\mathrm{lm}(\tilde{f})$.

"3. $\Leftrightarrow$ 4.": The implication 3. $\Rightarrow$ 4. is clear. To prove 4. $\Rightarrow$ 3. we again denote $J = \{w\,\mathrm{lm}(g)w' \mid g \in G,\ w, w' \in \langle X\rangle\}$. The inclusion $J \subseteq \mathrm{lm}(I)$ follows as above. For the other inclusion, assume $(\mathrm{lm}(I)) = (\mathrm{lm}(G))$ and let $t \in \mathrm{lm}(I)$. By assumption, we can write t as a linear combination of the elements in $\mathrm{lm}(G)$, i.e. there exist $c_i \in K$, $u_i, v_i \in \langle X\rangle$ and $g_i \in G$ such that

$$t = \sum_{i=1}^{n} c_i v_i\,\mathrm{lm}(g_i)w_i.$$

But since t is a monomial, all summands in this linear combination except for one must cancel, so that

$$t = v_{i_0}\,\mathrm{lm}(g_{i_0})w_{i_0},$$

for some $1 \leq i_0 \leq n$, which means that there exists $g \in G$ such that $\mathrm{lm}(g) \mid t$ and consequently $t \in J$.

"2. $\Rightarrow$ 5.": Suppose $f \xrightarrow{*}_G 0$ holds for all $f \in I$. Hence, for an arbitrary but fixed $f \in I \setminus \{0\}$ there exist $a_i, b_i \in K\langle X\rangle$ and $g_i \in G$ such that

$$f \to_{a_1,g_1,b_1} \cdots \to_{a_n,g_n,b_n} 0.$$

Writing out this reduction process yields

$$f = f - 0 = \sum_{i=1}^{n} a_i g_i b_i.$$

The property that $\mathrm{lm}(a_i g_i b_i) \preceq \mathrm{lm}(f)$ for $1 \leq i \leq n$ follows immediately from the definition of $\to_G$.

"5. $\Rightarrow$ 3.": As before, we denote $J = \{w\,\mathrm{lm}(g)w' \mid g \in G,\ w, w' \in \langle X\rangle\}$ and note that $J \subseteq \mathrm{lm}(I)$ follows as above. Now suppose that every $f \in I \setminus \{0\}$ can be written as a bounded linear combination of elements in G and let $f \in I \setminus \{0\}$ be arbitrary. If we expand all cofactors a_i and b_i in the linear combination, we can write f as

$$f = \sum_{i=1}^{m} c_i v_i g_i w_i,$$

for some $c_i \in K$ and $v_i, w_i \in \langle X\rangle$. We must have $\mathrm{lm}(f) \in \mathrm{supp}(v_i g_i w_i)$ for some $1 \leq i \leq n$. Since $\mathrm{lm}(v_i g_i w_i) \preceq \mathrm{lm}(f)$ by assumption, this implies $\mathrm{lm}(f) = \mathrm{lm}(v_i g_i w_i) = v_i\,\mathrm{lm}(g_i)w_i \in J$, where the last equality follows from Proposition 3.14.3. □

From these characterisations we can see that a Gröbner basis of an ideal I is by no means unique. In fact, if $G \subseteq I$ is a Gröbner basis of I, then so is $G \cup \{f\}$ for every nonzero $f \in I$. Furthermore, the set $I \setminus \{0\}$ is always a Gröbner basis of I. Thus, we can deduce the following corollary.

Corollary 3.39. *Every ideal in $K\langle X\rangle$ has a Gröbner basis.*

Another consequence of Theorem 3.38 is a nice result about monomial ideals, i.e. ideals which have a system of generators consisting only of monomials.

Corollary 3.40. *Let $S \subseteq \langle X \rangle$ be a set of monomials generating an ideal $(S) \subseteq K\langle X \rangle$. Then, S is a Gröbner basis of (S).*

Proof. S clearly satisfies condition 3 of Theorem 3.38. Hence, it is a Gröbner basis of (S). □

Example 3.41. We reconsider Example 3.36 where we worked with $F = \{f_1, f_2, f_3\} \subseteq \mathbb{Q}\langle x, y \rangle$ with $f_1 = yx - xy - y$, $f_2 = xy - x$ and $f_3 = xxy$. We claim that if we use $\preceq_{deglex}$ as a monomial ordering where we ordered the indeterminates as $x \prec_{lex} y$, then a Gröbner basis G of (F) is given by

$$G = \{x, y\}.$$

First, we show that indeed $(G) = (F)$, which follows from

$$\begin{aligned} x &= -xf_1 + f_2x - (x+1)f_2 \in (F), \\ y &= -xf_1y - f_1xy + (2x-1)f_1 + (2x-y)f_2 - 2f_2x - (x+1)f_2y + (y-1)f_3 \in (F). \end{aligned}$$

Now, Corollary 3.40 tells us that G is indeed a Gröbner basis of (F).

Although a Gröbner basis G of an ideal $I \subseteq K\langle X \rangle$ is not unique in general, we can demand certain additional properties from G in order to obtain a special and unique Gröbner basis of I, the so-called *reduced Gröbner basis* of I. To be able to introduce this notion, we first have to define an *interreduced* set of polynomials.

Definition 3.42. Let $F \subseteq K\langle X \rangle \setminus \{0\}$. We say that F is *interreduced* if every $f \in F$ is in normal form with respect to $\to_{F \setminus \{f\}}$.

Definition 3.43. Let $I \subseteq K\langle X \rangle$ be an ideal and let G be a Gröbner basis of I. Then, G is called the *reduced Gröbner basis* of I if G is interreduced and all polynomials in G are monic.

Example 3.44. The Gröbner basis in Example 3.41 is the reduced Gröbner basis of (F).

Proposition 3.45. *Every ideal $I \subseteq K\langle X \rangle$ has a unique reduced Gröbner basis.*

In Section 4.1 we give a constructive proof of Proposition 3.45 for the special case that I has a finite Gröbner basis. For a proof of the general case we refer to [Xiu12, Proposition 3.3.17]. As a first step towards our proof, we note that we can reduce the elements in a Gröbner basis by each other without losing the property of being a Gröbner basis.

Proposition 3.46. *Let $I \subseteq K\langle X \rangle$ be an ideal and let G be a Gröbner basis of I. Furthermore, let $g \in G$ and $g' \in K\langle X \rangle$ be such that $g \to_{G \setminus \{g\}} g'$. Then, $(G \setminus \{g\}) \cup \{g'\}$ is also a Gröbner basis of I.*

To prove this proposition, we first establish the fact that such an interreduction does not change the generated ideal.

Lemma 3.47. *Let $F \in K\langle X \rangle$, $f \in (F)$ and $f' \in K\langle X \rangle$ be such that $f \to_{F \setminus \{f\}} f'$. Then, $(F) = ((F \setminus \{f\}) \cup \{f'\})$.*

Proof. We denote $F' = (F \setminus \{f\}) \cup \{f'\}$ and note that it suffices to prove that $f \in (F')$ and $f' \in (F)$. By Theorem 3.35, $f \to_{F\setminus\{f\}} f'$ implies $f \equiv_{F\setminus\{f\}} f'$, i.e. $f = f' + g$ for some $g \in F \setminus \{f\}$, which shows that $f \in (F')$. Conversely, we can write $f' = f - g \in (F)$, which completes the proof. □

Proof of Proposition 3.46. We denote $G' = (G \setminus \{g\}) \cup \{g'\}$. According to Lemma 3.47, we have $I = (G) = (G')$. Hence, it remains to show that G' is a Gröbner basis of I. We prove this by showing that every element in I that can be reduced by g, can also be reduced by some element in G'. Then, the statement follows from the confluence of $\to_G$. To this end, let $\tilde{g} \in G \setminus \{g\}$ be such that $g \to_{\tilde{g}} g'$. If $\tilde{g}$ reduces the leading term of g, then we must have $\mathrm{lm}(\tilde{g}) \mid \mathrm{lm}(g)$. If $\tilde{g}$ reduces some monomial in the tail of g, then the leading term of g remains unchanged, i.e. $\mathrm{lm}(g') = \mathrm{lm}(g)$. In any case, we see that we can replace a reduction done by g either by $\tilde{g}$ or by g', which are both elements of G'. □

In contrast to the commutative case, the ring $K\langle X\rangle$ is not Noetherian if X contains more than one element. Due to this fact, there are ideals of noncommutative polynomials which do not have a finite Gröbner basis. To see that this can even be the case for principal ideals, we consider the following example.

Example 3.48. Let K be any field and $X = \{x, y\}$. We consider $(g) \subseteq K\langle X\rangle$ with $g = xyx - xy$. We use $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y$. We observe that

$$xyyx - xyy = xyg - gyx + gy \in (g),$$

but $xyyx - xyy$ cannot be reduced to zero by g. Hence, $\{g\}$ cannot be a Gröbner basis of (g). We claim that the reduced Gröbner basis of (g) is given by

$$G = \{xy^i x - xy^i \mid i \in \mathbb{N}\},$$

which implies that (g) cannot have a finite Gröbner basis. Because if (g) had a finite Gröbner basis G', then we could use G' to compute the reduced Gröbner basis of (g) (see Corollary 4.2), which would turn out to be finite; a contradiction to our claim and the uniqueness of the reduced Gröbner basis.

We conclude this example by proving our claim. Obviously, G is interreduced and all its elements are monic. Furthermore, for $g_i = xy^i x - xy^i$ we have

$$g_{i+1} = xyg_i - g_1 y^i x + g_1 y^i$$

and since $g_1 = g \in (g)$ this shows inductively that $(G) = (g)$. Hence, it remains to check that $\to_G$ is confluent in order to prove that G is indeed the reduced Gröbner basis of (g). We postpone this last step to Section 4.2.

Hence, even if we use Gröbner bases, which induce a terminating and confluent reduction system, the ideal membership problem of noncommutative polynomials still remains undecidable in general simply because we cannot guarantee that we are always able to compute a complete (potentially infinite) Gröbner basis. However, there still exist procedures to enumerate Gröbner

bases in $K\langle X\rangle$. If such an enumeration is stopped after finitely many steps without obtaining a Gröbner basis, we refer to the result as a *partial Gröbner basis*. In practice, such partial Gröbner bases often suffice to confirm the ideal membership of a given polynomial $f \in K\langle X\rangle$ by reducing it to zero. To this end, we discuss procedures to enumerate (partial) Gröbner bases in the following chapter.

Chapter 4

Gröbner bases computations in $K\langle X\rangle$

After recalling the main results of the theory of Gröbner bases in the noncommutative setting in the previous chapter, we now want to use the insights gained there to derive procedures to actually compute (partial) Gröbner bases in $K\langle X\rangle$. However, since there are ideals in $K\langle X\rangle$ that do not have a finite Gröbner basis, we cannot expect any of these procedures to terminate in general and have to content ourselves with enumeration procedures.

In the commutative case, Gröbner bases computations are either based on the classical Buchberger algorithm [Buc65], which uses the commutative analogue of our polynomial reduction relation, or on Faugère's F4 algorithm [Fau99], which relies on linear algebra to do polynomial reduction. Both of these algorithms can be generalised to the setting of noncommutative polynomials. For Buchberger's algorithm this was first done in 1986 by Mora [Mor86] and in case of Faugère's F4 algorithm this was done by Xiu in 2012 [Xiu12]. Another approach to the computation of Gröbner bases in $K\langle X\rangle$, only developed recently by La Scala and Levandovskyy [LSL09, LSL13], is to embed the noncommutative polynomial ring in a larger commutative ring, the so-called Letterplace ring, and execute the Gröbner bases computations there, providing the possibility to reuse the routines for commutative structures.

In this thesis, we focus on the first two approaches, i.e. the generalisation of Buchberger's algorithm and Faugère's F4 algorithm. But before discussing these algorithms, we first present an interreduction procedure in Section 4.1 that enables us to detect and delete redundant generators of a finite system of generators of an ideal. Following upon that, we elaborate on Bergman's famous Diamond Lemma [Ber78], which generalises Buchberger's criterion [Buc65] to the noncommutative case and allows us to state the noncommutative version of the Buchberger algorithm as an almost immediate consequence. This is done in Section 4.2. In Section 4.3, we connect the reduction of a finite set of polynomials with a matrix normal form computation. Using this relation, we can then state the F4 algorithm in $K\langle X\rangle$. Following upon that, we extend Buchberger's algorithm as well as the F4 algorithm to keep track of cofactors during the computation. Provided that we can verify the membership of a given polynomial $f \in K\langle X\rangle$ in an ideal $I \subseteq K\langle X\rangle$, this enables us to produce a cofactor representation of f with respect to the generators of I. To end this chapter, we present some basic optimisation strategies for Gröbner bases computations in Section 4.5. In particular, we discuss deletion criteria, such as a noncommutative version of the chain criterion, that allow us to delete unnecessary ambiguities during a

Gröbner basis computation, as well as strategies to remove redundant generators of a (partial) Gröbner basis. Furthermore, we also state the Faugère-Lachartre elimination algorithm [FL10], which can be used to efficiently compute the reduced row echelon form of matrices appearing during the execution of F4.

Throughout this chapter, we let K be a computable field. Also, recall that we assume that the set of indeterminates X is finite and that we fixed a monomial ordering $\preceq$ on $\langle X \rangle$. Furthermore, we note that the scope of programming instructions such as **if**, **while**, **for**, etc. in our pseudocode is indicated by a uniform indentation of the affected statements, as also done in Python for example.

4.1 Interreduction

Given a finite system of generators $F \subseteq K\langle X \rangle$ of an ideal I, it is often useful to delete redundant generators, which are just the multiple of another element in F or can be expressed as a linear combination of other generators, as a preprocessing step in order to avoid unnecessary computations and costly zero reductions during a Gröbner basis computation. One way of detecting and deleting such redundant generators is by transforming the initial set F into an interreduced set $\tilde{F}$, which generates the same ideal, i.e. $(F) = (\tilde{F})$. This can be achieved by the following algorithm.

Algorithm 1 Interreduce

Input: $F = \{f_1, \ldots, f_n\} \subseteq K\langle X \rangle$
Output: $\tilde{F} \subseteq K\langle X \rangle \setminus \{0\}$ such that $\tilde{F}$ is interreduced and $(F) = (\tilde{F})$

1: $\tilde{F} = F$ ▷ make a copy so that we do not modify F
2: $i = 1$
3: **while** $i \leq n$ **do**
4: compute a normal form $\tilde{f}_i'$ of $\tilde{f}_i \in \tilde{F}$ w.r.t. to $\to_{\tilde{F} \setminus \{\tilde{f}_i\}}$
5: **if** $\tilde{f}_i' = 0$ **then**
6: set $\tilde{f}_i = 0$
7: $i = i + 1$
8: **else if** $\tilde{f}_i' \neq \tilde{f}_i$ **then**
9: set $\tilde{f}_i = \tilde{f}_i'$
10: i = 1
11: **else**
12: $i = i + 1$
13: **return** $\tilde{F} \setminus \{0\}$

Theorem 4.1. *Let $F \subseteq K\langle X \rangle$ and let $\tilde{F} \subseteq K\langle X \rangle \setminus \{0\}$ be the result of applying Algorithm 1 to F. Then, $\tilde{F}$ is interreduced and $(F) = (\tilde{F})$.*

Proof. First we prove the termination of Algorithm 1 by showing that the parameter i is set to 1 only finitely many times in step 10. Assume that this is not the case. Then, we must have $\tilde{f}_i' \neq 0$ and $\tilde{f}_i' \neq f_i$ infinitely often. More precisely, this must hold infinitely often for a particular value

of i, say i_0. We denote the computed normal forms by $\tilde{f}_{i_0}^{(j)}$ for $j \in \mathbb{N}$ and $\tilde{f}_{i_0}^{(0)} = \tilde{f}_{i_0} = f_{i_0}$. Since we obtain $\tilde{f}_{i_0}^{(j+1)}$ as a normal form of $\tilde{f}_{i_0}^{(j)}$ and since $\tilde{f}_{i_0}^{(j+1)} \neq 0$ and $\tilde{f}_{i_0}^{(j+1)} \neq \tilde{f}_{i_0}^{(j)}$ by assumption, this yields the infinite sequence $\tilde{f}_{i_0}^{(0)} \gg \tilde{f}_{i_0}^{(1)} \gg \ldots$, which is a contradiction to Proposition 3.18. Furthermore, the output $\tilde{F}$ of Algorithm 1 is interreduced, since we set i to 1 whenever we replace an element $\tilde{f}_i$ by a nonzero normal form $\tilde{f}_i'$, and thereby, assure that in the end all elements $\tilde{f} \in \tilde{F}$ are in normal form with respect to $\to_{\tilde{F}\setminus\{f\}}$. Finally, the property $(F) = (\tilde{F})$ follows from Lemma 3.47. □

Note that for an arbitrary set $F \subseteq K\langle X\rangle$, the output of Algorithm 1 when given F as input is not necessarily unique. It depends on the (in general not unique) normal forms computed during the execution of the algorithm. However, given a finite Gröbner basis G of an ideal $I \subseteq K\langle X\rangle$ as input, Algorithm 1 computes an interreduced Gröbner basis G' of I, which is essentially unique.

Corollary 4.2. *Let $I \subseteq K\langle X\rangle$ be an ideal and let G be a finite Gröbner basis of I. Furthermore, let $\tilde{G}$ be the result of applying Algorithm 1 to G. Then, $\{\frac{g}{\mathrm{lc}(g)} \mid g \in \tilde{G}\}$ is the unique reduced Gröbner basis of I.*

Proof. We denote $G' = \{\frac{g}{\mathrm{lc}(g)} \mid g \in \tilde{G}\}$. It is clear that all elements of G' are monic. Furthermore, according to Theorem 4.1, $\tilde{G}$ and consequently also G' are interreduced and we have $(G) = (\tilde{G}) = (G')$. Proposition 3.46 tells us that given a Gröbner basis G as input, Algorithm 1 maintains the Gröbner basis property at every step. In particular, this means that the output $\tilde{G}$ and therefore also G' are Gröbner bases of I.

To prove uniqueness, we let G_1 and G_2 be two finite Gröbner bases of I and we denote

$$G_1' = \{\frac{g}{\mathrm{lc}(g)} \mid g \in \mathrm{Interreduce}(G_1)\} \text{ and}$$
$$G_2' = \{\frac{g}{\mathrm{lc}(g)} \mid g \in \mathrm{Interreduce}(G_2)\}.$$

We show that $G_1' = G_2'$. In particular, we prove $G_1' \subseteq G_2'$. The other inclusion will follow from the symmetry of our argument. To this end, let $g_1 \in G_1'$. Since G_1' and G_2' are both Gröbner bases of I there exist $g_2, g_2' \in G_2'$ such that $\mathrm{lm}(g_2) \mid \mathrm{lm}(g_1) \mid \mathrm{lm}(g_2')$ but because G_2' is interreduced this is only possible if $g_2 = g_2'$. Consequently, we must have $\mathrm{lm}(g_1) = \mathrm{lm}(g_2)$. If $g_1 = g_2$ we are done. Otherwise, we know that we can reduce $g_1 - g_2$ to zero by G_1'. In fact, since $\mathrm{lm}(g_1 - g_2) \prec \mathrm{lm}(g_1)$ we can reduce $g_1 - g_2$ to zero by $G_1' \setminus \{g_1\}$. But since g_1 is in normal form with respect to $G_1' \setminus \{g_1\}$ this is only possible if g_2 can be reduced to g_1 by $G_1' \setminus \{g_1\}$, which yields $g_2 \gg g_1$. By considering G_2' instead of G_1' we get by the same argument that $g_1 \gg g_2$ but both inequalities together lead to a contradiction. Hence, we must have $g_1 = g_2 \in G_2'$. □

As a first application of Algorithm 1, we consider the following example.

Example 4.3. Let $K = \mathbb{Q}$ and $X = \{a, b, c, d\}$. We equip $K\langle X\rangle$ with $\preceq_{deglex}$ where we order the indeterminates as $a \prec_{lex} b \prec_{lex} c \prec_{lex} d$ and consider the set of polynomials $F =$

$\{f_1, f_2, f_3, f_4, f_5\} \subseteq K\langle X\rangle$ with

$$f_1 = aca - a, \qquad f_2 = db - ac, \qquad f_3 = bdb - b$$
$$f_4 = ac - db, \qquad f_5 = ba.$$

In the following, we apply Algorithm 1 to F. To this end, we copy F to $\tilde{F}$ and set $i = 1$.

$i = 1$: $\tilde{f}_1 = aca - a$ is irreducible with respect to $\to_{\tilde{F}\setminus\{\tilde{f}_1\}}$. Consequently, we cannot do any reductions and have to increase i by one.

$i = 2$: $\tilde{f}_2 = db - ac$ can be reduced to zero by $\tilde{f}_4 = ac - db$. Hence, we set $\tilde{f}_2 = 0$ and increase i by one.

$i = 3$: $\tilde{f}_3 = bdb - b$ can be reduced to b by $\tilde{F} \setminus \{\tilde{f}_3\}$, which is a new nonzero normal form. So, we set $\tilde{f}_3 = b$ and reset i to one.

$i = 1$: $\tilde{f}_1 = aca - a$ is still irreducible. Hence, we increase i by one.

$i = 2$: $\tilde{f}_2$ is already zero. So, we have to increase i by one.

$i = 3$: $\tilde{f}_3 = b$ is irreducible. So, we increase i by one.

$i = 4$: $\tilde{f}_4 = ac - db$ can be reduced to ac by $\tilde{f}_3 = b$. Hence, we set $\tilde{f}_4 = ac$ and reset i to one.

$i = 1$: $\tilde{f}_1 = aca - a$ can be reduced to a by $\tilde{f}_4 = ac$. So, we set $\tilde{f}_1 = a$ and reset i to one.

$i = 1$: $\tilde{f}_1 = a$ is now irreducible. So, we increase i by one.

$i = 2$: $\tilde{f}_2$ is already zero. So, we have to increase i by one.

$i = 3$: $\tilde{f}_3 = b$ is still irreducible. Hence, we increase i by one.

$i = 4$: $\tilde{f}_4 = ac$ can be reduced to zero by $\tilde{f}_1 = a$. So, we set $\tilde{f}_4 = 0$ and increase i by one.

$i = 5$: $\tilde{f}_5 = ba$ can be reduced to zero either by $\tilde{f}_1 = a$ or by $\tilde{f}_3 = b$. Hence, we set $\tilde{f}_5 = 0$ and increase i by one.

Since now $i > |\tilde{F}|$, the algorithm terminates and returns $\tilde{F} \setminus \{0\} = \{a, b\}$.

4.2 Buchberger algorithm

The goal of this section is to describe a procedure for computing (partial) Gröbner bases in $K\langle X\rangle$. Therefore, we first of all have to be able to algorithmically decide whether a given set $G \subseteq K\langle X\rangle$ is a Gröbner basis. According to the definition, we would have to check whether $\to_G$ is confluent, however, in practice this is not possible since it would involve testing infinitely many cases $g_1 \,{}_G\!\overset{*}{\leftarrow} f \overset{*}{\to}_G g_2$ whether $g_1 \downarrow_G g_2$ holds. In the commutative case, Buchberger was able to reduce the number of these checks to a finite amount. This is now known as Buchberger's criterion. In the noncommutative setting, Bergman's Diamond Lemma does a similar job. To be able to state the Diamond Lemma, we first define the notion of *ambiguities* as done in [Ber78]. Note that ambiguities correspond to compositions in [Bok76].

Definition 4.4. Let $G \subseteq K\langle X\rangle$ and let $f, g \in G\setminus\{0\}$ be such that $\mathrm{lm}(f) = AB$ and $\mathrm{lm}(g) = BC$ for some words $A, B, C \in \langle X\rangle \setminus \{1\}$. Then, we call the tuple

$$(ABC, A, C, f, g)$$

an *overlap ambiguity* of G. It is called *resolvable*, if the *S-polynomial* $h_{1,f,C}(ABC) - h_{A,g,1}(ABC)$ can be reduced to 0 by G, i.e.

$$h_{1,f,C}(ABC) - h_{A,g,1}(ABC) \xrightarrow{*}_G 0.$$

Definition 4.5. Let $G \subseteq K\langle X\rangle$ and let $f, g \in G \setminus \{0\}$ be such that $f \neq g$, $\mathrm{lm}(f) = ABC$ and $\mathrm{lm}(g) = B$ for some words $A, B, C \in \langle X\rangle$. Then, we call the tuple

$$(ABC, A, C, f, g)$$

an *inclusion ambiguity* of G. It is called *resolvable*, if the *S-polynomial* $h_{1,f,1}(ABC) - h_{A,g,C}(ABC)$ can be reduced to 0 by G, i.e.

$$h_{1,f,1}(ABC) - h_{A,g,C}(ABC) \xrightarrow{*}_G 0.$$

We denote the set of all ambiguities of G by amb_G.

Remark. Note that a polynomial $f \in G$ can have an overlap ambiguity with itself but it cannot have an inclusion ambiguity with itself. Furthermore, two elements $f, g \in G \setminus \{0\}$ can give rise to several ambiguities but only finitely many.

Later, we need the notion of the *degree* of an ambiguity.

Definition 4.6. Let $G \subseteq K\langle X\rangle$ and let $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ be an ambiguity of two polynomials $f, g \in G \setminus \{0\}$. We call $|ABC|$ the *degree* of a.

For a set $G \subseteq K\langle X\rangle$ and an ambiguity $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ of two polynomials $f, g \in G \setminus \{0\}$, we denote the S-polynomial corresponding to a by $\mathrm{sp}(a)$, i.e.

$$\mathrm{sp}(a) \coloneqq h_{1,f,C}(ABC) - h_{A,g,1}(ABC),$$

in case that a is an overlap ambiguity and

$$\mathrm{sp}(a) \coloneqq h_{1,f,1}(ABC) - h_{A,g,C}(ABC),$$

respectively, if a is an inclusion ambiguity. It is easy to see that $\mathrm{sp}(a) \in (G)$.

Proposition 4.7. *Let $G \subseteq K\langle X\rangle$ and $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$. Then, $\mathrm{sp}(a) \in (G)$.*

Proof. We prove the statement for the case that a is an inclusion ambiguity. The case for an overlap ambiguity works analogously. By Theorem 3.35, we have $ABC - h_{1,f,1}(ABC) \in (G)$ and $ABC - h_{A,g,C}(ABC) \in (G)$. Hence, also

$$\mathrm{sp}(a) = h_{1,f,1}(ABC) - h_{A,g,C}(ABC) = -(ABC - h_{1,f,1}(ABC)) + (ABC - h_{A,g,C}(ABC)) \in (G).$$

□

Example 4.8. Let K be an arbitrary field and $X = \{x, y\}$. We consider $G = \{g_1, g_2\} \subseteq K\langle X\rangle$ with $g_1 = xyx - x$ and $g_2 = yx - xy$. If we use $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y$, then g_1 and g_2 have an inclusion ambiguity

$$a_1 = (xyx, x, 1, g_1, g_2)$$

and an overlap ambiguity

$$a_2 = (yxyx, y, yx, g_2, g_1).$$

Additionally, g_1 has an overlap ambiguity with itself:

$$a_3 = (xyxyx, xy, yx, g_1, g_1).$$

The ambiguity a_3 is resolvable, since

$$\mathrm{sp}(a_3) = h_{1,f_1,yx}(xyxyx) - h_{xy,f_1,1}(xyxyx) = xyx - xyx = 0.$$

Also a_2 is resolvable. Here we have

$$\mathrm{sp}(a_2) = h_{1,f_2,yx}(yxyx) - h_{y,f_1,1}(yxyx) = xyyx - yx \to_{1,f_2,1} xyyx - xy \to_{xy,f_2,1} xyxy - xy \to_{1,f_1,y} 0.$$

The ambiguity a_1, however, is not resolvable because

$$\mathrm{sp}(a_1) = h_{1,f_1,1}(xyx) - h_{x,f_2,1}(xyx) = x - xxy$$

is irreducible with respect to $\to_G$.

Later, we also need a weaker notion of resolvability. Hence, following [Ber78], we introduce the notion of $\preceq$-*resolvability*.

Definition 4.9. Let $G \subseteq K\langle X\rangle$ and let $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$. We say that a is $\preceq$-*resolvable* if its S-polynomial is contained in the K-vector space $I_{G,ABC}$ generated by

$$\{wg'w' \mid g' \in G,\, w, w' \in \langle X\rangle \text{ such that } w\,\mathrm{lm}(g')w' \prec ABC\}.$$

Remark. It is easy to see that resolvability implies $\preceq$-resolvability.

Given a set $G \subseteq K\langle X\rangle$, the resolvability of an ambiguity $(ABC, A, C, f, g) \in \mathrm{amb}_G$ implies the (local) confluence condition on the two possible ways that the monomial ABC can be reduced by f and g, respectively. It turns out that it is enough to only check all these cases to determine whether $\to_G$ is confluent, and consequently, whether G is a Gröbner basis. Also, if G is a Gröbner basis, the notions of resolvability and $\preceq$-resolvability coincide.

Theorem 4.10. *(Diamond Lemma)*
Let $G \subseteq K\langle X\rangle$. Then, the following conditions are equivalent.

1. *All ambiguities of G are resolvable.*
2. *All ambiguities of G are $\preceq$-resolvable.*

3. All elements of $K\langle X\rangle$ have a unique normal form under $\to_G$.

4. G is a Gröbner basis of (G).

Proof. See [Ber78, Theorem 1.2] □

Hence, according to Theorem 4.10 it suffices to check whether all ambiguities of a given set $G \subseteq K\langle X\rangle$ are resolvable to determine whether G is a Gröbner basis.

Example 4.11. Recall that in Example 3.48 we considered the principal ideal $(g) \subseteq K\langle x, y\rangle$ with $g = xyx - xy$ over an arbitrary field K. Furthermore, we used $\preceq_{deglex}$ where we ordered the indeterminates as $x \prec_{lex} y$. Our claim was that the reduced Gröbner basis of (g) is given by

$$G = \{xy^i x - xy^i \mid i \in \mathbb{N}\}$$

and we postponed the verification that G is indeed a Gröbner basis. Using the Diamond Lemma, we can now finish this proof. To this end, we denote $g_i = xy^i x - xy^i \in G$ and note that all ambiguities of G are overlap ambiguities of the form

$$a_{i,j} = (xy^i xy^j x, xy^i, y^j x, g_i, g_j)$$

for g_i, $g_j \in G$. The corresponding S-polynomials are given by

$$\mathrm{sp}(a_{i,j}) = xy^{i+j}x - xy^i xy^j,$$

which can be reduced to zero as follows

$$xy^{i+j}x - xy^i xy^j \;\to_{1,g_i,y^j}\; xy^{i+j}x - xy^{i+j} \;\to_{1,g_{i+j},1}\; 0.$$

This shows that all ambiguities of G are resolvable, and consequently, that G is indeed a Gröbner basis of (g).

In case that G is finite, the following algorithm enables us to decide algorithmically whether G is a Gröbner basis.

Algorithm 2 CheckResolvability

Input: a finite set $G \subseteq K\langle X\rangle$
Output: $spol \subseteq K\langle X\rangle$ such that $spol = \emptyset$ if and only if G is a Gröbner basis of (G)

```
1: spol = ∅
2: foreach a ∈ amb_G do
3:     compute a normal form s' of sp(a) w.r.t. to →_G
4:     if s' ≠ 0 then
5:         spol = spol ∪ {s'}
6: return spol
```

Proposition 4.12. *Let $G \subseteq K\langle X\rangle$ be finite and let $spol \subseteq K\langle X\rangle$ be the result of applying Algorithm 2 to G. Then, $spol = \emptyset$ if and only if G is a Gröbner basis of (G).*

Proof. First, we note that since G is finite, also the set amb_G of all ambiguities of G is finite, which implies the termination of Algorithm 2. So, we can proceed to prove the claim of Proposition 4.12. If $spol = \emptyset$, then all ambiguities of G are resolvable. Hence, Theorem 4.10 yields that G is a Gröbner basis of (G). Conversely, if G is a Gröbner basis of (G), then all elements of (G) can be reduced to 0 by G according to Theorem 3.38. Since all S-polynomials are elements of (G), the claim follows. □

If we apply Algorithm 2 to a finite system of generators $F \subseteq K\langle X\rangle$ of an ideal I, we usually do not expect to obtain $spol = \emptyset$. Typically, there will be some ambiguities which are not resolvable. By adding the reduced S-polynomials corresponding to these ambiguities to the set F, we enlarge our reduction system and make these ambiguities resolvable. Of course, then we have to check whether all ambiguities involving the newly added elements are resolvable. If they are, then we have successfully computed a Gröbner basis of I. If not, then we simply repeat this procedure. These instructions basically already describe Buchberger's algorithm for noncommutative polynomials.

Algorithm 3 Buchberger

Input: a finite set $F \subseteq K\langle X\rangle$

Output if the algorithm terminates: $G \subseteq K\langle X\rangle$ such that G is a Gröbner basis of (F)

```
1:  G = F
2:  spol = CheckResolvability(G)
3:  while spol ≠ ∅ do
4:      while spol ≠ ∅ do
5:          select f ∈ spol
6:          spol = spol \ {f}
7:          compute a normal form f' of f w.r.t. to →_G
8:          if f' ≠ 0 then
9:              G = G ∪ {f'}
10:     spol = CheckResolvability(G)
11: return G
```

Theorem 4.13. *Let $F \subseteq K\langle X\rangle$ be a finite set, denote $G_0 = F$ and for $n \in \mathbb{N}$, let G_n be the result of Algorithm 3 after n iterations of the outer **while** loop given F as input. Then, $G = \bigcup_{i\geq 0} G_n$ is a Gröbner basis of (F). In this sense, Algorithm 3 enumerates a Gröbner basis G of (F).*

Proof. To prove that $G = \bigcup_{i\geq 0} G_n$ is a Gröbner basis of (F), we first note that it follows from Proposition 4.7 and Theorem 3.35 that all normal forms f', which are added to G during the execution of Algorithm 3, are elements of (F). Since also $F \subseteq G$, this implies $(G) = (F)$. Hence, according to Theorem 4.10 it remains to show that all ambiguities of G are resolvable. To this end, let $a = (ABC, A, C, g_i, g_j) \in \mathrm{amb}_G$ be an ambiguity of two elements $g_i, g_j \in G$. Since $G_n \subseteq G_{n+1}$ for all $n \geq 0$, there exists $N \geq 0$ such that $g_i, g_j \in G_N$. If $\mathrm{sp}(a)$ can be reduced to zero by G_N we are done. Otherwise, we add the nonzero normal form of $\mathrm{sp}(a)$ to G_N during the

N-th iteration of Algorithm 3. Hence, in any way $\mathrm{sp}(a)$ can be reduced to zero by $G_{N+1} \subseteq G$ and therefore a is resolvable. □

Given a finite set $F \subseteq K\langle X\rangle$ as input, we can in fact show that Algorithm 3 always terminates in case that the ideal (F) admits a finite Gröbner basis.

Proposition 4.14. *Let $F \subseteq K\langle X\rangle$ be a finite set such that (F) has a finite Gröbner basis. Then, Algorithm 3 terminates when given F as input and returns a Gröbner basis of (F).*

Proof. Let $G' = \{g_1, \ldots, g_m\} \subseteq K\langle X\rangle$ be a finite Gröbner basis of (F). Furthermore, for $n \geq 0$ let G_n be as in Theorem 4.13. Then, $\tilde{G} = \bigcup_{i\geq 0} G_n$ is also a Gröbner basis of (F) according to Theorem 4.13. Hence, condition 3 of Theorem 3.38 tells us that for every $f \in (F)$ there exist $g \in G'$ and $\tilde{g} \in \tilde{G}$ such that $\mathrm{lm}(g) \mid \mathrm{lm}(f)$ and $\mathrm{lm}(\tilde{g}) \mid \mathrm{lm}(f)$, respectively. So, in particular, there exist $\tilde{g}_i \in \tilde{G}$ such that $\mathrm{lm}(\tilde{g}_i) \mid \mathrm{lm}(g_i)$ for $1 \leq i \leq m$. Again by condition 3 of Theorem 3.38, this shows that the set $\{\tilde{g}_1, \ldots, \tilde{g}_m\} \subseteq \tilde{G}$ is already a Gröbner basis of (F). Since $\tilde{g}_i \in G_{n_i}$ for some $n_i \geq 0$, it follows that G_N with $N = \max\{n_1, \ldots, n_m\}$ is a Gröbner basis of (F). But then, $\mathrm{CheckResolvability}(G_N) = \emptyset$ and hence the algorithm terminates and returns G_N. □

In the following, we collect some remarks on Algorithm 3. If (F) has no finite Gröbner basis, Algorithm 3 never terminates. Hence, in practice, we usually add some additional constraints to guarantee termination. One way of doing this is by keeping track of the number of iterations of the outer **while** loop and stopping the algorithm after a prescribed number of iterations. Another way is to impose an upper limit on the degree of the ambiguities which are considered. If during the execution of Algorithm 3 an ambiguity arises that has a larger degree than the designated limit, this ambiguity is simply ignored. Both approaches ensure the termination of the algorithm but in both cases we might only obtain a partial Gröbner basis of (F). This might even be the case if (F) has a finite Gröbner basis but we stop the algorithm too early or impose a degree limit which is too low.

One might also wonder why we compute a normal form of each $f \in \mathit{spol}$ in step 7 of Algorithm 3 although we have already reduced all elements in *spol* to normal form in the CheckResolvability subroutine. This is mainly due to performance reasons. In fact, only doing one of those two reductions would suffice for the algorithm to work correctly, however, in practice both reduction steps provide certain advantages. Most of the reductions are done in the first reduction step in the CheckResolvability subroutine and this step can be implemented in a very efficient way (for example this can be done in parallel). Thus, by doing the main work there we are able to save quite a lot of time. In comparison, the second reduction step, even if not implemented efficiently, does not really contribute to the overall time needed simply because only few reductions are executed in this step. However, it provides the advantage that we might be able to reduce some elements in *spol* to zero which could not be reduced to zero in CheckResolvability since the set G might have been enlarged in the meantime by some elements in *spol*. This helps us to keep the set G as small as possible.

Also, note that Line 5 of Algorithm 3 is intentionally phrased in an ambiguous way. This provides the opportunity to apply different strategies on how to choose $f \in \mathit{spol}$, which can have a huge impact on the performance of the algorithm in practice. Often, applying a selection strategy, which always chooses an S-polynomials with minimal leading monomial, works quite

well. However, when working with different selection strategies, one usually has to take care that a so-called *fair* selection strategy is used. Basically, this ensures that every S-polynomial, that is generated at some point, eventually gets processed and reduced. For a more in-depth discussion on fair selection strategies, we refer to [Mor16, Section 47.6.3]. Note that, in case of Algorithm 3, every selection strategy is fair as new ambiguities are only generated when all old ambiguities, and consequently, all old S-polynomials have been processed.

To end this section, we use Algorithm 3 to compute a Gröbner basis.

Example 4.15. Let $K = \mathbb{Q}$ and $X = \{x, y\}$. We consider $K\langle X\rangle$ equipped with $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y$ and want to compute a Gröbner basis of the ideal generated by $F = \{f_1, f_2, f_3, f_4\} \subseteq K\langle X\rangle$, with

$$f_1 = yyx + x, \qquad f_2 = y + 1, \qquad f_3 = xy - x, \qquad f_4 = xyx + xx.$$

We start by setting $G = F$. According to Algorithm 3, we first have to execute the commands in the CheckResolvability subroutine. So, we compute all ambiguities of G. It is not hard to see that amb_G consists of the overlap ambiguities

$$\begin{aligned} &a_{13} = (yyxy, yy, y, f_1, f_3), \quad && a_{14} = (yyxyx, yy, yx, f_1, f_4), \quad && a_{31} = (xyyx, x, yx, f_3, f_1),\\ &a_{43} = (xyxy, xy, y, f_4, f_3), && a_{44} = (xyxyx, xy, yx, f_4, f_4) \end{aligned}$$

and the inclusion ambiguities

$$\begin{aligned} &a_{12} = (yyx, 1, yx, f_1, f_2), \quad && a'_{12} = (yyx, y, x, f_1, f_2), \quad && a_{32} = (xy, x, 1, f_3, f_2),\\ &a_{42} = (xyx, x, x, f_4, f_2), && a'_{43} = (xyx, 1, x, f_4, f_3). \end{aligned}$$

Now, we have to do the first reduction, which is done in the CheckResolvability subroutine. In particular, we have to compute the normal form of $\mathrm{sp}(a)$ with respect to G for each $a \in \mathrm{amb}_G$. This yields

$$\begin{aligned} \mathrm{sp}(a_{13}) &= -yyx - xy \ \to_{1,f_1,1} \ x - xy \ \to_{1,f_3,1} \ 0,\\ \mathrm{sp}(a_{14}) &= yyxx - xyx \ \to_{1,f_1,x} \ -xyx - xx \ \to_{1,f_4,1} \ 0,\\ \mathrm{sp}(a_{31}) &= xyx + xx \ \to_{x,f_2,x} \ 0,\\ \mathrm{sp}(a_{43}) &= -xyx - xxy \ \to_{x,f_2,x} \ -xxy + xx \ \to_{xx,f_2,1} \ 2xx,\\ \mathrm{sp}(a_{44}) &= xyxx - xxyx \ \to_{x,f_2,xx} \ -xxx - xxyx \ \to_{xx,f_2,x} \ 0,\\ \mathrm{sp}(a_{12}) &= yx - x \ \to_{1,f_2,x} \ 2x,\\ \mathrm{sp}(a'_{12}) &= yx - x \ \to_{1,f_2,x} \ 2x,\\ \mathrm{sp}(a_{32}) &= 2x,\\ \mathrm{sp}(a_{42}) &= 0,\\ \mathrm{sp}(a'_{43}) &= -2xx. \end{aligned}$$

The only nonzero normal forms that we obtain are $2x$, $2xx$ and $-2xx$, and therefore, the CheckResolvability subroutine returns the set $spol = \{2x, 2xx, -2xx\}$. Since this set is nonempty, we

have to enter the **while** loops in Algorithm 3 and keep reducing the elements in *spol* individually. Now, the selection strategy comes into play and in this example, our selection strategy is to always choose an S-polynomial with minimal leading monomial. So, we first process $2x$, which we also immediately remove from *spol*. Since $2x$ is already a normal form with respect to $\to_G$, we have to add it to G. Hence, we obtain

$$G = \{f_1, f_2, f_3, f_4, f_5\},$$

with $f_5 = 2x$. Now, we are in the situation that several polynomials in $spol = \{2xx, -2xx\}$ have the same minimal leading monomial. In this case, we can choose one of them randomly. We select $2xx$ and remove it from *spol*. Because we have added $2x$ to G in the previous step, $2xx$ can be reduced to zero by G. Hence, in this step we do not have to add anything to G. Finally, we process $-2xx$, which can also be reduced to zero by G. So, again, nothing has to be added to G. As *spol* is now empty, we go back into the CheckResolvability subroutine, and compute all ambiguities of G. Of course, we do not have to recompute ambiguities that we have already processed in a previous step. Hence, we only have to consider ambiguities which contain at least one element of G that has been added during the last iteration. In this case, we obtain the following four inclusion ambiguities

$$\begin{aligned} &a_{15} = (yyx, yy, 1, f_1, f_5), \qquad a_{35} = (xy, 1, y, f_3, f_5), \qquad a_{45} = (xyx, 1, yx, f_4, f_5),\\ &a'_{45} = (xyx, xy, 1, f_4, f_5). \end{aligned}$$

As before, we now have to reduce $\mathrm{sp}(a)$ by G for each $a \in \mathrm{amb}_G$. This yields

$$\begin{aligned} \mathrm{sp}(a_{15}) &= -x \ \to_{1,f_5,1}\ 0,\\ \mathrm{sp}(a_{35}) &= x \ \to_{1,f_5,1}\ 0,\\ \mathrm{sp}(a_{45}) &= -xx \ \to_{1,f_5,1}\ 0,\\ \mathrm{sp}(a'_{45}) &= -xx \ \to_{1,f_5,1}\ 0. \end{aligned}$$

We can see that all S-polynomials reduce to 0. Hence, CheckResolvability returns $spol = \emptyset$, and therefore,

$$G = \{f_1, f_2, f_3, f_4, 2x\},$$

is a Gröbner basis of (f_1, f_2, f_3, f_4).

4.3 F4

In contrast to Buchberger's algorithm where we reduce only one polynomial at a time, the main idea of the F4 algorithm is to reduce several polynomials by a list of polynomials simultaneously. This is done by representing polynomials in terms of a matrix and computing a reduced row echelon form of this matrix, cf. [Fau99, Xiu12].

As a first step towards formulating this procedure in our setting of noncommutative polynomials, we elaborate on the connection between a finite set of elements in $K\langle X\rangle$ and a matrix over K. To this end, we recall what the linear span of a set of polynomials is.

Definition 4.16. Let $P \subseteq K\langle X\rangle$ be a set of polynomials. The linear span of P over K, denoted by $\mathrm{span}_K(P)$, is the set of all K-linear combinations of elements in P, i.e.,

$$\mathrm{span}_K(P) \coloneqq \{\sum_{i=1}^{m} c_i p_i \mid m \in \mathbb{N},\, c_i \in K,\, p_i \in P\} \subseteq K\langle X\rangle.$$

As done in [Xiu12, Definition 5.4.1], we relate polynomials and matrices via an isomorphism.

Definition 4.17. Let $T = \{t_1, \ldots, t_m\} \subseteq \langle X\rangle$ be a set of monomials such that $t_1 \succ \cdots \succ t_m$. Furthermore, let K^m be the vector space of dimension m over K equipped with the canonical basis $e_1 \ldots, e_m \in K^m$. We consider T as a subset of $K\langle X\rangle$ and define an isomorphism

$$\varphi_T : K^m \to \mathrm{span}_K(T)$$

given by $\varphi_T(e_i) = t_i$ for $i = 1, \ldots, m$.

For a fixed set $T = \{t_1, \ldots, t_m\} \subseteq \langle X\rangle$, this isomorphism allows us to identify every finite set of polynomials with support in T with a matrix, and conversely, every matrix with a finite set of polynomials with support in T.

Definition 4.18. Let $M \in K^{n\times m}$ be a matrix and $T = \{t_1, \ldots, t_m\} \subseteq \langle X\rangle$ be a set of monomials. We call the set $P_{M,T} \coloneqq \{\varphi_T(r_1), \ldots, \varphi_T(r_n)\} \subseteq \mathrm{span}_K(T)$, where $r_i \in K^m$ denotes the i-th row of M, a *polynomial form* of M with respect to T.

Conversely, given a finite set of polynomials $P = \{p_1, \ldots, p_n\} \subseteq K\langle X\rangle$ we let $T = \mathrm{supp}(P)$, and we call the matrix $M_P \in K^{n\times m}$ whose i-th row is given by $\varphi_T^{-1}(p_i)$ a *matrix form* of P.

Example 4.19. In this example we work over $\mathbb{Q}\langle x, y, z\rangle$ equipped with $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y \prec_{lex} z$. First, we consider $P = \{p_1, p_2, p_3\} \subseteq \mathbb{Q}\langle x, y, z\rangle$ with

$$p_1 = xxyz + 2xyy + x, \qquad p_2 = xyy - yz, \qquad p_3 = yz - 2x.$$

The matrix form M_P of P is given by

$$M_P = \begin{array}{c} \begin{array}{cccc} xxyz & xyy & yz & x \end{array} \\ \begin{pmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -2 \end{pmatrix} \end{array} \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array}.$$

Conversely, for $T = \{xxyz, xyy, yz, x\}$, the polynomial form $P_{M,T}$ of the matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -2 \end{pmatrix}$$

is $P_{M,T} = \{xxyz + 5x, xyy - 2x, yz - 2x\}$.

Recall that a matrix is in *row echelon form* if it satisfies the following conditions:

- all nonzero rows are above any rows of all zeros,
- the leftmost nonzero entry of a nonzero row (this entry is called the *pivot*) is 1,
- and the pivot of a nonzero row is strictly to the right of the pivot of the row above it.

If additionally all entries above each pivot are 0, then the matrix is in *reduced row echelon form*. We denote the unique reduced row echelon form of a matrix $M \in K^{n\times m}$ by $\mathrm{RRef}(M)$ and recall that it can be computed via Gauss-Jordan elimination. The main step in this algorithm can be described as follows. If M has two different rows r_i and r_j, such that the entry in r_j in the column of the pivot of r_i is nonzero, we subtract a multiple of r_i from r_j in order to set this entry of r_j to zero and thereby obtain a new row r'_j. In terms of polynomials this operation corresponds to a reduction of $\varphi_T(r_j)$ by $\varphi_T(r_i)$, i.e.

$$\varphi_T(r_j) \ \to_{1,\varphi_T(r_i),1} \ \varphi_T(r'_j),$$

for an appropriate set $T \subseteq \langle X\rangle$.

Hence, we can relate the polynomials in a finite set $P \subseteq K\langle X\rangle$ with the polynomials in $\tilde{P} = P_{\mathrm{RRef}(M_P),T} \subseteq K\langle X\rangle$ with $T = \mathrm{supp}(P)$, as follows. For every polynomial $p \in P$ there exists an element $\tilde{p} \in \tilde{P}$ such that either $p = c\tilde{p}$ for some nonzero constant $c \in K \setminus \{0\}$ or

$$p \ \to_{1,\varphi_T(r_1),1} \ \cdots \ \to_{1,\varphi_T(r_k),1} \ c\tilde{p},$$

where $r_1, \ldots, r_k \in K^m$ denote some rows of the intermediate matrices obtained during the computation of $\mathrm{RRef}(M_P)$. Note that the nonzero scaling factor c has to be added to account for the fact that during Gauss-Jordan elimination we are also allowed to scale rows, which does not correspond to a polynomial reduction but only a scaling of the corresponding polynomial.

Remark. If we start with a finite set of polynomials $P \subseteq K\langle X\rangle$ and want work with a polynomial form of $\mathrm{RRef}(M_P)$ with respect to $T = \mathrm{supp}(P)$, we omit the subscript T in $P_{\mathrm{RRef}(M_P),T}$ and simply write $P_{\mathrm{RRef}(M_P)}$.

Example 4.20. If we reconsider the matrices M and M_P from from Example 4.19, we can see that M is the reduced row echelon form of the matrix M_P, i.e. $M = \mathrm{RRef}(M_P)$. Hence, each polynomial in $P_{\mathrm{RRef}(M_P)} = \{xxyz + 5x, xyy - 2x, yz - 2x\}$ can be obtained by an interreduction of the polynomials in $P = \{p_1, p_2, p_3\}$, where $p_1 = xxyz + 2xyy + x, p_2 = xyy - yz, p_3 = yz - 2x$. In particular, we have

$$\begin{aligned} xxyz + 2xyy + x \ &\to_{1,p_2,1} \ xxyz + 2yz + x \ \to_{1,p_3,1} \ xxyz + 5x, \\ xyy - yz \ &\to_{1,p_3,1} \ xyy - 2x. \end{aligned}$$

This already indicates that we can reduce a finite set of polynomials $P \subseteq K\langle X\rangle$ by another set $G \subseteq K\langle X\rangle$ by computing a normal form of the matrix $M_{P\cup G}$. However, as soon as this matrix is set up we cannot add cofactors to the reduction as multiplying a row of $M_{P\cup G}$ by a monomial is not allowed. Furthermore, if G is infinite we cannot even form $M_{P\cup G}$. Hence, we first have to pick the finitely many elements of G which are actually needed for a reduction of the polynomials in P and multiply them by the corresponding cofactors. This is a purely

combinatorial problem which does not require to do any actual reductions and can be achieved as follows.

Algorithm 4 SymbolicPreprocessing

Input: a finite set $P \subseteq K\langle X\rangle$ and $G \subseteq K\langle X\rangle$
Output: $G' \subseteq \{agb \mid a, b \in \langle X\rangle,\ g \in G\}$

$G' = \emptyset$
$T = \operatorname{supp}(\operatorname{tail}(P))$
$done = \operatorname{lm}(P)$
while $T \neq \emptyset$ **do**
    select $t \in T$
    $T = T \setminus \{t\}$
    $done = done \cup \{t\}$
    if there exist $g \in G$, $a, b \in \langle X\rangle$ such that $h_{a,g,b}$ acts nontrivially on t **then**
        $G' = G' \cup \{agb\}$
        $T = T \cup (\operatorname{supp}(h_{a,g,b}(t)) \setminus done)$
return G'

Proposition 4.21. *Let $P \subseteq K\langle X\rangle$ be a finite set and $G \subseteq K\langle X\rangle$. Then, Algorithm 4 terminates given P and G as input.*

Proof. We note that $T = \operatorname{supp}(\operatorname{tail}(P))$ is finite because P is finite. Hence, we can consider $m = \max_{\preceq} T$ and as $\preceq$ is a well-ordering, there are only finitely many terms $m' \in \langle X\rangle$ such that $m' \prec m$. Since we remove one element $t \in T$ at every iteration of the **while** loop, we can prove termination by showing that the terms added to T are strictly smaller than m and that we never add a monomial to T that has already been processed. The first condition follows from the fact that $t \preceq m$ for every $t \in T$ and that $\operatorname{supp}(h_{a,g,b}(t))$ only contains terms which are strictly smaller than t. Furthermore, since we add every term $t \in T$ to the set *done* when processing it and enlarge T by $\operatorname{supp}(h_{a,g,b}(t)) \setminus done$, we ensure that we only add terms that have not yet been processed. □

Example 4.22. In this example, we let $K\langle X\rangle = \mathbb{Q}\langle x, y\rangle$ and equip it with $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y$. We want to apply Algorithm 4 to $G = \{f_1, f_2, f_3, f_4, f_5\} \subseteq \mathbb{Q}\langle x, y\rangle$, with

$$f_1 = yyx + x, \qquad f_2 = y + 1, \qquad f_3 = xy - x, \qquad f_4 = xyx + xx, \qquad f_5 = x$$

and

$$P = \{yyxy + xy, yyxy - yyx, yyxyx + xyx, yyxyx + yyxx, xyyx - xyx,\\ xyyx + xx, xyxy + xxy, xyxy - xyx, xyxyx + xxyx, xyxyx + xyxx\}.$$

To this end, we initialise $G' = \emptyset$, $done = \operatorname{lm}(P)$ and form the set $T = \operatorname{supp}(\operatorname{tail}(P))$, which is given by

$$T = \{xy, yyx, xyx, yyxx, xx, xxy, xxyx, xyxx\}.$$

The first term $t \in T$, that we process, is $t = xy$, which we remove from T and add to *done*. Since $\mathrm{lm}(f_3) = t$, we can use f_3 to reduce t and therefore add f_3 to G'. Note that we could have as well chosen f_2 or f_5. Then, we would had to add xf_2 or f_5y, respectively, to G'. But as we have chosen f_3, we now also have to add all monomials of f_3, that have not already been processed, to the set T. In this case, this is just x, so that we now have

$$T = T \cup \{x\}.$$

After that, we proceed and process $t = yyx$. As $\mathrm{lm}(f_1) = t$, we add f_1 to G'. Again, we have to add all monomials of f_1, which have not yet been processed, to T. This is only x, which is already in T. Hence, in this case, the set T does not change. For $t = xyx$, we add f_4 to G', for $t = yyxx$, we add f_1x and for $t = xx$ we add xf_5. In all these cases, we have to add nothing to T as all monomials appearing in the polynomials that we added to G' are either in *done* or already T. So, now we have

$$\begin{aligned} T &= \{xxy, xxyx, xyxx, x\}, \\ G' &= \{f_3, f_1, f_4, f_1x, xf_5\}, \\ \mathit{done} &= \mathrm{lm}(P) \cup \{xy, yyx, xyx, yyxx, xx\}. \end{aligned}$$

Next, we pick $t = xxy$, which can be reduced by f_3. Therefore, we add xf_3 to G' but have to add nothing to T as all monomials in xf_3 are already in *done*. For $t = xxyx$, $t = xyxx$ and $t = x$, respectively, we add f_5xyx, $xyxf_5$ and f_5, respectively, to G'. As also in these cases, we have to add nothing to T, this leaves us with

$$\begin{aligned} T &= \emptyset, \\ G' &= \{f_3, f_1, f_4, f_1x, f_5x, xf_3, f_5xyx, xyxf_5, f_5\}, \\ \mathit{done} &= \mathrm{lm}(P) \cup \{xy, yyx, xyx, yyxx, xx, xxy, xxyx, xyxx, x\}, \end{aligned}$$

and the algorithm terminates.

Algorithm 4 collects the finitely many elements of G that can be used to reduce the polynomials in P and multiplies them by the corresponding cofactors. Hence, a normal form computation of $M_{P \cup G'}$, where G' is the result of applying Algorihm 4 to P and G, corresponds to a reduction of the elements in P by G. However, note that we initialise the set *done* with $\mathrm{lm}(P)$. Hence, we do not pick reductors for the leading terms of the elements in P and can therefore actually only reduce the tails of the polynomials in P. Nevertheless, we shall see that we can use Algorithm 4 as a subroutine to check the resolvability of ambiguities, because, in this case, we can form the set P in such a way that we do not have to reduce the leading term of any element in P. To this end, we introduce the notion of the *critical pair* of an ambiguity.

Definition 4.23. Let $G \subseteq K\langle X \rangle$ and let $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ be an ambiguity of two elements $f, g \in G$. If a is an overlap ambiguity, we call the pair

$$\mathrm{cp}(a) \coloneqq (ABC - h_{1,f,C}(ABC), ABC - h_{A,g,1}(ABC))$$

the *critical pair* of a. If a is an inclusion ambiguity, we call the pair

$$\mathrm{cp}(a) \coloneqq (ABC - h_{1,f,1}(ABC), ABC - h_{A,g,C}(ABC))$$

the *critical pair* of a. We denote the set of all critical pairs of G by crit_G.

Since we need this term later, we extend the notion of the degree of an ambiguity to its critical pair.

Definition 4.24. Let $G \subseteq K\langle X\rangle$ and let $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ be an ambiguity of two elements $f, g \in G$. Then, the *degree* of $\mathrm{cp}(a)$ is given by the degree of a.

Remark. Let $G \subseteq K\langle X\rangle$ and let $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ be an ambiguity of two elements $f, g \in G$. If f and g are monic, then $\mathrm{cp}(a)$ can be written as

$$\mathrm{cp}(a) = (fC, Ag)$$

if a is an overlap ambiguity, respectively as

$$\mathrm{cp}(a) = (f, AgC)$$

if a is an inclusion ambiguity.

Considering the critical pair $\mathrm{cp}(a) = (f_a, g_a)$ of an ambiguity $a \in \mathrm{amb}_G$ we have $\mathrm{sp}(a) = g_a - f_a$ and $\mathrm{supp}(\mathrm{sp}(a)) \subseteq \mathrm{supp}(\mathrm{tail}(P))$, where $P = \{f_a, g_a\}$. Having this fact in mind, we can apply Algorithm 4 to P and G to find reductors $g_1, \ldots, g_m \in G$ for $\mathrm{sp}(a)$ and collect them in a set G'. We can then set up the matrix

$$M_{P\cup G'} = \begin{pmatrix} 1 & * & \cdots & \cdots & \cdots & * \\ 1 & * & \cdots & \cdots & \cdots & * \\ \hline & * & \cdots & \cdots & \cdots & * \\ & & \ddots & & & \vdots \\ & & & * & \cdots & * \end{pmatrix} \begin{matrix} f_a \\ g_a \\ g_1 \\ \vdots \\ g_m \end{matrix}$$

and reduce it to reduced row echelon form. At some point during this computation, w.l.o.g. as the first step, the first row gets subtracted from the second one and we get the following intermediate matrix.

$$\begin{pmatrix} 1 & * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & \cdots & * \\ \hline & * & \cdots & \cdots & \cdots & * \\ & & \ddots & & & \vdots \\ & & & * & \cdots & * \end{pmatrix} \begin{matrix} f_a \\ g_a - f_a \\ g_1 \\ \vdots \\ g_m \end{matrix}$$

So, we obtain a row corresponding to $\mathrm{sp}(a)$ and if we finish the reduction of $M_{P\cup G'}$, we end up with a row corresponding to a normal form $\mathrm{sp}(a)'$ of $\mathrm{sp}(a)$ with respect to G. Note that this is achieved without having to reduce the leading term of any element in P. The only remaining

task is to then find the row corresponding to $\mathrm{sp}(a)'$ in $\mathrm{RRef}(M_{P\cup G'})$. But this is also not too difficult, since $\mathrm{sp}(a)'$ has to either be zero or, if it is nonzero, it has to introduce a new leading monomial, i.e. a leading monomial that is not contained in $\mathrm{lm}(P \cup G')$. So, if $\mathrm{RRef}(M_{P\cup G'})$ contains a zero row, we have that $\mathrm{sp}(a)' = 0$. Otherwise, $\mathrm{sp}(a)'$ corresponds to the unique row in $\mathrm{RRef}(M_{P\cup G'})$ that introduces a new leading monomial.

Surprisingly, we do not have to restrict ourselves to initialising the set P with just one critical pair. In fact, we can start with as many critical pairs as we want. Hence, this procedure allows us to compute normal forms of multiple S-polynomials simultaneously. The following algorithm formalises the steps described above and builds the main ingredient of the F4 algorithm.

Algorithm 5 Reduction

Input: a finite set $P \subseteq K\langle X\rangle$ and $G \subseteq K\langle X\rangle$
Output: $\tilde{P} \subseteq K\langle X\rangle \setminus \{0\}$

1: $G' = \mathrm{SymbolicPreprocessing}(P, G)$
2: $P' = P \cup G'$
3: $\tilde{P} = \{p \in P_{\mathrm{RRef}(M_{P'})} \mid p \neq 0 \text{ and } \mathrm{lm}(p) \notin \mathrm{lm}(P')\}$
4: **return** $\tilde{P}$

In the following, we collect some crucial properties of the set $\tilde{P}$ returned by Algorithm 5.

Lemma 4.25. *Let $P \subseteq K\langle X\rangle$ be a finite set and $G \subseteq K\langle X\rangle$. Furthermore, let $\tilde{P} = \mathrm{Reduction}(P, G)$. Then, $\mathrm{lm}(p) \notin \{a\,\mathrm{lm}(g)b \mid a, b \in \langle X\rangle,\ g \in G\}$ for all $p \in \tilde{P}$.*

Proof. Assume that there exists $p \in \tilde{P}$ such that $\mathrm{lm}(p) \in \{a\,\mathrm{lm}(g)b \mid a, b \in \langle X\rangle,\ g \in G\}$. So, there exist $a, b \in \langle X\rangle$ and $g \in G$ such that $\mathrm{lm}(p) = a\,\mathrm{lm}(g)b = \mathrm{lm}(agb)$. Since $p \in \tilde{P}$, we know that $\mathrm{lm}(p) \notin \mathrm{lm}(P')$ and, in particular, $\mathrm{lm}(p) \notin \mathrm{lm}(P)$. Hence, $\mathrm{lm}(p)$ has been processed at some point during the execution of $\mathrm{SymbolicPreprocessing}(P, G)$ and at this point agb must have been added to $G' \subseteq P'$. But this contradicts the fact that $\mathrm{lm}(p) \notin \mathrm{lm}(P')$. □

Lemma 4.26. *Let $P \subseteq K\langle X\rangle$ be a finite set and $G \subseteq K\langle X\rangle$. Furthermore, let $\tilde{P} = \mathrm{Reduction}(P, G)$ and let P' be as obtained in Algorithm 5 during the computation of $\tilde{P}$. Then, all elements in $\mathrm{span}_K(P')$ can be reduced to zero by $P' \cup \tilde{P}$.*

Proof. Assume that there exists an element in $\mathrm{span}_K(P')$ that cannot be reduced to zero by $P' \cup \tilde{P}$. Let $p \in \mathrm{span}_K(P')$ be such an element and such that $\mathrm{lm}(p)$ is minimal among all such elements. Then, clearly $p \neq 0$. Since the rows of $M_{P'}$ and $\mathrm{RRef}(M_{P'})$ generate the same vector space over K, we also have $\mathrm{span}_K(P') = \mathrm{span}_K(P'')$, where $P'' = P_{\mathrm{RRef}(M_{P'})}$. Hence, we can write p as a K-linear combination of elements $p_1, \ldots, p_n \in P'' \setminus \{0\}$. We note that the leading monomials of the p_i must all be different, since each p_i corresponds to a row in $\mathrm{RRef}(M_{P'})$ and $\mathrm{RRef}(M_{P'})$ cannot have several rows with pivots in the same column. This implies that we must have $\mathrm{lm}(p) = \mathrm{lm}(p_{i_0})$ for some $1 \leq i_0 \leq n$. By the way we constructed $\tilde{P}$ we then either have $p_{i_0} \in \tilde{P}$ or $\mathrm{lm}(p_{i_0}) \in \mathrm{lm}(P')$. In any case, there exists $g \in P' \cup \tilde{P}$ such that $\mathrm{lm}(g) = \mathrm{lm}(p_{i_0}) = \mathrm{lm}(p)$ and since $\tilde{P} \subseteq P'' \subseteq \mathrm{span}_K(P'') = \mathrm{span}_K(P')$ we know that $g \in \mathrm{span}_K(P')$. Using this g to reduce p, we obtain $p' \in \mathrm{span}_K(P')$ such that $\mathrm{lm}(p') \prec \mathrm{lm}(p)$

and since p cannot be reduced to zero, neither can p', which is a contradiction to the minimality of p. □

Using the previous observations, we can prove that Algorithm 5 can be compared to the CheckResolvability routine in the Buchberger algorithm. The following result can be compared to to Proposition 5.4.9 in [Xiu12], respectively to Lemma 2.2 in [Fau99].

Theorem 4.27. *Let $G \subseteq K\langle X\rangle$ and let $C \subseteq \operatorname{crit}_G$ be finite. Furthermore, let $P = \bigcup_{(f_a,g_a)\in C}\{f_a, g_a\}$ be the set of all polynomials appearing in the critical pairs of C and let $\tilde{P}$ = Reduction(P, G). Then, the ambiguities corresponding to the critical pairs in C are all resolvable with respect to $G \cup \tilde{P}$, i.e.*

$$\operatorname{sp}(a) \xrightarrow{*}_{G\cup\tilde{P}} 0,$$

for all $a \in amb_G$ such that $\operatorname{cp}(a) \in C$.

Proof. Let $a \in \operatorname{amb}_G$ be an ambiguity such that $\operatorname{cp}(a) = (f_a, g_a) \in C$. Furthermore, let P' be as obtained in Algorithm 5 during the computation of $\tilde{P}$. Since $f_a, g_a \in P'$ and $\operatorname{sp}(a) = g_a - f_a$, we have that $\operatorname{sp}(a) \in \operatorname{span}_K(P')$. Hence, according to Lemma 4.26 we can reduce $\operatorname{sp}(a)$ to zero by $P' \cup \tilde{P}$. Note that every element of a critical pair of G can be written as $cwgw'$ for some $c \in K$, $w, w' \in \langle X\rangle$ and $g \in G$. In particular, all elements of P are of this form and this clearly also holds for the elements in G' = SymbolicPreprocessing(P, G), which are added to P to obtain P'. Consequently, every reduction done by some $p \in P'$ can also be done by some $g \in G$, which means that $\operatorname{sp}(a)$ can be reduced to zero by $G \cup \tilde{P}$. □

As a consequence of Theorem 4.27, we get an analogous statement to Proposition 4.12, which allows us to decide when a given finite set $G \subseteq K\langle X\rangle$ is a Gröbner basis of (G).

Corollary 4.28. *Let $G \subseteq K\langle X\rangle$ be finite and let $P = \bigcup_{(f_a,g_a)\in\operatorname{crit}_G}\{f_a, g_a\}$. Furthermore, let $\tilde{P}$ = Reduction(P, G). Then, $\tilde{P} = \emptyset$ if and only if G is a Gröbner basis of (G).*

Proof. We note that since G is finite, also P is finite. Hence, $\tilde{P}$ is well defined. If $\tilde{P} = \emptyset$, then Theorem 4.27 yields that all ambiguities of G are resolvable with respect to $G \cup \tilde{P} = G$. Consequently, G is a Gröbner basis of (G) according to Theorem 4.10. Now, assume that $\tilde{P} \neq \emptyset$ and let $p \in \tilde{P}$. Furthermore, let P' be as obtained in Algorithm 5 during the computation of $\tilde{P}$. Then, $p \in \operatorname{span}_K(P') \subseteq (G)$. By Lemma 4.25, we have $\operatorname{lm}(p) \notin \{a \operatorname{lm}(g)b \mid a, b \in \langle X\rangle,\ g \in G\}$. This implies that $\operatorname{lm}(p)$ cannot be reduced by G and since the leading term of a polynomial cannot cancel after a reduction, this shows that $p \in (G)$ cannot be reduced to zero by G, which violates condition 2 of Theorem 3.38. Hence, G cannot be a Gröbner basis of (G). □

Based upon Theorem 4.27 and Corollary 4.28, we can formulate the F4 algorithm to enumerate Gröbner bases in $K\langle X\rangle$.

Algorithm 6 F4

Input: a finite set $F \subseteq K\langle X\rangle$

Output if the algorithm terminates: $G \subseteq K\langle X\rangle$ such that G is a Gröbner basis of (F)

1: $G = F$
2: $s = 0$
3: **while** $s \neq |G|$ **do**
4: $\quad s = |G|$
5: $\quad critPairs = crit_G$
6: $\quad$ **while** $critPairs \neq \emptyset$ **do**
7: $\quad\quad$ select $C \subseteq critPairs$
8: $\quad\quad critPairs = critPairs \setminus C$
9: $\quad\quad P = \bigcup_{(f_a,g_a)\in C}\{f_a, g_a\}$
10: $\quad\quad \tilde{P} = \text{Reduction}(P, G)$
11: $\quad\quad G = G \cup \tilde{P}$
12: **return** G

Theorem 4.29. *Let $F \subseteq K\langle X\rangle$ be a finite set, denote $G_0 = F$ and for $n \in \mathbb{N}$, let G_n be the result of Algorithm 6 after n iterations of the outer **while** loop given F as input. Then, $G = \bigcup_{i\geq 0} G_n$ is a Gröbner basis of (F). In this sense, Algorithm 6 enumerates a Gröbner basis G of (F). Furthermore, if (F) admits a finite Gröbner basis, then Algorithm 6 terminates and returns a Gröbner basis of (F).*

Proof. To prove that $G = \bigcup_{i\geq 0} G_n$ is a Gröbner basis of (F), we first note that all elements in $\tilde{P}_n$, which are added to G during the n-th iteration of Algorithm 3, are elements of (F). Since also $F \subseteq G$, this implies $(G) = (F)$. Hence, according to Theorem 4.10 it remains to show that all ambiguities of G are resolvable. To this end, let $a = (ABC, A, C, g_i, g_j) \in \text{amb}_G$ be an ambiguity of two elements $g_i, g_j \in G$. Since $G_n \subseteq G_{n+1}$ for all $n \geq 0$, there exists $N \geq 0$ such that $g_i, g_j \in G_N$. If $\text{sp}(a)$ can be reduced to zero by G_N then we are done. Otherwise, $\text{sp}(a)$ can be reduced to zero by $G_{N+1} = G_N \cup \tilde{P}_N$ according to Theorem 4.27. Hence, in any way $\text{sp}(a)$ can be reduced to zero by $G_{N+1} \subseteq G$ and therefore a is resolvable. The proof of termination if (F) admits a finite Gröbner basis proceeds exactly as the proof of Proposition 4.14. □

To ensure the termination of Algorithm 6, we can add the same constraints as for the Buchberger algorithm. We can also apply different strategies on how to select the subset $C \subseteq critPairs$ in each step, which, as for Buchberger's algorithm, can have a huge impact on the performance of the algorithm in practice. Usually, the so-called *normal* selection strategy, where always all critical pairs with minimal degree are selected, performs quite well. Furthermore, as in the case of the Buchberger algorithm, we have phrased the F4 algorithm in such a way that every selection strategy is a fair selection strategy. Also, note that if we always pick only one critical pair in step 7 of Algorithm 6, we obtain the classical Buchberger algorithm as presented in the previous section.

To end this section, we compute a Gröbner basis of the ideal considered in Example 4.15, but now using the F4 algorithm instead of the Buchberger algorithm.

Example 4.30. As in Example 4.15, we work over $\mathbb{Q}\langle x, y\rangle$ equipped with $\preceq_{deglex}$ where we order the indeterminates as $x \prec_{lex} y$ and consider the ideal generated by $F = \{f_1, f_2, f_3, f_4\} \subseteq \mathbb{Q}\langle x, y\rangle$, with

$$f_1 = yyx + x, \qquad f_2 = y + 1, \qquad f_3 = xy - x, \qquad f_4 = xyx + xx.$$

In this example, we use the F4 algorithm to compute a Gröbner basis of (F).

We start by setting $G = F$. According to Algorithm 6, we first have to compute all critical pairs of G. To this end, we first compute all ambiguities of G. Recall from Example 4.15, that amb_G consists of the overlap ambiguities

$$\begin{aligned} &a_{13} = (yyxy, yy, y, f_1, f_3), \quad a_{14} = (yyxyx, yy, yx, f_1, f_4), \quad a_{31} = (xyyx, x, yx, f_3, f_1),\\ &a_{43} = (xyxy, xy, y, f_4, f_3), \quad a_{44} = (xyxyx, xy, yx, f_4, f_4) \end{aligned}$$

and the inclusion ambiguities

$$\begin{aligned} &a_{12} = (yyx, 1, yx, f_1, f_2), \quad a'_{12} = (yyx, y, x, f_1, f_2), \quad a_{32} = (xy, x, 1, f_3, f_2),\\ &a_{42} = (xyx, x, x, f_4, f_2), \quad a'_{43} = (xyx, 1, x, f_4, f_3). \end{aligned}$$

The corresponding critical pairs are given by

$$\begin{aligned} &\mathrm{cp}(a_{13}) = (yyxy + xy, yyxy - yyx), && \mathrm{cp}(a_{14}) = (yyxyx + xyx, yyxyx + yyxx),\\ &\mathrm{cp}(a_{31}) = (xyyx - xyx, xyyx + xx), && \mathrm{cp}(a_{43}) = (xyxy + xxy, xyxy - xyx),\\ &\mathrm{cp}(a_{44}) = (xyxyx + xxyx, xyxyx + xyxx), && \mathrm{cp}(a_{12}) = (yyx + x, yyx + yx),\\ &\mathrm{cp}(a'_{12}) = (yyx + x, yyx + yx), && \mathrm{cp}(a_{32}) = (xy - x, xy + x),\\ &\mathrm{cp}(a_{42}) = (xyx + xx, xyx + xx), && \mathrm{cp}(a'_{43}) = (xyx + xx, xyx - xx). \end{aligned}$$

In this example, we apply the normal selection strategy. Hence, we first have to select all critical pairs of minimal degree 2, which is just one pair. So, the set P is given by

$$P = \{xy - x, xy + x\}.$$

Note that the symbolic preprocessing of P with respect to G yields the empty set, i.e.

$$G' = \emptyset.$$

Now, we can form the matrix $M_{P'}$ with $P' = P \cup G' = P$, which is given by

$$M_{P'} = \begin{array}{c} \begin{array}{cc} xy & x \end{array} \\ \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \end{array} \begin{array}{l} xy - x \\ xy + x \end{array},$$

and reduce it to reduced row echelon form

$$\mathrm{RRef}(M_{P'}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The polynomial form of this matrix is $P_{\mathrm{RRef}(M_{P'})} = \{xy, x\}$, and since $\mathrm{lm}(xy) = xy \in \mathrm{lm}(P')$ but $\mathrm{lm}(x) = x \notin \mathrm{lm}(P)$, we only have to add $f_5 = x$ to G. This means that we now have

$$G = \{f_1, f_2, f_3, f_4, f_5\}.$$

Following the normal selection strategy, we next process all critical pairs of degree 3. Hence, P is given by

$$P = \{yyx + x, yyx + yx, xyx + xx, xyx - xx\}.$$

During the symbolic preprocessing of P with respect to G, the following set of monomials has to be considered

$$T = \{x, yx, xx\}.$$

The corresponding set of reductors is given by

$$G' = \{f_5, f_2x, f_5x\}.$$

As in the previous step, we form the matrix $M_{P'}$ with $P' = P \cup G'$ and compute its reduced row echelon form. This yields

$$M_{P'} = \begin{array}{c} \begin{matrix} yyx & xyx & yx & xx & x \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{c} yyx + x \\ yyx + yx \\ xyx + xx \\ xyx - xx \\ f_5 \\ f_2x \\ f_5x \end{array} \longrightarrow \mathrm{RRef}(M_{P'}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since no row of $\mathrm{RRef}(M_{P'})$ introduces a new leading monomial, we cannot add any new element to G in this case. So, we continue by processing the remaining critical pairs. However, now we slightly deviate from the normal selection strategy and instead of only selecting all critical pairs of degree 4, we process all remaining critical pairs. Therefore, we obtain the following set P

$$\begin{aligned} P = \{&yyxy + xy, yyxy - yyx, yyxyx + xyx, yyxyx + yyxx, xyyx - xyx, \\ &xyyx + xx, xyxy + xxy, xyxy - xyx, xyxyx + xxyx, xyxyx + xyxx\}. \end{aligned}$$

Note that we have already done the symbolic preprocessing of P with respect to G in Example 4.22, where we obtained

$$G' = \{f_3, f_1, f_4, f_1x, f_5x, xf_3, f_5xyx, xyxf_5, f_5\}.$$

Then, $M_{P'}$ with $P' = P \cup G'$ is given by

$$M_{P'} = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}.$$

Its reduced row echelon form

$$\mathrm{RRef}(M_{P'}) = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

does not introduce any new leading monomial. So, again, we cannot add any new element to G.

As there are now no critical pairs left to process, we exit the inner **while** loop of Algorithm 6, but because the size of G has changed compared to the beginning this iteration, we cannot exit the whole algorithm. Instead, we have to recompute all critical pairs of G. In particular, we only have to recompute critical pairs that have not already been processed in a previous step. Hence, we only have to consider critical pairs which contain at least one element of G that has been added during the last iteration. In this case, we obtain the following four inclusion ambiguities

$$\begin{aligned} &a_{15} = (yyx, yy, 1, f_1, f_5), \qquad a_{35} = (xy, 1, y, f_3, f_5), \qquad a_{45} = (xyx, 1, yx, f_4, f_5), \\ &a'_{45} = (xyx, xy, 1, f_4, f_5), \end{aligned}$$

and consequently, the following critical pairs

$$\begin{aligned} &\mathrm{cp}(a_{15}) = (yyx + x, yyx), && \mathrm{cp}(a_{35}) = (xy - x, xy), \\ &\mathrm{cp}(a_{45}) = (xyx + xx, xyx), && \mathrm{cp}(a'_{45}) = (xyx + xx, xyx). \end{aligned}$$

Again, we slightly deviate from the normal selection strategy and select all critical pairs at once. This gives

$$P = \{yyx + x, yyx, xy - x, xy, xyx + xx, xyx, xyx + xx, xyx).$$

It is easy to see that the symbolic preprocessing of P with respect to G yields

$$G' = \{f_5, xf_5\}.$$

Therefore, we get the following matrix $M_{P'}$ with $P' = P \cup G'$ and its reduced row echelon form $\mathrm{RRef}(M_{P'})$.

$$M_{P'} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \longrightarrow \mathrm{RRef}(M_{P'}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

So, once more $\mathrm{RRef}(M_{P'})$ does not introduce any new leading monomial, and hence, we cannot add anything to G. Since now there are no critical pairs left to process and the size of G has not changed during this iteration, we can terminate our computation knowing that

$$G = \{f_1, f_2, f_3, f_4, f_5\}$$

is a Gröbner basis of (f_1, f_2, f_3, f_4).

4.4 Cofactor representations

So far, we have seen that we can verify the ideal membership of a given polynomial $f \in K\langle X\rangle$ in an ideal (F) generated by a finite set $F \subseteq K\langle X\rangle$ by reducing f to zero. Typically, this is done using a (partial) Gröbner basis G of (F). While such a reduction to zero is indeed a valid proof that $f \in (F)$, it is in practice often irreproducible. When we claim that $f \in (F)$ because we were able to reduce f to zero by G, then the reader can either trust us that we did not make a mistake during the computation of G and the reduction of f, or they can redo all computations, which can be quite tedious. This is where the notion of cofactor representations comes into play. If we can provide a cofactor representation of f with respect to F, then it is easily verifiable that indeed $f \in (F)$ holds, simply by multiplying out the corresponding linear combination. Fortunately, we can use the tools and techniques developed in the previous sections to obtain such a cofactor representation. In fact, we basically only have to trace the cofactors during the computation of the (partial) Gröbner basis G and during the reduction of f to zero. In the following, we explain this in more detail.

Recall that if we can reduce f to $f' \in K\langle X\rangle$ by some element $g \in G$ using the cofactors $a, b \in \langle X\rangle$, then the three polynomials are related as follows

$$f' = f - cagb,$$

where $c = \frac{\operatorname{coeff}(f, \operatorname{lm}(agb))}{\operatorname{lc}(g)} \in K$ is a particular constant that is not of special interest for us now. Hence, a reduction of f to zero by G, i.e.

$$f \ \to_{a_1,g_1,b_1} \ \dots \ \to_{a_n,g_n,b_n} \ 0,$$

with $a_i, b_i \in \langle X\rangle$ and $g_i \in G$ for $1 \leq i \leq n$, yields the equation

$$0 = f - \sum_{i=1}^{n} c_i a_i g_i b_i, \tag{1}$$

where the $c_i \in K$ are again certain constants. So, to obtain a cofactor representation of f with respect to G, we basically only have to keep track of the cofactors and reductors used during the reduction of f. However, since usually not all g_i appearing in this cofactor representation are elements of F, it remains to rewrite the linear combination in terms of the generators in F. To this end, if suffices to find cofactor representations of the g_i with respect to F because if we know that g_i can be written as

$$g_i = \sum_{j=1}^{n_i} c_{i,j} a_{i,j} f_{i,j} b_{i,j},$$

with $c_{i,j} \in K$, $a_{i,j}, b_{i,j} \in \langle X\rangle$ and $f_{i,j} \in F$ for $1 \leq j \leq n_i$ and $1 \leq i \leq n$, then we can plug these linear combinations into (1) to obtain

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n_i} c_i c_{i,j} a_i a_{i,j} f_{i,j} b_{i,j} b_i,$$

which is a cofactor representation of f with respect to F. After collecting equal cofactors of several summands (if possible), we can write the equation above as

$$f = \sum_{i=1}^{m} \tilde{a}_i f_i \tilde{b}_i,$$

with $\tilde{a}_i, \tilde{b}_i \in K\langle X\rangle$ and $f_i \in F$. Typically, we represent such a cofactor representation as a list of triples $\{(\tilde{a}_1, f_1, \tilde{b}_1), \ldots, (\tilde{a}_m, f_m, \tilde{b}_m)\}$. So, it only remains to find cofactor representations of the elements in G with respect to F. This can be done by tracing the computation of the (partial) Gröbner basis G and slightly differs depending on the algorithm used to compute G. We first consider the Buchberger algorithm.

4.4.1 Cofactors in the Buchberger algorithm

Starting with a finite set $F \subseteq K\langle X\rangle$ as input of the Buchberger algorithm, we have to keep track of every computation done in the ideal (F) during the execution of the algorithm, beginning with the computation of the S-polynomials in the CheckResolvability subroutine. So, this is the first algorithm that we have to extend in order to allow us to obtain cofactor representations. As already stated before, we represent a cofactor representation as a list of triples.

Algorithm 7 ExtendedCheckResolvability

Input: a finite set $G \subseteq K\langle X\rangle$ such that all elements of G are monic

Output: A set *spol* such that $spol = \emptyset$ if and only if G is a Gröbner basis of (G). If $spol \neq \emptyset$, then $spol = \{(s'_i, linearComb_i) \mid 1 \leq i \leq n\}$ such that $linearComb_i$ is a cofactor representation of s'_i w.r.t. G.

1: $spol = \emptyset$
2: **foreach** $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ **do**
3: $\quad linearComb = \emptyset$
4: $\quad$ **if** a is an Overlap ambiguity **then**
5: $\quad\quad linearComb = \{(1, f, C), (-A, g, 1)\}$
6: $\quad$ **else**
7: $\quad\quad linearComb = \{(1, f, 1), (-A, g, C)\}$
8: $\quad s_1 = \mathrm{sp}(a)$
9: $\quad$ compute a normal form s' of s_1 w.r.t. to $\to_G$, i.e.

$$s_1 \to_{a_1,g_1,b_1} s_2 \to_{a_2,g_2,b_2} \cdots \to_{a_m,g_m,b_m} s'$$

10: $\quad$ **if** $s' \neq 0$ **then**
11: $\quad\quad linearComb = linearComb \cup \{(-c_i a_i, g_i, b_i) \mid c_i = \mathrm{coeff}(s_i, \mathrm{lm}(a_i g_i b_i)),\ 1 \leq i \leq m\}$
12: $\quad\quad spol = spol \cup \{(s', linearComb)\}$
13: **return** *spol*

Basically, Algorithm 7 serves the same purpose as Algorithm 2, namely to test whether G is a Gröbner basis of (G) by checking the resolvability of all ambiguities of G and returning the

normal forms of those S-polynomials that could not be reduced to zero. However, Algorithm 7 does not only return the reduced S-polynomials but also a cofactor representation with respect to G for each element it returns. Also, note that we require the polynomials in G to be monic so that we do not have to worry about leading coefficients, which would only make the involved formulae more complicated.

Based upon Algorithm 7, we can extend the Buchberger algorithm.

Algorithm 8 ExtendedBuchberger

Input: a finite set $F \subseteq K\langle X\rangle$ such that all elements of F are monic

Output if the algorithm terminates: $G \subseteq K\langle X\rangle$ such that G is a Gröbner basis of (F) and a set $cofactors$ such that for every $g_i \in G \setminus F$ there is a pair $(g_i, linearComb_i) \in cofactors$ and $linearComb_i$ is a cofactor representation of g_i w.r.t. F.

1: $G = F$
2: $cofactors = \emptyset$
3: $spol = \text{ExtendedCheckResolvability}(G)$
4: **while** $spol \neq \emptyset$ **do**
5: **while** $spol \neq \emptyset$ **do**
6: select $(f, linearComb) \in spol$
7: $spol = spol \setminus \{(f, linearComb)\}$
8: $f_1 = f$
9: compute a normal form f' of f_1 w.r.t. to $\to_G$, i.e.

$$f_1 \to_{a_1,g_1,b_1} f_2 \to_{a_2,g_2,b_2} \cdots \to_{a_m,g_m,b_m} f'$$

10: **if** $f' \neq 0$ **then**
11: $linearComb = linearComb \cup \{(-c_i a_i, g_i, b_i) \mid c_i = \text{coeff}(f_i, \text{lm}(a_i g_i b_i)),\ 1 \leq i \leq m\}$
12: **if** $\text{lc}(f') \neq 1$ **then**
13: $f' = \frac{1}{\text{lc}(f')} f'$
14: $linearComb = \{(\frac{1}{\text{lc}(f')} a, g, b) \mid (a, g, b) \in linearComb\}$
15: $G = G \cup \{f'\}$
16: $cofactors = cofactors \cup \{(f', linearComb)\}$
17: $spol = \text{ExtendedCheckResolvability}(G)$
18: $cofactors = \text{Rewrite}(cofactors)$
19: **return** $(G, cofactors)$

Before explaining the subroutine Rewrite($cofactors$), we make some remarks on Algorithm 8. First of all, note that we again restrict ourselves to monic polynomials in order to avoid issues with different leading coefficients. This is also why we make every polynomial f' monic before adding it to G. But apart from this difference, the set G returned by Algorithm 8 is the same set that the usual Buchberger algorithm would return. The new feature of this algorithm is that a cofactor representation of every element $g_i \in G \setminus F$ that gets added to G is stored in the set

$cofactors$ in form of a pair $(g_i, linearComb_i)$, where $linearComb_i$ is a set of triples forming a cofactor representation of g_i with respect to F, i.e. $linearComb_i = \{(a_{i,j}, f_{i,j}, b_{i,j}) \mid f_{i,j} \in F,\ 1 \leq j \leq n_i\}$ and $g_i = \sum_{j=1}^{n_i} a_{i,j} f_{i,j} b_{i,j}$.

Before calling Rewrite($cofactors$) all cofactor representations saved in $cofactors$ are with respect to G. More precisely, we know that the cofactor representation of $g_i \in G \setminus F$ can only contain elements that have been added to G before g_i. To make this more precise, we write G as $G = \{f_1, \ldots, f_m, g_1, \ldots, g_n\}$, where the $f_i \in F$ are in an arbitrary but fixed order and the $g_i \in G \setminus F$ are labelled chronologically according to the moment they have been added to G. Then, the cofactor representation of g_i is with respect to $F \cup \{g_1, \ldots, g_{i-1}\}$. So, we can incrementally rewrite the cofactor representation of g_i using the already rewritten cofactor representations of $\{g_1, \ldots, g_{i-1}\}$. This basically already describes the functionality of Rewrite($cofactors$).

Algorithm 9 Rewrite

Input: a set of pairs $cofactors = \{(g_i, linearComb_i) \mid 1 \leq i \leq n\}$ as obtained in Algorithm 8, where the g_i are labelled chronologically according to the moment they have been added to $cofactors$

Output: a set of pairs $newCofactors = \{(g_i, newLinearComb_i) \mid 1 \leq i \leq n\}$ containing rewritten cofactor representations

1: $newCofactors = \emptyset$
2: **for** $i = 1, \ldots, n$ **do**
3: $\quad newLinearComb = \emptyset$
4: $\quad$ **foreach** $(a_{i,j}, g_{i,j}, b_{i,j}) \in linearComb_i$ **do**
5: $\quad\quad$ **if** there exists $1 \leq k < i$ such that $g_{i,j} = g_k$ **then**
6: $\quad\quad\quad newLinearComb = newLinearComb \cup \{(a_{i,j}a, g, bb_{i,j}) \mid (a, g, b) \in linearComb_k\}$
7: $\quad\quad$ **else**
8: $\quad\quad\quad newLinearComb = newLinearComb \cup \{(a_{i,j}, g_{i,j}, b_{i,j})\}$
9: $\quad newCofactors = newCofactors \cup \{(g_i, newLinearComb)\}$
10: **return** $newCofactors$

4.4.2 Cofactors in the F4 algorithm

As in the case of Buchberger's algorithm, also during the execution of the F4 algorithm, we have to keep track of every computation done in the ideal (F) when given a finite set $F \subseteq K\langle X\rangle$ as input. In this case, this starts when the critical pairs are formed. Hence, we introduce the following procedure that allows us to trace these computations.

Algorithm 10 ExtendedCritPairs

Input: a finite set $G \subseteq K\langle X\rangle$ such that all elements in G are monic
Output: a set $critPairs$

1: $critPairs = \emptyset$
2: **foreach** $a = (ABC, A, C, f, g) \in \mathrm{amb}_G$ **do**
3: $\quad (f_a, g_a) = \mathrm{cp}(a)$
4: $\quad$ **if** a is an Overlap ambiguity **then**
5: $\quad\quad critPairs = critPairs \cup \{(f_a, g_a, (1, f, C), (A, g, 1))\}$
6: $\quad$ **else**
7: $\quad\quad critPairs = critPairs \cup \{(f_a, g_a, (1, f, 1), (A, g, C))\}$
8: **return** $critPairs$

Remark. For reason as in the previous section, we restrict ourselves to monic polynomials.

So, basically we only extend each critical pair by cofactor representations of the two critical polynomials. The next algorithm that we have to adapt is the SymbolicPreprocessing subroutine, where we now return the triple (a, g, b) instead of the reductor $g \in G$ already multiplied by the corresponding cofactors $a, b \in \langle X\rangle$. Everything else remains as in Algorithm 4.

Algorithm 11 ExtendedSymbolicPreprocessing

Input: a finite set $P \subseteq K\langle X\rangle$ and $G \subseteq K\langle X\rangle$
Output: $G' \subseteq \{(a, g, b) \mid a, b \in \langle X\rangle,\ g \in G\}$

1: $G' = \emptyset$
2: $T = \mathrm{supp}(\mathrm{tail}(P))$
3: $done = \mathrm{lm}(P)$
4: **while** $T \neq \emptyset$ **do**
5: $\quad$ select $t \in T$
6: $\quad T = T \setminus \{t\}$
7: $\quad done = done \cup \{t\}$
8: $\quad$ **if** there exist $g \in G$, $a, b \in \langle X\rangle$ such that $h_{a,g,b}$ acts nontrivially on t **then**
9: $\quad\quad G' = G' \cup \{(a, g, b)\}$
10: $\quad\quad T = T \cup (\mathrm{supp}(h_{a,g,b}(t)) \setminus done)$
11: **return** G'

Of course, we then also have to adapt the Reduction algorithm accordingly. To this end, we denote the i-th row of a matrix M by M_i and the j-th entry in the i-th row by $M_{i,j}$.

Algorithm 12 ExtendedReduction

Input: a set of pairs $\{(p_i, triple_i) \mid 1 \leq i \leq n\}$ with $p_i \in K\langle X\rangle$ and $triple_i = (a_i, g_i, b_i) \in \langle X\rangle \times G \times \langle X\rangle$ such that $a_i g_i b_i = p_i$ and $G \subseteq K\langle X\rangle$

Output: a set $\tilde{P}$ such that either $\tilde{P} = \emptyset$ or $\tilde{P} = \{(\tilde{p}_i, linearComb_i) \mid 1 \leq i \leq k\}$, where $linearComb_i$ is a cofactor representation of $\tilde{p}_i$ w.r.t. G

1: $P = \{p_1, \ldots, p_n\}$
2: $G' = \text{ExtendedSymbolicPreprocessing}(P, G)$
3: $P' = F \cup \{agb \mid (a, g, b) \in G'\}$
4: $M = \text{RRef}(M_{P'})$
5: also compute the transformation matrix T such that $TM_{P'} = M$
6: set $triples = \{t_1, \ldots, t_m\}$ such that

$$\{t_1, \ldots, t_m\} = \{triple_1, \ldots, triple_n\} \cup G'$$

and t_i corresponds to the i-th row in $M_{P'}$
7: $\tilde{P} = \emptyset$
8: **for** $i = 1, \ldots, m$ **do**
9: $\quad p = \varphi(M_i)$
10: $\quad$ **if** $p \neq 0$ and $\text{lm}(p) \notin \text{lm}(P)$ **then**
11: $\quad\quad linearComb = \{(T_{i,j}a_j, g_j, b_j) \mid t_j = (a_j, g_j, b_j),\ 1 \leq j \leq m\}$
12: $\quad\quad \tilde{P} = \tilde{P} \cup \{(p, linearComb)\}$
13: **return** $\tilde{F}$

Similar to Algorithm 5, the main task of this algorithm is to reduce the polynomials contained in the pairs of the input by G. Simultaneously, we now also compute a cofactor representation of the returned elements with respect to G. The polynomials in the input are expected to come from the ExtendedCriticalPairs(G) subroutine, and hence, are elements of G multiplied from the left and right by some monomials. This information is stored in the $triple_i$ part of the pairs. As a first step in Algorithm 12, we extract the polynomials from the pairs. So, the set P corresponds to the input of the usual Reduction algorithm. In step 2, we call the ExtendedSymbolicPreprocessing subroutine, which returns the reductors and cofactors in the form of triples $(a, g, b) \in \langle X\rangle \times G \times \langle X\rangle$. Then, we form the set P' that contains all polynomials to be reduced and the reductors, but now multiplied by the corresponding cofactors.

After that, we compute $M = \text{RRef}(M_{P'})$ as in Algorithm 5. However, now we also have to compute the transformation matrix T such that $TM_{P'} = M$, since this matrix encodes the cofactor representations we are looking for. It is easy to see that

$$M_i = \sum_{j=1}^{|P'|} T_{i,j} \cdot (M_{P'})_j,$$

where the multiplication $T_{i,j} \cdot (M_{P'})_j$ of the scalar $T_{i,j}$ with the vector $(M_{P'})_j$ is done component-

wise. In terms of polynomials, this equation reads

$$\varphi(M_i) = \sum_{j=1}^{|P'|} T_{i,j} \cdot \varphi((M_{P'})_j), \tag{2}$$

where φ denotes the isomorphism that maps a row of a matrix to a polynomial in $\mathrm{span}_K(\mathrm{supp}(P'))$. Since all polynomials $\varphi((M_{P'})_j)$ are of the form $a_j g_j b_j$ with $a_j, b_j \in \langle X \rangle$ and $g_j \in G$ this is a cofactor representation of $\varphi(M_i)$ with respect to G. However, we want to know the cofactors a_j, b_j and the elements g_j explicitly. To this end, we form the set $triples = \{t_1, \ldots, t_m\}$ in step 6 of Algorithm 12. This set consists of the triples of the input and the triples returned by ExtendedSymbolicPreprocessing(P, G). We note that we have exactly one triple $t_j = (a_j, g_j, b_j)$ corresponding to each element $p' \in P'$, and hence, also to each row in $M_{P'}$, such that $p' = a_j g_j b_j$. When we label the triples in $triples$ such that t_j corresponds to the polynomial represented by the j-th row of $M_{P'}$, we can go through all rows of M and translate equation (2) into an explicit cofactor representation of $\varphi(M_i)$ basically by replacing $\varphi((M_{P'})_j)$ by t_j. More precisely, this is done by forming the set $\{(T_{i,j} a_j, g_j, b_j) \mid t_j = (a_j, g_j, b_j),\ 1 \leq j \leq m\}$. Of course, this only has to be done if $\varphi(M_i)$ actually appears in the output, i.e. if $\varphi(M_i) \neq 0$ and $\mathrm{lm}(\varphi(M_i)) \notin \mathrm{lm}(f)$. Finally, we return the set $\tilde{P}$ consisting of pairs $(p, linearComb)$ where p is a reduced polynomial and $linearComb$ is a cofactor representation of p with respect to G.

We can now put all pieces together and state an extended version of the F4 algorithm.

Algorithm 13 ExtendedF4

Input: a finite set $F \subseteq K\langle X \rangle$ such that all elements in F are monic

Output if the algorithm terminates: $G \subseteq K\langle X \rangle$ such that G is a Gröbner basis of (F) and a set $cofactors$ such that for every $g_i \in G \setminus F$ there is a pair $(g_i, linearComb_i) \in cofactors$ with $linearComb_i$ being a cofactor representation of g_i w.r.t. F.

1: $G = F$
2: $s = 0$
3: **while** $s \neq |G|$ **do**
4: $\quad s = |G|$
5: $\quad critPairs = \mathrm{ExtendedCritPairs}(G)$
6: $\quad$ **while** $critPairs \neq \emptyset$ **do**
7: $\quad\quad$ select $C \subseteq critPairs$
8: $\quad\quad critPairs = critPairs \setminus C$
9: $\quad\quad P = \{(f_a, t_f) \mid (f_a, g_a, t_f, t_g) \in C\} \cup \{(g_a, t_g) \mid (f_a, g_a, t_f, t_g) \in C\}$
10: $\quad\quad \tilde{P} = \mathrm{ExtendedReduction}(P, G)$
11: $\quad\quad G = G \cup \{p \mid (p, linearComb) \in \tilde{P}\}$
12: $\quad\quad cofactors = cofactors \cup \tilde{P}$
13: $cofactors = \mathrm{Rewrite}(cofactors)$
14: **return** $(G, cofactors)$

As a result of this algorithm, we obtain exactly the same set G that the usual F4 algorithm would return. Additionally, we also get a set of pairs $cofactors$ that is of the same structure

as the equally named set returned by the extended Buchberger algorithm. As in the previous section, if we label the elements in $G \setminus F$ chronologically according to the moment they have been added to G, then the cofactor representation of $g_i \in G \setminus F$ saved in $cofactors$ is initially with respect to $F \cup \{g_1, \ldots, g_{i-1}\}$. This allows us to reuse the Rewrite subroutine presented in the previous section to obtain cofactor representations with respect to F. Also, note that here we do not have to worry about making the polynomials monic that we add to G, since each of these polynomials corresponds to a row of a matrix in reduced row echelon form, and therefore, has to be monic.

4.5 Optimisations

It is well known that already in the commutative case the time complexity of the Buchberger algorithm is doubly exponential in the number of indeterminates [MM82]. So, in order to make this algorithm also applicable to larger problems, much effort was put into optimising it. The two most prominent approaches to do this are deletion criteria [Buc79, GM88] and selection strategies [GMNRT91]. The latter one is concerned with implementing certain heuristics specifying when to process which S-polynomial in order to keep the computation as short as possible. Selection strategies also ultimately led to the development of the F4 algorithm, which should get rid of some of these heuristics by processing several S-polynomials simultaneously. Here, however, we will focus on the first approach, deletion criteria, which are used to detect and delete unnecessary critical pairs. In the commutative case, there are two classical deletion criteria: the *product criterion* and the *chain criterion*, both of which are due to Buchberger [Buc79]. In Section 4.5.1, we consider their noncommutative analogs for excluding redundant ambiguities from consideration. Instead of deleting redundant ambiguities, we can also try to directly remove unnecessary elements from a (partial) Gröbner basis. In Section 4.5.2, we discuss how this can be done. While these first two optimisation techniques are applicable to both, the (noncommutative) Buchberger algorithm as well as the (noncommutative) F4 algorithm, the third optimisation that we consider is relevant solely for F4. In Section 4.5.3, we present an algorithm due to Faugère and Lachartre [FL10] that can be used to compute the reduced row echelon form of a matrix and that is especially efficient on matrices appearing during the executing of the F4 algorithm.

4.5.1 Deletion criteria

In the commutative case, the product criterion allows us to delete a critical pair whenever the leading monomials of the two polynomials involved are coprime. In the noncommutative case, this translates to: ambiguities of two polynomials $f, g \in K\langle X\rangle \setminus \{0\}$ whose leading terms do not simultaneously divide monomials of length less than $|\operatorname{lm}(f)| + |\operatorname{lm}(g)|$ need not be considered during a Gröbner basis computation. However, this criterion is superfluous since such polynomials f and g do not form any ambiguities anyway. Hence, in the following, we only concern ourselves with generalising the chain criterion, cf. [Mor94, HRR18].

First, we consider the noncommutative chain criterion for overlap ambiguities. To this end, we recall that an ambiguity $a = (ABC, A, C, f, g)$ of a set $G \subseteq K\langle X\rangle$ is called $\preceq$-resolvable if

$\mathrm{sp}(a) \in I_{G,ABC} = \mathrm{span}_K(\{wg'w' \mid g' \in G,\, w, w' \in \langle X\rangle \text{ such that } w\,\mathrm{lm}(g')w' \prec ABC\})$.

Theorem 4.31. *(Chain criterion for overlap ambiguities)*
Let $g_1, g_2 \in G \subseteq K\langle X\rangle$ have an overlap ambiguity (ABC, A, C, g_1, g_2) with $A, B, C \in \langle X\rangle$. Furthermore, let $g_3 \in G$ be such that $\mathrm{lm}(g_3) \mid ABC$ and such that one of the following cases holds.

1. *$\mathrm{lm}(g_3)$ is a subword of $A = L\,\mathrm{lm}(g_3)R$ and the inclusion ambiguity $(L\,\mathrm{lm}(g_3)RB, L, RB, g_1, g_3)$ is $\preceq$-resolvable.*
2. *$\mathrm{lm}(g_3)$ is a subword of $B = L\,\mathrm{lm}(g_3)R$ and the two inclusion ambiguities $(AL\,\mathrm{lm}(g_3)R, AL, R, g_1, g_3)$ and $(L\,\mathrm{lm}(g_3)RC, L, RC, g_2, g_3)$, respectively, are $\preceq$-resolvable.*
3. *$\mathrm{lm}(g_3)$ is a subword of $C = L\,\mathrm{lm}(g_3)R$ and the inclusion ambiguity $(BL\,\mathrm{lm}(g_3)R, BL, R, g_2, g_3)$ is $\preceq$-resolvable.*
4. *$\mathrm{lm}(g_3)$ is a subword of $AB = L\,\mathrm{lm}(g_3)R$ (with nonempty U, V such that $\mathrm{lm}(g_3) = UV$ and $B = VR$) and the inclusion ambiguity $(L\,\mathrm{lm}(g_3)R, L, R, g_1, g_3)$ as well as the overlap ambiguity $(UVRC, U, RC, g_3, g_2)$ are $\preceq$-resolvable.*
5. *$\mathrm{lm}(g_3)$ is a subword of $BC = L\,\mathrm{lm}(g_3)R$ (with nonempty U, V such that $\mathrm{lm}(g_3) = UV$ and $B = LU$) and the overlap ambiguity $(ALUV, AL, V, g_1, g_3)$ as well as the inclusion ambiguity $(L\,\mathrm{lm}(g_3)R, L, R, g_2, g_3)$ are $\preceq$-resolvable.*
6. *There are nonempty L, R such that $\mathrm{lm}(g_3) = LBR$ (with $A = A_1L$ and $C = RC_2$) and the overlap ambiguity $(A_1LBR, A_1, R, g_1, g_3)$, respectively the inclusion ambiguity (if A_1 is empty) $(A_1LBR, A_1, R, g_3, g_1)$, as well as the inclusion/overlap ambiguity (depending on whether C_2 is empty or not) $(LBRC_2, L, C_2, g_3, g_2)$ are $\preceq$-resolvable.*

Then, the overlap ambiguity (ABC, A, C, g_1, g_2) is $\preceq$-resolvable.

We will not prove this theorem but instead state and prove the following theorem, which generalises the chain criterion also to inclusion ambiguities. We note that the proof of Theorem 4.31 works along the same lines as the proof that we will do. For a proof of Theorem 4.31 in the setting of tensor rings, we refer to [HRR18].

Theorem 4.32. *(Chain criterion for inclusion ambiguities)*
Let $g_1, g_2 \in G \subseteq K\langle X\rangle$ have an inclusion ambiguity (ABC, A, C, g_1, g_2) with $A, B, C \in \langle X\rangle$. Furthermore, let $g_3 \in G$ be such that $\mathrm{lm}(g_3) \mid ABC$ and such that one of the following cases holds.

1. *$\mathrm{lm}(g_3)$ is a subword of $A = L\,\mathrm{lm}(g_3)R$ and the inclusion ambiguity $(L\,\mathrm{lm}(g_3)RBC, L, RBC, g_1, g_3)$ is $\preceq$-resolvable.*
2. *$\mathrm{lm}(g_3)$ is a subword of $B = L\,\mathrm{lm}(g_3)R$ and the two inclusion ambiguities $(AL\,\mathrm{lm}(g_3)RC, AL, RC, g_1, g_3)$ and $(L\,\mathrm{lm}(g_3)R, L, R, g_2, g_3)$, respectively, are $\preceq$-resolvable.*
3. *$\mathrm{lm}(g_3)$ is a subword of $C = L\,\mathrm{lm}(g_3)R$ and the inclusion ambiguity $(ABL\,\mathrm{lm}(g_3)R, ABL, R, g_1, g_3)$ is $\preceq$-resolvable.*

4. *There are L, R with nonempty R such that $AB = L\,\mathrm{lm}(g_3)R$ (with nonempty U, V such that $\mathrm{lm}(g_3) = UV$ and $B = VR$) and the inclusion ambiguity $(L\,\mathrm{lm}(g_3)RC, L, RC, g_1, g_3)$ as well as the overlap ambiguity (UVR, U, R, g_3, g_2) are $\preceq$-resolvable.*

5. *There are L, R with nonempty L such that $BC = L\,\mathrm{lm}(g_3)R$ (with nonempty U, V such that $\mathrm{lm}(g_3) = UV$ and $B = LU$) and the inclusion ambiguity $(AL\,\mathrm{lm}(g_3)R, AL, R, g_1, g_3)$ as well as the overlap ambiguity (LUV, L, V, g_2, g_3) are $\preceq$-resolvable.*

6. *There are L, R, not both empty, such that $\mathrm{lm}(g_3) = LBR$ (with $A = A_1L$ and $C = RC_2$) and the two inclusion ambiguities $(A_1LBRC_2, A_1, C_2, g_1, g_3)$ and (LBR, L, R, g_3, g_2), respectively, are $\preceq$-resolvable.*

Then, the inclusion ambiguity (ABC, A, C, g_1, g_2) is $\preceq$-resolvable.

In order to prove Theorem 4.32, we first establish the following result.

Lemma 4.33. *Let $g_1, g_2 \in G \subseteq K\langle X\rangle$ be nonzero and let $W \in \langle X\rangle$ be such that $\mathrm{lm}(g_1)$ and $\mathrm{lm}(g_2)$ do not overlap in W, i.e. $W = A\,\mathrm{lm}(g_1)B\,\mathrm{lm}(g_2)C$ for some $A, B, C \in \langle X\rangle$. Then,*

$$h_{A,g_1,B\,\mathrm{lm}(g_2)C}(W) - h_{A\,\mathrm{lm}(g_1)B,g_2,C}(W) \in I_{G,W}.$$

Proof. We denote $c_i = \mathrm{lc}(g_i), m_i = \mathrm{lm}(g_i), t_i = \mathrm{tail}(g_i)$ for $i = 1, 2$. Then, a simple computation yields

$$\begin{aligned}
&h_{A,g_1,Bm_2C}(W) - h_{Am_1B,g_2,C}(W)\\
=&\frac{1}{c_1}At_1Bm_2C - \frac{1}{c_2}Am_1Bt_2C\\
=&\frac{1}{c_1c_2}(c_2At_1Bm_2C + At_1Bt_2C) - \frac{1}{c_1c_2}(c_1Am_1Bt_2C + At_1Bt_2C)\\
=&\frac{1}{c_1c_2}(At_1Bg_2C - Ag_1Bt_2C)
\end{aligned}$$

Since $\mathrm{lm}(At_1Bg_2C) \prec \mathrm{lm}(Ag_1Bg_2C) = W$, we have that $At_1Bg_2C \in I_{G,W}$. Analogously, we get that $Ag_1Bt_2C \in I_{G,W}$, and therefore, also $\frac{1}{c_1c_2}(At_1Bg_2C - Ag_1Bt_2C) \in I_{G,W}$. □

Now, we can proceed to prove Theorem 4.32.

Proof of Theorem 4.32. According to Definition 4.9, we have to show that $\mathrm{sp}(a) = h_{1,g_1,1}(ABC) - h_{A,g_2,C}(ABC) \in I_{G,ABC}$. We do this for each case individually.

Case 1: We note that $L\,\mathrm{lm}(g_3)RBC = ABC$ and that we can write $\mathrm{sp}(a)$ as

$$\mathrm{sp}(a) = \big(h_{1,g_1,1}(ABC) - h_{L,g_3,RBC}(ABC)\big) + \big(h_{L,g_3,RBC}(ABC) - h_{A,g_2,C}(ABC)\big).$$

According to our assumptions, we have that $h_{1,g_1,1}(ABC) - h_{L,g_3,RBC}(ABC) \in I_{G,ABC}$ and since $\mathrm{lm}(g_3)$ and $\mathrm{lm}(g_2)$ do not overlap in ABC, Lemma 4.33 yields that $h_{L,g_3,RBC}(ABC) - h_{A,g_2,C}(ABC) \in I_{G,ABC}$. Hence, also $\mathrm{sp}(a) \in I_{G,ABC}$.

Case 2: We note that $AL\,\mathrm{lm}(g_3)RC = ABC$ and that we can write $\mathrm{sp}(a)$ as

$$\mathrm{sp}(a) = \big(h_{1,g_1,1}(ABC) - h_{AL,g_3,RC}(ABC)\big) + \big(h_{AL,g_3,RC}(ABC) - h_{A,g_2,C}(ABC)\big).$$

According to our assumptions, we have that $h_{1,g_1,1}(ABC) - h_{AL,g_3,RC}(ABC) \in I_{G,ABC}$ and $h_{L,g_3,R}(B) - h_{1,g_2,1}(B) \in I_{G,B}$. Hence,

$$h_{AL,g_3,RC}(ABC) - h_{A,g_2,C}(ABC) = A\big(h_{L,g_3,R}(B) - h_{1,g_2,1}(B)\big)C \in I_{G,ABC},$$

and therefore, also $\mathrm{sp}(a) \in I_{G,ABC}$.

Case 3: We note that $ABL\,\mathrm{lm}(g_3)R = ABC$ and that we can write $\mathrm{sp}(a)$ as

$$\mathrm{sp}(a) = \big(h_{1,g_1,1}(ABC) - h_{ABL,g_3,R}(ABC)\big) + \big(h_{ABL,g_3,R}(ABC) - h_{A,g_2,C}(ABC)\big).$$

According to our assumptions, we have that $h_{1,g_1,1}(ABC) - h_{ABL,g_3,R}(ABC) \in I_{G,ABC}$ and since $\mathrm{lm}(g_3)$ and $\mathrm{lm}(g_2)$ do not overlap in ABC, Lemma 4.33 yields that $h_{ABL,g_3,R}(ABC) - h_{A,g_2,C}(ABC) \in I_{G,ABC}$. Hence, also $\mathrm{sp}(a) \in I_{G,ABC}$.

Case 4: We note that $LUVRC = ABC$ and that we can write $\mathrm{sp}(a)$ as

$$\mathrm{sp}(a) = \big(h_{1,g_1,1}(ABC) - h_{L,g_3,RC}(ABC)\big) + \big(h_{L,g_3,RC}(ABC) - h_{A,g_2,C}(ABC)\big).$$

According to our assumptions, we have that $h_{1,g_1,1}(ABC) - h_{L,g_3,RC}(ABC) \in I_{G,ABC}$ and $h_{1,g_3,R}(UVR) - h_{U,g_2,1}(UVR) \in I_{G,UVR}$. Hence,

$$h_{L,g_3,RC}(ABC) - h_{A,g_2,C}(ABC) = L\big(h_{1,g_3,R}(UVR) - h_{U,g_2,1}(UVR)\big)C \in I_{G,LUVRC} = I_{G,ABC},$$

and therefore, also $\mathrm{sp}(a) \in I_{G,ABC}$.

Case 5: We note that $ALUVR = ABC$ and that we can write $\mathrm{sp}(a)$ as

$$\mathrm{sp}(a) = \big(h_{1,g_1,1}(ABC) - h_{AL,g_3,R}(ABC)\big) + \big(h_{AL,g_3,R}(ABC) - h_{A,g_2,C}(ABC)\big).$$

According to our assumptions, we have that $h_{1,g_1,1}(ABC) - h_{AL,g_3,R}(ABC) \in I_{G,ABC}$ and $h_{1,g_2,V}(LUV) - h_{L,g_3,1}(LUV) \in I_{G,LUV}$. Hence,

$$h_{AL,g_3,R}(ABC) - h_{A,g_2,C}(ABC) = -A\big(h_{1,g_2,V}(LUV) - h_{L,g_3,1}(LUV)\big)R \in I_{G,ALUVR} = I_{G,ABC},$$

and therefore, also $\mathrm{sp}(a) \in I_{G,ABC}$.

Case 6: We note that $A_1LBRC_2 = ABC$ and that we can write $\mathrm{sp}(a)$ as

$$\mathrm{sp}(a) = \big(h_{1,g_1,1}(ABC) - h_{A_1,g_3,C_2}(ABC)\big) + \big(h_{A_1,g_3,C_2}(ABC) - h_{A,g_2,C}(ABC)\big).$$

According to our assumptions, we have that $h_{1,g_1,1}(ABC) - h_{A_1,g_3,C_2}(ABC) \in I_{G,ABC}$ and $h_{1,g_3,1}(LBR) - h_{L,g_2,R}(LBR) \in I_{G,LBR}$. Hence,

$$h_{A_1,g_3,C_2}(ABC) - h_{A,g_2,C}(ABC) = A_1\big(h_{1,g_3,1}(LBR) - h_{L,g_2,R}(LBR)\big)C_2 \in I_{G,A_1LBRC_2} = I_{G,ABC},$$

and therefore, also $\mathrm{sp}(a) \in I_{G,ABC}$. □

So, the previous two theorems allow us to safely delete an ambiguity, given that one or two other ambiguities are ($\preceq$-)resolvable. When we want to use the chain criterion as part of an actual Gröbner basis computation, we could, given an ambiguity $a = (ABC, A, C, g_1, g_2) \in \mathrm{amb}_G$ and $g_3 \in G$ such that $\mathrm{lm}(g_3) \mid ABC$, analyse in which particular case of Theorem 4.31 or Theorem 4.32 we are and explicitly check whether the other ambiguities appearing in this case of the chain criterion are resolvable to decide whether we can delete a or have to process it. However, this is very costly and would most likely take more time than just reducing $\mathrm{sp}(a)$ as we would usually do. Hence, we have to find simpler criteria that assure that the other ambiguities appearing in the particular case of the chain criterion that we are in are resolvable. We note that these other ambiguities are for sure resolvable if we process them during the next step of our Gröbner basis computation. To ensure that this is indeed the case, we have to take care that we do not end up in a cycle of deleting ambiguities, where we remove an ambiguity a_1 assuming that another ambiguity a_2 is resolvable and simultaneously remove a_2 assuming that a_1 is resolvable. To avoid such cycles, we can impose a strict partial ordering on ambiguities and only apply the chain criterion to delete an ambiguity a in situations where a is larger than all ambiguities that the chain criterion relates to a.

Proposition 4.34. *Let $G = \{g_1, \dots, g_n\} \subseteq K\langle X\rangle$ and let $\lessdot$ be a strict partial ordering on amb_G. Furthermore, let $S \subseteq \mathrm{amb}_G$ be the set of all ambiguities $a = (ABC, A, C, g_i, g_j) \in \mathrm{amb}_G$ such that there exists $g_k \in G$ with $\mathrm{lm}(g_k) \mid ABC$ and such that the ambiguities $a_1, a_1' \in \mathrm{amb}_G$, that the chain criterion relates to a and g_k, are both smaller than a with respect to $\lessdot$, i.e. $a_1 \lessdot a$ and $a_1' \lessdot a$ (respectively, such that a_1 is smaller than a if the chain criterion relates only one ambiguity to a and g_k). Then, all ambiguities in amb_G are $\preceq$-resolvable if and only if all ambiguities in $\mathrm{amb}_G \setminus S$ are $\preceq$-resolvable.*

Proof. It is clear that all ambiguities in $\mathrm{amb}_G \setminus S$ are $\preceq$-resolvable if all ambiguities in amb_G are $\preceq$-resolvable. For the other implication, assume that all ambiguities in $\mathrm{amb}_G \setminus S$ are $\preceq$-resolvable but that not all ambiguities in amb_G are $\preceq$-resolvable. Let $a_0 \in \mathrm{amb}_G$ be such an ambiguity. Then, in particular, we have $a_0 \in S$, which implies that the chain criterion is applicable to a_0 and some $g_k \in G$, where the ambiguities $a_1, a_1' \in \mathrm{amb}_G$, that the chain criterion relates to a_0 and g_k, are both smaller than a_0 with respect to $\lessdot$ (respectively, where a_1 is smaller than a_0 if the chain criterion relates only one ambiguity to a and g_k). Since a_0 is not $\preceq$-resolvable, at least one of these ambiguities can also not be $\preceq$-resolvable because of Theorem 4.31 and Theorem 4.32. W.l.o.g. let this be a_1. Then, we must have $a_1 \in S$ and by the same arguments as above, we get the existence of $a_2 \in S$ such that a_2 is not $\preceq$-resolvable and $a_1 \gg a_2$. If we keep doing this process, we obtain a sequence of non $\preceq$-resolvable elements $a_i \in S$, where $a_i \gg a_{i+1}$. Since G is finite, also amb_G and S are finite. This implies that we can do this process only a finite number of times, and consequently, that the sequence obtained in this way has to be finite. Let $a_n \in S$ be the last, and hence, smallest element of this sequence. Then, all ambiguities that the chain criterion relates to a_n and some $g_k \in G$ are $\preceq$-resolvable, because otherwise, at least one of them would have to be in S and would be smaller than a_n. But this implies that also a_n is $\preceq$-resolvable, which is a contradiction. □

One example of a strict partial ordering, that can be used for this purpose, is the following. For a set of polynomials $G = \{g_1, \dots, g_n\} \subseteq K\langle X\rangle$ and $a = (ABC, A, C, g_i, g_j)$,

$a' = (A'B'C', A', C', g_k, g_l) \in \mathrm{amb}_G$, we define

$$a \lessdot a' :\Leftrightarrow \min\{i,j\} < \min\{k,l\}.$$

It is easy to see that $\lessdot$ is indeed a strict partial ordering on amb_G. Based on this ordering, the following simple test assures that the chain criterion is only applied to delete an ambiguity a in situations where a is larger than the ambiguities that the chain criterion relates to a, and consequently, allows us to safely delete a during a Gröbner basis computation without having to make any explicit ($\preceq$-)resolvability checks.

Corollary 4.35. *Let $G = \{g_1, \ldots, g_n\} \subseteq K\langle X\rangle$. Furthermore, let $a = (ABC, A, C, g_i, g_j) \in \mathrm{amb}_G$ and $g_k \in G$ be such that $\mathrm{lm}(g_k) \mid ABC$ and $k < \min\{i,j\}$. Then, a does not have to be considered during a Gröbner basis computation with G as input.*

Proof. Independent of the particular case of the chain criterion, all ambiguities that are related to a and g_k are between g_i and g_k or g_j and g_k. Hence, the assumption $k < \min\{i,j\}$ and the definition of $\lessdot$ imply that a is larger than all these ambiguities with respect to $\lessdot$. Then, the result follows from Proposition 4.34 and the fact that all ambiguities, that are not removed, are processed at some point during the Gröbner basis computation. □

While this is already a very good test that allows to very efficiently apply the chain criterion, it is too strict in some cases. This means that sometimes instances where the chain criterion could be applied remain undetected. In order to improve this performance, we tried different orderings on ambiguities and modifications of Corollary 4.35 on several examples and the following strict partial ordering combined with the criteria stated in Corollary 4.36 and Corollary 4.37 turned out to provide the best tradeoff between efficiency in applying the tests and detecting as many redundant ambiguities as possible.

For a set of polynomials $G = \{g_1, \ldots, g_n\} \subseteq K\langle X\rangle$ and $a = (ABC, A, C, g_i, g_j)$, $a' = (A'B'C', A', C', g_k, g_l) \in \mathrm{amb}_G$, we define $a \lessdot a'$ if one of the following holds.

1. a is an inclusion ambiguity and a' is an overlap ambiguity.
2. a and a' are both overlap ambiguities and
   (a) $|ABC| < |A'B'C'|$, or
   (b) $|ABC| = |A'B'C'|$ and $|A| < |A'|$, or
   (c) $|ABC| = |A'B'C'|$ and $|A| = |A'|$ and $|C| < |C'|$.
3. a and a' are both inclusion ambiguities and
   (a) $|ABC| < |A'B'C'|$, or
   (b) $|ABC| = |A'B'C'|$ and $j < l$.

It is straightforward to check that this is indeed a strict partial ordering on amb_G. We split the corresponding set of conditions to be checked during a Gröbner basis computation into two parts. The criteria to delete an overlap ambiguity are as follows.

Corollary 4.36. *Let $G = \{g_1, \ldots, g_n\} \subseteq K\langle X\rangle$. Furthermore, let $a = (ABC, A, C, g_i, g_j) \in \mathrm{amb}_G$ be an overlap ambiguity and let $g_k \in G$ be such that $\mathrm{lm}(g_k) \mid ABC$ and such that one of the following conditions holds.*

1. *$\mathrm{lm}(g_k) \mid m$ for some $m \in \{A, B, C\}$*
2. *There exist $L, R \in \langle X\rangle$ such that $ABC = L\,\mathrm{lm}(g_k)R$ and*
   (a) *$|L| = 0$ and $|R| < |C|$, or*
   (b) *$0 < |L| < |A|$, or*
   (c) *$|L| \geq |A|$ and $|R| > 0$.*

Then, a does not have to be considered during a Gröbner basis computation with G as input.

Proof. Due to Proposition 4.34 and the fact that all ambiguities, that are not removed, are processed at some point during the Gröbner basis computation, it suffices to show that the conditions above assure that the ambiguity a is larger with respect to $\lessdot$ than all ambiguities that the chain criterion relates to a and g_k. To show this, we consider the different cases of Corollary 4.36 separately.

Case 1. If $\mathrm{lm}(g_k) \mid m$ for some $m \in \{A, B, C\}$, then we are either in case 1,2 or 3 of Theorem 4.31 and in all these cases only inclusion ambiguities are related to a and g_k. Then, it follows by the definition of $\lessdot$ that a is larger than all these ambiguities.

Now, assume that there exist $L, R \in \langle X\rangle$ such that $ABC = L\,\mathrm{lm}(g_k)R$.

Case 2.(a) If $|L| = 0$ and $|R| < |C|$, then we are in case 6 of Theorem 4.31. In this case, the related ambiguity between g_i and g_k is an inclusion ambiguity, since $|L| = 0$, and consequently, smaller than a. The related ambiguity between g_j and g_k is $a' = (\mathrm{lm}(g_k)R, A, R, g_k, g_j)$, which is either an inclusion ambiguity (if $|R| = 0$) or an overlap ambiguity, where $|\,\mathrm{lm}(g_k)R| = |ABC|$ and $|R| < |C|$. In both cases, our ordering assures that we have $a' \lessdot a$.

Case 2.(b) If $0 < |L| < |A|$, then we are either in case 4 or 6 of Theorem 4.31, depending on whether $|R| \geq |C|$ or $|R| < |C|$. First, we note that if $|R| = 0$, we are in a similar situation as in the proof of Case 2.(a) only with the roles of L and R, respectively A and C, swapped. In particular, then the related ambiguity between g_k and g_j is an inclusion ambiguity, since $|R| = 0$, and consequently, smaller than a. The related overlap ambiguity between g_i and g_k is $a' = (L\,\mathrm{lm}(g_k), L, C, g_i, g_k)$, where $|L\,\mathrm{lm}(g_k)| = |ABC|$ and $|L| < |A|$. Hence, $a' \lessdot a$. If $0 < |R| < |C|$, then we can write ABC as $ABC = LA_1BC_2R$ with nonempty A_1, C_2 such that $A_1BC_2 = \mathrm{lm}(g_k)$. In this case, the related ambiguities are the overlap ambiguities $a' = (LA_1BC_2, L, C_2, g_i, g_k)$ and $a'' = (A_1BC_2R, R, A_1, g_k, g_j)$. Since $|L|, |R| > 0$, we have that $|LA_1BC_2| < |ABC|$ and $|A_1BC_2R| < |ABC|$, and therefore, $a' \lessdot a$ and $a'' \lessdot a$. Finally, we consider the case where $|R| \geq |C|$. Then, we are in case 4 of Theorem 4.31, which relates an inclusion ambiguity as well as an overlap ambiguity to a and g_k. The inclusion ambiguity is trivially smaller than a. The overlap ambiguity is $a' = (UVR, U, R, g_j, g_k)$, with U, V such that $\mathrm{lm}(g_k) = UV$ and $A = LU$. Since $|L| > 0$, we have $|UVR| < |LUVR| = |ABC|$, and consequently, $a' \lessdot a$.

Case 2.(c) If $|L| \geq |A|$ and $|R| > 0$, then we are in case 5 of Theorem 4.31, which relates an inclusion ambiguity between g_j and g_k as well as an overlap ambiguity between g_i and g_k

to a and g_k. The inclusion ambiguity is trivially smaller than a. The overlap ambiguity is $a' = (LUV, L, V, g_i, g_k)$, with U, V such that $\mathrm{lm}(g_k) = UV$ and $C = VR$. Since $|R| > 0$, we have $|LUV| < |LUVR| = |ABC|$, and consequently, $a' \lll a$. □

The criteria to delete an inclusion ambiguity look as follows.

Corollary 4.37. *Let $G = \{g_1, \ldots, g_n\} \subseteq K\langle X\rangle$. Furthermore, let $a = (ABC, A, C, g_i, g_j) \in \mathrm{amb}_G$ be an inclusion ambiguity and let $g_k \in G$ with $k < j$ be such that $\mathrm{lm}(g_k) \mid ABC$ and such that one of the following conditions holds.*

1. *$\mathrm{lm}(g_k) \mid m$ for some $m \in \{A, C\}$*
2. *$\mathrm{lm}(g_k) \mid B$ and $|AC| > 0$*
3. *$B \mid \mathrm{lm}(g_k)$ and $|\mathrm{lm}(g_k)| < |ABC|$*

Then, a does not have to be considered during a Gröbner basis computation with G as input.

Proof. As in the proof of Corollary 4.36, we only have to show that the conditions above assure that the ambiguity a is larger with respect to $\lll$ than all ambiguities that the chain criterion relates to a and g_k. To show this, we consider the different cases of Corollary 4.37 separately.

Case 1. If $\mathrm{lm}(g_k) \mid m$ for some $m \in \{A, C\}$, then we are either in case 1 or 3 of Theorem 4.32. In both of these cases, the chain criterion relates an inclusion ambiguity $a' = (ABC, L, R, g_i, g_k)$ with some $L, R \in \langle X\rangle$ to a and g_k. Then, it follows from the assumption $k < j$ and the definition of $\lll$ that $a' \lll a$.

Case 2. If $\mathrm{lm}(g_k) \mid B$ and $|AC| > 0$, then we are in case 2 of Theorem 4.32. In this case, the related ambiguities are inclusion ambiguities of the form $a' = (ABC, L, R, g_i, g_k)$ and $a'' = (B, L', R', g_j, g_k)$ with some $L, L', R, R' \in \langle X\rangle$. Then, $k < j$ implies that a' is smaller than a. Furthermore, $|AC| > 0$ implies that $|B| < |ABC|$, and consequently, that also a'' is smaller than a.

Case 3. If $B \mid \mathrm{lm}(g_k)$ and $|\mathrm{lm}(g_k)| < |ABC|$, then we are in case 6 of Theorem 4.32. In this case, the chain criterion relates the inclusion ambiguities $a' = (ABC, L, R, g_i, g_k)$ and $a'' = (\mathrm{lm}(g_k), L', R', g_k, g_j)$ with some $L, L', R, R' \in \langle X\rangle$ to a and g_k. Then, $k < j$ implies that a' is smaller than a. Furthermore, $|\mathrm{lm}(g_k)| < |ABC|$ implies that also a'' is smaller than a.

□

At the end of the following section, we show the effectiveness of Corollary 4.36 and Corollary 4.37 on the basis of several examples.

4.5.2 Generator reduction

In the previous section, we concerned ourselves with finding criteria that allow us to reduce the number of ambiguities, that actually have to be processed, i.e. checked for resolvability, during a Gröbner basis computation, by deleting redundant ambiguities on the basis of other ambiguities. In this section, we discuss a different approach that also helps us to reduce the number of ambiguities, namely redundant generator reduction. This optimisation strategy, which is also presented in [Xiu12, Section 4.2.2], aims at using elements that are about to be

added to a (partial) Gröbner basis G to remove other elements in G. We additionally propose to occasionally interreduce parts of G. In this way, we keep the (partial) Gröbner basis as small as possible, and therefore, also reduce the total number of ambiguities that can arise.

The theoretical foundation for this section is the following refinement of Proposition 3.46.

Proposition 4.38. *Let $G \subseteq K\langle X\rangle$ be a partial Gröbner basis obtained as an intermediate result during the execution of Algorithm 3 or Algorithm 6. Furthermore, let $g \in G$ and $g' \in K\langle X\rangle$ be such that $g \to_{G\setminus\{g\}} g'$. Then, we can replace G by $(G \setminus \{g\}) \cup \{g'\}$ in the Gröbner basis computation.*

Proof. We denote $G' = (G \setminus \{g\}) \cup \{g'\}$ and let $\tilde{G}$ be the result of Algorithm 3 or Algorithm 6, where we replaced G by G' at some point. To prove Proposition 4.38, we show that $\tilde{G}$ is a Göbner basis of (G). To this end, we first note that $(G) = (\tilde{G})$ follows from Lemma 3.47. Hence, it remains to show that all ambiguities of $\tilde{G}$ are $\preceq$-resolvable. To this end, let $a = (ABC, A, C, f, f') \in \mathrm{amb}_G$ be an ambiguity of $\tilde{G}$. We distinguish between two cases depending on when a was processed during the computation of $\tilde{G}$. If a was processed after G was replaced by G', then a is clearly $\preceq$-resolvable with respect to $\tilde{G}$. If a was processed before G was replaced by G', then we have $\mathrm{sp}(a) \in I_{G,ABC}$. To show that also $\mathrm{sp}(a) \in I_{\tilde{G},ABC}$ holds, we prove that $I_{G,ABC} \subseteq I_{\tilde{G},ABC}$. This statement trivially holds if $g \notin I_{G,ABC}$ because then $I_{G,ABC} \subseteq I_{G',ABC} \subseteq I_{\tilde{G},ABC}$. For the other case, we note that by the properties of the reduction relation, we can write g as $g = g' + l\tilde{g}r$, with $l, r \in \langle X\rangle$ and $\tilde{g} \in G \setminus \{g\}$ such that $\mathrm{lm}(g) \succeq \mathrm{lm}(g'), \mathrm{lm}(\tilde{g})$. Hence, if $g \in I_{G,ABC}$, then also $g', l\tilde{g}r \in I_{\tilde{G},ABC}$, and therefore, $g \in I_{\tilde{G},ABC}$, which implies that $I_{G,ABC} \subseteq I_{\tilde{G},ABC}$. □

In particular, Proposition 4.38 allows us to interreduce a partial Gröbner basis G at any time during a Gröbner basis computation. Thereby, we can remove redundant generators from G and keep this set, and consequently also amb_G, as small as possible. However, interreducing the whole partial Gröbner basis can be computationally expensive, especially when G is already fairly large. Hence, we are seeking a simple criterion that tells us when an element $g \in G$ would be interreduced to zero by $G \setminus \{g\}$. The following corollary provides such a test.

Corollary 4.39. *Let $G \subseteq K\langle X\rangle$ be a partial Gröbner basis obtained as an intermediate result during the execution of Algorithm 3 or Algorithm 6. Furthermore, let $g, g' \in G$ be such that $\mathrm{lm}(g) = A\,\mathrm{lm}(g')C$ for some $A, C \in \langle X\rangle$. If the inclusion ambiguity $(A\,\mathrm{lm}(g')C, A, C, g, g')$ is resolvable with respect to G, then g can be removed from G.*

Proof. We denote $a = (A\,\mathrm{lm}(g')C, A, C, g, g')$ and note that $g \to_{A,g',C} -\frac{1}{\mathrm{lc}(g)}\,\mathrm{sp}(a)$. Hence, Proposition 4.38 allows us to replace G by $G' = (G \setminus \{g\}) \cup \{-\frac{1}{\mathrm{lc}(g)}\,\mathrm{sp}(a)\}$. Since a is resolvable with respect to G and $\mathrm{lm}(\mathrm{sp}(a)) \prec \mathrm{lm}(g)$, we have that $-\frac{1}{\mathrm{lc}(g)}\,\mathrm{sp}(a) \to^*_{G\setminus\{g\}} 0$. So, another application of Proposition 4.38 yields that we can replace G' by $G' \setminus \{-\frac{1}{\mathrm{lc}(g)}\,\mathrm{sp}(a)\} = G \setminus \{g\}$. We do not have to add zero to the partial Gröbner basis as it does not influence further computations anyway. □

Note that we do not have to explicitly check the resolvability of the inclusion ambiguity $(A\,\mathrm{lm}(g')C, A, C, g, g')$. It suffices to ensure that this ambiguity will be processed at some point

of the Gröbner basis computation. Hence, we can safely remove g from G if we add the inclusion ambiguity $(A\,\mathrm{lm}(g')C, A, C, g, g')$ to the ambiguities that are about to be processed.

Intuitively, we would want to apply Corollary 4.39 whenever we add a new element to the Gröbner basis. However, in practice, it turned out to be advantageous to apply this criterion only at the end of each iteration of Algorithm 3 or Algorithm 6, respectively, i.e. when we compute a new set of ambiguities. Additionally, we can also interreduce the elements which have been added to the partial Gröbner basis during an iteration before applying Corollary 4.39. In this way, we can "reveal" more new leading terms, which allows us to reduce the number of generators even more. Also, since the number of newly added elements is usually small compared to the size of the whole partial Gröbner basis, this interreduction does not affect the overall computation time too much.

In the following, we adapt Algorihtm 6 so that it incorporates the generator reduction as discussed above. We note that Algorithm 3 can be adapted in a similar fashion.

Algorithm 14 F4 with generator reduction

Input: a finite set $F \subseteq K\langle X\rangle$
Output if the algorithm terminates: $G \subseteq K\langle X\rangle$ such that G is a Gröbner basis of (F)

1: $G_{old} = \emptyset$
2: $G_{new} = F$
3: **while** $G_{new} \neq \emptyset$ **do**
4: $\quad G_{new} = \mathrm{Interreduce}(G_{new})$
5: $\quad amb = \emptyset$
6: $\quad$ **for** $g \in G_{old}$ **do**
7: $\quad\quad$ **if** there exist $g' \in G_{new}$, $A, C \in \langle X\rangle$ such that $\mathrm{lm}(g) = A\,\mathrm{lm}(g')C$ **then**
8: $\quad\quad\quad G_{old} = G_{old} \setminus \{g\}$
9: $\quad\quad\quad amb = amb \cup \{(A\,\mathrm{lm}(g')C, A, C, g, g')\}$
10: $\quad G_{old} = G_{old} \cup G_{new}$
11: $\quad G_{new} = \emptyset$
12: $\quad critPairs = crit_{G_{old}} \cup \{\mathrm{cp}(a) \mid a \in amb\}$
13: $\quad$ **while** $critPairs \neq \emptyset$ **do**
14: $\quad\quad$ select $C \subseteq critPairs$
15: $\quad\quad critPairs = critPairs \setminus C$
16: $\quad\quad P = \bigcup_{(f_a,g_a)\in C}\{f_a, g_a\}$
17: $\quad\quad \tilde{P} = \mathrm{Reduction}(P, G_{old} \cup G_{new})$
18: $\quad\quad G_{new} = G_{new} \cup \tilde{P}$
19: **return** G_{old}

Theorem 4.40. *At the end of every iteration of the outer* ***while*** *loop of Algorithm 14, all ambiguities of G_{old} are resolvable with respect to $G_{old} \cup G_{new}$. Furthermore, for a finite set $F \subseteq K\langle X\rangle$ such that (F) admits a finite Gröbner basis, Algorithm 6 terminates given F as input and returns a Gröbner basis of (F).*

Proof. The result follows from Theorem 4.29, Proposition 4.38 and Corollary 4.39. □

To end this section, we illustrate the effectiveness of the deletion criteria developed in the previous section and the methods to remove redundant generators discussed in this section by applying them to compute Gröbner bases for several ideals.

Example 4.41. We work over $\mathbb{Q}\langle a, b\rangle$ equipped with $\preceq_{deglex}$ where we order the indeterminates as $b \prec_{lex} a$ and consider 13 different ideals $I_n \subseteq \mathbb{Q}\langle a, b\rangle$, $n = 1, \ldots, 13$. As done in [Xiu12], we generate these ideals by finite systems of generators derived from finite generalised triangular groups taken from [RM02, Theorem 2.12]. This means that, for $n = 1, \ldots, 13$, we have $I_n = (G_n)$, with

$$G_1 = \{a^2 - 1, b^3 - 1, (ababab^2ab^2)^2 - 1\}, \qquad G_2 = \{a^2 - 1, b^3 - 1, (abababb^2)^3 - 1\},$$
$$G_3 = \{a^3 - 1, b^3 - 1, (abab^2)^2 - 1\}, \qquad G_4 = \{a^3 - 1, b^3 - 1, (aba^2b^2)^2 - 1\},$$
$$G_5 = \{a^2 - 1, b^5 - 1, (abab^2)^2 - 1\}, \qquad G_6 = \{a^2 - 1, b^5 - 1, (abababb^4)^2 - 1\},$$
$$G_7 = \{a^2 - 1, b^5 - 1, (abab^2ab^4)^2 - 1\}, \qquad G_8 = \{a^2 - 1, b^4 - 1, (abababb^3)^2 - 1\},$$
$$G_9 = \{a^2 - 1, b^3 - 1, (abab^2)^2 - 1\}, \qquad G_{10} = \{a^2 - 1, b^3 - 1, (abababb^2)^2 - 1\},$$
$$G_{11} = \{a^2 - 1, b^3 - 1, (abababab^2)^2 - 1\}, \qquad G_{12} = \{a^2 - 1, b^3 - 1, (ababab^2abab^2)^2 - 1\},$$
$$G_{13} = \{a^2 - 1, b^3 - 1, (abababababab^2ab^2)^2 - 1\}.$$

In the following table, we list the total number of ambiguities that have to be considered to compute a Gröbner basis for I_n when using Algorithm 6 together with the different optimisation techniques discussed so far. In particular, we compare the total number of ambiguities that are checked for resolvability once using no optimisations (these numbers are listed in the column named "*no optimisation*"), once using the deletion criteria from Corollary 4.36 and Corollary 4.37 (these numbers are listed in the column named "*deletion criteria*"), once using Corollary 4.39 as described in Algorithm 14 (these numbers are listed in the column named "*generator reduction*") and once using deletion criteria and generator reduction together (these numbers are listed in the column named "*combined*"). We also want to relate the total number of ambiguities with the size of the obtained Gröbner bases. However, note that we cannot directly compare the (sizes of the) Gröbner bases that we obtain using the different optimisation strategies as these need not necessarily be equal. So, in the last column we list the number of elements in the (unique) reduced Gröbner basis of I_n. All computations were done with the SageMath version of the `OperatorGB` package (see Chapter 6).

n	*no optimisation*	*deletion criteria*	*generator reduction*	*combined*	*RedGB*
1	18831	608	5473	**562**	35
2	137606	**1362**	46994	1534	96
3	4079	**292**	2586	325	40
4	9906	**480**	4596	603	28
5	2525	**183**	1203	222	21
6	124293	**2759**	75019	3413	164
7	392806	**5885**	101095	7719	164
8	11822	545	4036	**452**	37
9	227	**38**	103	47	5
10	2708	140	610	**116**	15
11	5378	**287**	2238	329	21
12	106563	1182	25651	**1115**	70
13	442728	5470	144542	**3820**	194

For each example, we have highlighted the column where the least number of ambiguities has to be considered to compute a Gröbner basis. Surprisingly, in most cases, this is the column where only deletion criteria are used. Nevertheless, we note that using our implementation, the overall computation time in almost all of these examples, and especially in the larger examples, is the lowest when both optimisation strategies are combined. To get a feeling of the magnitude of speedup these optimisations provide, we note that the computation of the Gröbner basis of I_{13} took more than eight hours using the plain F4 algorithm as described in Section 4.3 but only about 80 seconds when used together with all the optimisation strategies discussed in this chapter. Also, similar results are obtained when the Buchberger algorithm is used instead of the F4 algorithm. So, independent of the actual algorithm, this shows that these optimisations can really speed up a Gröbner basis computation.

4.5.3 Faugère-Lachartre elimination

To end this chapter, we explain an optimisation technique that is only relevant for the F4 algorithm and is concerned with fast reduced row echelon form computations. This is important because, usually, when computing a (partial) Gröbner basis with the F4 algorithm, most of the time is spent on reducing the matrices to reduced row echelon form, especially as the involved matrices can get pretty big. However, these matrices all share certain special properties.

- *sparse*: Typically, most entries in each row are zero.
- *rank deficient*: The matrices do not necessarily have full rank.
- *almost block triangular*: A huge part of each matrix is already in triangular form and many pivots are known prior to the computation.

The Faugère-Lachartre elimination algorithm [FL10] makes use of these properties to compute the reduced row echelon form of a matrix, and hence, is especially efficient on matrices

coming from Gröbner basis computations. Given a matrix M to reduce, the main idea of this algorithm is to first analyse M and to permute all already visible pivots into an upper triangular submatrix A by swapping rows and columns of M. The rest of the matrix is divided into three blocks B, C and D, respectively. So that after this first step we obtain the following decomposition.

$$M \rightarrow \left(\begin{array}{cccc|cccc} * & \cdots & \cdots & * & * & \cdots & \cdots & * \\ & \ddots & A & \vdots & \vdots & & B & \vdots \\ & & \ddots & \vdots & \vdots & & & \vdots \\ & & & * & * & \cdots & \cdots & * \\ \hline * & \cdots & \cdots & * & * & \cdots & \cdots & * \\ \vdots & & C & \vdots & \vdots & & D & \vdots \\ * & \cdots & \cdots & * & * & \cdots & \cdots & * \end{array}\right)$$

Note that, when permuting columns, one has to take care that within each block the columns are still ordered so that the corresponding monomials are decreasing with respect to the monomial ordering used. It is no problem when there are monomials associated to columns of B that are larger than monomials associated to columns of A.

Now, each submatrix is treated separately. Since all elements in the diagonal of A are nonzero, A is invertible and a reduction of these rows yields an identity and $A^{-1}B$. Then, the identity part is used to zero out the submatrix C. Finally, the lower right block is reduced using usual Gauss-Jordan elimination (or any other reduction algorithm). These steps can be visualised as follows.

$$M \rightarrow \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \rightarrow \left(\begin{array}{c|c} I & A^{-1}B \\ \hline C & D \end{array}\right) \rightarrow \left(\begin{array}{c|c} I & A^{-1}B \\ \hline 0 & D - CA^{-1}B \end{array}\right) \rightarrow \left(\begin{array}{c|c} I & A^{-1}B \\ \hline 0 & \mathrm{RRef}(D - CA^{-1}B) \end{array}\right)$$

Since the block D is typically much smaller than A, the reduction of $D - CA^{-1}B$ does not really contribute to the overall time needed by the algorithm. In the end, the previously permuted rows and columns can be permuted back to reconstruct a row echelon form of M. Note that in order to obtain a reduced row echelon form of M, we would also have to reduce $A^{-1}B$ by $\mathrm{RRef}(D - CA^{-1}B)$. However, we can see that all polynomials returned by our reduction procedure must be in the block $\mathrm{RRef}(D - CA^{-1}B)$ because the pivots of the upper rows have not changed and the lower left part has been zeroed out. So, in fact, we do not have to reverse any row or column permutation and can immediately return all polynomials corresponding to the nonzero rows of $\mathrm{RRef}(D - CA^{-1}B)$.

Since we are also interested in cofactor representations, we as well have to compute the transformation matrix T such that

$$T \cdot \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{c|c} I & A^{-1}B \\ \hline 0 & \mathrm{RRef}(D - CA^{-1}B) \end{array}\right).$$

It is easy to see that T is given by

$$T = \left(\begin{array}{c|c} A^{-1} & 0 \\ \hline -\tilde{T}CA^{-1} & \tilde{T} \end{array}\right),$$

with $\tilde{T}$ such that $\tilde{T} \cdot (D - CA^{-1}B) = \text{RRef}(D - CA^{-1}B)$. In particular, we are only interested in cofactor representations of the polynomials corresponding to the rows of $\text{RRef}(D - CA^{-1}B)$. So, actually only the lower part of T is of interest for us as these rows encode the cofactor representations of $\text{RRef}(D - CA^{-1}B)$. We also see that we in fact never need solely A^{-1} but only CA^{-1}, which can be computed efficiently using forward substitution since A is upper triangular.

Chapter 5

Proving operator identities

Now, we can finally dedicate ourselves to the actual goal of this thesis, namely to present an algebraic framework for proving operator identities. In a nutshell, we want to use already known identities of linear operators, our assumptions, to verify other, new identities, our claims, by a purely algebraic computation.

To this end, we model such identities via noncommutative polynomials. Note that we have to use noncommutative polynomials as the composition of linear operators is also noncommutative. Then, proving that a claimed identity holds, basically translates into showing that the polynomial associated to this claim lies in the ideal generated by the polynomials corresponding to our assumptions. In order to verify this ideal membership, we can utilise the theory of Gröbner bases developed in the previous chapters.

The first to apply the theory of noncommutaive Gröbner bases to operator identities were Helton et al. [HW94, HSW98] at the end of the last century. They used Gröbner bases to simplify matrix identities in linear system theory. Around the same time, Gröbner bases techniques were also applied to discover operator identities and to solve matrix equations and matrix completion problems [HS99, Kro01]. At the beginning of this century, Rosenkranz et al. used noncommutative Gröbner bases to solve linear boundary value problems [RBE03].

However, when we transition from linear operators to noncommutative polynomials, we lose important information. When working with actual operators, their domains and codomains impose certain restrictions on which sums and products we are allowed to form but in case of polynomials our computations are not restricted at all. So, in order to derive a statement about operators from a verified ideal membership of noncommutative polynomials, we would have to make sure that all computations done in $K\langle X\rangle$ respect the restrictions imposed by the domains and codomains of the operators. While it has already been observed in the pioneering work that this is the case for the operations executed in the noncommutative version of Buchberger's algorithm, cf. [HSW98, Theorem 25], only recently Raab et al. [RRH19] have developed a simple criterion on the polynomials corresponding to the assumptions and claims that allows us to immediately translate a verified ideal membership to a valid statement about operators. In fact, we then obtain a valid statement about operators for all situations in which the assumptions and claims can be formulated (e.g. matrices of different sizes, bounded linear operators on Hilbert spaces, etc.), all by a single formal computation.

In the first section of this chapter, we summarise the theory developed in [RRH19], which eventually allows us to state a three step procedure to prove operator identities in a fully algebraic fashion. We want to note that we only explain the main results of this theory that are relevant for this thesis. For further details, proofs and generalisations of the concepts presented here, we refer to [RRH19]. In Section 5.2, we then exemplarily apply this method to prove the uniqueness of the Moore-Penrose inverse of a matrix.

5.1 Algebraic framework based on quivers

In this section, we derive a procedure that allows us to prove operator identities in a fully algebraic fashion. To this end, we should first clarify what we mean by "proving operator identities". Typically, this means that we are given some linear operators, such as matrices or bounded operators on Hilbert spaces, having certain properties. These properties have to be expressible as identities involving the operators and the arithmetic operations addition, composition and scaling. We refer to these identities as our *assumptions*. Then, basically only using these assumptions, we want to verify whether other identities of the given operators also hold. These new identities are our *claims*.

Algebraically, we can model such identities by noncommutative polynomials. This is done by uniformly replacing each linear operator by an indeterminate from some set X and forming the difference of the left and right hand side of each identity. We illustrate this procedure on a simple statement about the reverse order law for inner inverses, which we use as a running example throughout this section.

Example 5.1. Let A and B be two linear operators. Furthermore, let A^- and B^- be linear operators satisfying

$$AA^-A = A \qquad \text{and} \qquad BB^-B = B.$$

Such operators A^- and B^- are called inner inverses of A and B, respectively. We want to prove that an inner inverse of AB is given by B^-A^-, i.e. that

$$ABB^-A^-AB = AB$$

holds, if the operator A^-ABB^- is idempotent, which means that

$$A^-ABB^-A^-ABB^- = A^-ABB^-.$$

Hence, our assumptions are

$$AA^-A = A, \qquad BB^-B = B, \qquad A^-ABB^-A^-ABB^- = A^-ABB^-,$$

and our claim is

$$ABB^-A^-AB = AB.$$

All these properties are already phrased as identities as required and we can therefore proceed to translate them into noncommutative polynomials. If we replace A, A^-, B and B^- by a, a^-, b and b^-, respectively, our assumptions translate into

$$aa^-a - a, \qquad bb^-b - b, \qquad a^-abb^-a^-abb^- - a^-abb^- \ \in \ \mathbb{Q}\langle X\rangle,$$

and our claim corresponds to

$$abb^-a^-ab - ab \ \in \ \mathbb{Q}\langle X\rangle,$$

with $X = \{a, a^-, b, b^-\}$.

If we now collect all polynomials corresponding to our assumptions in a set $F \subseteq K\langle X\rangle$, the polynomials corresponding to all consequences of these assumptions, i.e. all operators that can be formed by adding, composing and scaling our assumptions, are contained in (F). Hence, if one of our claims is indeed a consequence of our assumptions its associate polynomial must be contained in (F), and therefore, proving an operator identity comes down to verifying ideal membership.

However, in general, not every element in an ideal generated by polynomials associated to linear operators corresponds to a valid operator itself. As an example, consider two rectangular matrices $A \in K^{n\times m}$ and $B \in K^{m\times k}$ such that $n \neq k$. Then, AB is well-defined but BA does not exist. However, if we denote their corresponding polynomials by $a, b \in \mathbb{Q}\langle a, b\rangle$, then both, ab as well as ba, are contained in (a). So, the ideal membership of the polynomial corresponding to a claim in (F) is a necessary condition for the corresponding operator identity to hold but it is not sufficient. The goal of this section is to state a simple and sufficient condition that ensures that the ideal membership indeed implies the corresponding operator identity.

To this end, we first have to transfer the restrictions, that the domains and codomains of the operators involved in our identities impose, to the indeterminates in X that we used to replace the actual linear operators. This can be done by considering a so-called *labelled quiver*, which is a directed multigraph where edges are labelled by the indeterminates in X.

Definition 5.2. Let V, E, X be sets and $s, t : E \to V$ and $l : E \to X$, respectively, be functions. We call the tuple (V, E, X, s, t, l) a *labelled quiver* with *vertices* V, *edges* E and *labels* X. For an edge $e \in E$, we refer to the vertices $s(e)$ and $t(e)$ as the *source* and *target* of e. Furthermore, the *label* of e is given by $l(e)$.

In the following, we extend the definition of a label from single edges to paths in a labelled quiver such that concatenation of paths in Q corresponds to a multiplication of the labels in $\langle X\rangle$. To this end, we fix a labelled quiver $Q = (V, E, X, s, t, l)$ for the rest of this section.

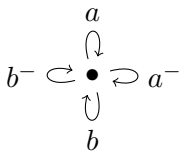
Definition 5.3. Let p be a nonempty path in Q, i.e. $p = e_n \dots e_1$ such that $e_i \in E$ for $1 \leq i \leq n$. Then, the *label* of p is given by

$$l(p) \coloneqq l(e_n) \dots l(e_1) \in \langle X\rangle$$

and its source and target are given by $s(p) \coloneqq s(e_1)$ and $t(p) \coloneqq t(e_n)$, respectively. The empty path ϵ_v describes the path starting at the vertex $v \in V$ and ending in v without passing through any other edge. Its label, source and target are defined to be $l(\epsilon_v) \coloneqq 1 \in \langle X\rangle$, $s(\epsilon_v) \coloneqq v$ and $t(\epsilon_v) \coloneqq v$, respectively.

We can now encode the restrictions imposed by the operators by considering their domains and codomains as the vertices of a labelled quiver and drawing an edge with the label $x \in X$ between two vertices $v, w \in V$ if the operator that has been replaced by x maps between the two spaces that are represented by v and w. To get a better understanding of this procedure, we continue Example 5.1.

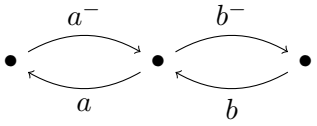
Example 5.4. If all four operators A, A^-, B and B^- involved in Example 5.1 act on a single space $\mathcal{V}$, then this translates into the following labelled quiver.



However, we can also consider another setting where the operators map between different spaces $\mathcal{V}_u$, $\mathcal{V}_v$ and $\mathcal{V}_w$. For example,

$$A : \mathcal{V}_v \to \mathcal{V}_w, \qquad A^- : \mathcal{V}_w \to \mathcal{V}_v, \qquad B : \mathcal{V}_u \to \mathcal{V}_v, \qquad B^- : \mathcal{V}_v \to \mathcal{V}_u.$$

Then, this yields the following labelled quiver.



So, depending on different configurations of the domains and codomains of the operators involved, we obtain different restrictions, and therefore, also different quivers. Note that we draw the vertices of the quivers as simple black dots and do not label them according to the underlying spaces. This should emphasise the fact that we are actually not interested in the particular spaces but only in how they are related via the different operators. This abstraction allows us to treat several different situations with the same quiver. For instance, in the first part of Example 5.4 we do not care whether the particular space all the operators act on is $\mathbb{R}^n$ or $\mathbb{C}^n$ or any other functional space; all these situations correspond to the same quiver. When we want to assign concrete spaces to the vertices of a labelled quiver Q and actual operators that map between these spaces to its edges, we have to consider a concrete *representation* of Q, cf. [DW05].

Definition 5.5. Let $\mathcal{V} = (\mathcal{V}_v)_{v \in V}$ be a family of K-vector spaces and φ be a map that assigns to each $e \in E$ a K-linear map $\varphi(e) : \mathcal{V}_{s(e)} \to \mathcal{V}_{t(e)}$. Then, we call the pair $(\mathcal{V}, \varphi)$ a *representation* of Q.

Remark. Note that every nonempty path $e_n \dots e_1$ in Q induces a K-linear map $\varphi(e_n) \cdot \ldots \cdot \varphi(e_1)$, since the definition of φ ensures that the maps $\varphi(e_{i+1})$ and $\varphi(e_i)$ can be composed for all $1 \leq i < n$. Additionally, for every $v \in V$, the empty path ϵ_v induces the identity map on $\mathcal{V}_v$.

So, a representation of Q helps us to translate the paths in Q, and consequently the monomials in $\langle X \rangle$, back into linear operators. In the course of this process it appears natural to demand that different paths having the same source, target and label get translated into the same operator. While this is trivially the case if every edge has a unique label, the situation becomes a bit more intricate if we allow nonunique labels. Therefore, we introduce what is called a *consistent* representation of Q.

Definition 5.6. Let $\mathcal{R} = (\mathcal{V}, \varphi)$ be a representation of Q. Then, $\mathcal{R}$ is called *consistent with the labelling* l if for all nonempty paths $p = e_n \dots e_1$ and $q = d_n \dots d_1$ in Q with the same source and target, equality of labels, $l(p) = l(q)$, implies equality of the induces K-linear maps, $\varphi(e_n) \cdot \ldots \cdot \varphi(e_1) = \varphi(d_n) \cdot \ldots \cdot \varphi(d_1)$.

Remark. In case that Q has unique labels, every representation of Q is consistent.

Given a consistent representation $\mathcal{R} = (\mathcal{V}, \varphi)$ of a labelled quiver $Q = (V, E, X, s, t, l)$, we can formalise the intuitive concept of transforming a polynomial $f \in K\langle X \rangle$ back into an actual linear operator by replacing the indeterminates in f by the K-linear maps $\varphi(e)$, $e \in E$. This, however, is only possible if f respects the restrictions imposed by Q. This means that all monomials in the support of f have to be labels of paths in Q with the same source and target. In the following, we characterise such f.

Definition 5.7. Let $m \in \langle X \rangle$ be a monomial. The *set of signatures* of m is given by

$$\sigma(m) \coloneqq \{(s(p), t(p)) \mid p \text{ a path in } Q \text{ with } l(p) = m\} \subseteq V \times V.$$

For a polynomial $f \in K\langle X \rangle$, we define the *set of signatures* of f to be

$$\sigma(f) \coloneqq \bigcap_{m \in \operatorname{supp}(f)} \sigma(m) \subseteq V \times V.$$

Additionally, for $v, w \in V$ we define the set

$$K\langle X \rangle_{v,w} \coloneqq \{f \in K\langle X \rangle \mid (v, w) \in \sigma(f)\}.$$

Remark. Note that $\sigma(0) = V \times V$ and $\sigma(1) = \{(v, v) \mid v \in V\}$.

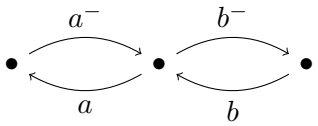
So, a polynomial $f \in K\langle X \rangle$ can be interpreted as a linear operator in the setting described by the labelled quiver Q if and only if its set of signatures is nonempty, or in other words, if and only if f lies in $K\langle X \rangle_{v,w}$ for some $v, w \in V$. We call such a polynomial *compatible* with Q.

Definition 5.8. Let $f \in K\langle X \rangle$. Then, f is *compatible* with the labelled quiver Q if $\sigma(f) \neq \emptyset$. If additionally all monomials $m \in \operatorname{supp}(f)$ have the same set of signatures $\sigma(m)$, then f is called *uniformly compatible* with Q.

Example 5.9. We revisit Example 5.1 and consider the following domains and codomains for the operators involved:

$$A : \mathcal{V}_v \to \mathcal{V}_w, \qquad A^- : \mathcal{V}_w \to \mathcal{V}_v, \qquad B : \mathcal{V}_u \to \mathcal{V}_v, \qquad B^- : \mathcal{V}_v \to \mathcal{V}_u.$$

We have already seen that this constellation can be encoded in the following labelled quiver Q with labels in $X = \{a, a^-, b, b^-\}$.



We can check that the sets of signatures of the assumptions

$$aa^-a - a, \qquad bb^-b - b, \qquad a^-abb^-a^-abb^- - a^-abb^-,$$

as well as the set of signatures of the claim $abb^-a^-ab - ab$, all contain a single element. Hence, all four polynomials are compatible with Q. In fact, it turns out that

$$\begin{aligned}
\sigma(aa^-a) &= \sigma(a) &&= \sigma(aa^-a - a),\\
\sigma(bb^-b) &= \sigma(b) &&= \sigma(bb^-b - b),\\
\sigma(a^-abb^-a^-abb^-) &= \sigma(a^-abb^-) &&= \sigma(a^-abb^-a^-abb^- - a^-abb^-),\\
\sigma(abb^-a^-ab) &= \sigma(ab) &&= \sigma(abb^-a^-ab - ab),
\end{aligned}$$

which shows that all these polynomials are actually uniformly compatible with Q.

Remark. If the edges of a quiver Q have unique labels, each nontrivial monomial has at most one signature. Therefore, in this case, a polynomial without constant term is compatible with Q if and only if it is uniformly compatible with Q.

Given a compatible polynomial $f \in K\langle X\rangle_{v,w}$ for some $v, w \in V$ and a consistent representation $\mathcal{R} = (\mathcal{V}, \varphi)$ of Q, we can plug in the linear maps $\varphi(e)$, $e \in E$, for the indeterminates and obtain an element in $L(\mathcal{V}_v, \mathcal{V}_w)$, the set of K-linear maps from $\mathcal{V}_v$ to $\mathcal{V}_w$. Note that, for fixed $v, w \in V$, the operator obtained in this way is independent of the specific paths chosen, since $\mathcal{R}$ is consistent. In particular, this means that we can define the following map $\varphi_{v,w}$.

Definition 5.10. Let $\mathcal{R} = (\mathcal{V}, \varphi)$ be a consistent representation of Q and $v, w \in V$. We define the K-linear map $\varphi_{v,w} : K\langle X\rangle_{v,w} \to L(\mathcal{V}_v, \mathcal{V}_w)$ by

$$\varphi_{v,w}(l(e_n \dots e_1)) \coloneqq \varphi(e_n) \cdot \ldots \cdot \varphi(e_1)$$

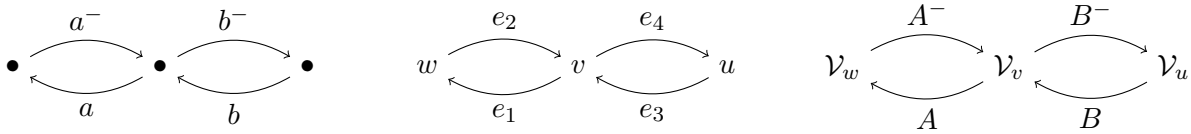
for all nonempty paths $e_n \dots e_1$ in Q starting in v and ending in w. If $v = w$ we define $\varphi_{v,v}(1) \coloneqq \mathrm{id}_{\mathcal{V}_v}$. For $f \in K\langle X\rangle_{v,w}$ we call $\varphi_{v,w}(f)$ a *realisation* of f with respect to the representation $\mathcal{R}$ of Q.

Remark. Note that a polynomial is compatible with Q if and only if it has at least one realisation as a linear operator. Furthermore, if Q has unique labels, then every nonconstant compatible polynomial has a unique realisation.

Example 5.11. Once again, we take a closer look at Example 5.1. We assume that $\mathcal{V}_u$, $\mathcal{V}_v$ and $\mathcal{V}_w$ are K-vector spaces and that

$$A \in L(\mathcal{V}_v, \mathcal{V}_w), \qquad A^- \in L(\mathcal{V}_w, \mathcal{V}_v), \qquad B \in L(\mathcal{V}_u, \mathcal{V}_v), \qquad B^- \in L(\mathcal{V}_v, \mathcal{V}_u).$$

The following three diagrams show a labelled quiver along with a labelling and a representation of it, which can be used to describe these assumptions.



Note that this representation is trivially consistent since every edge has a unique label. Furthermore, we have already checked that the polynomials

$$f_1 = aa^-a - a, \qquad f_2 = bb^-b - b, \qquad f_3 = a^-abb^-a^-abb^- - a^-abb^-, \quad f = abb^-a^-ab - ab$$

are (uniformly) compatible with this quiver. In fact, we have

$$f_1 \in \mathbb{Q}\langle X\rangle_{v,w}, \quad f_2 \in \mathbb{Q}\langle X\rangle_{u,v}, \quad f_3 \in \mathbb{Q}\langle X\rangle_{v,v}, \quad f \in \mathbb{Q}\langle X\rangle_{u,w}.$$

So, we can obtain the following realisations:

$$\varphi_{v,w}(f_1) = \varphi_{v,w}(l(e_1e_2e_1)) - \varphi_{v,w}(l(e_1)) = \varphi(e_1)\cdot\varphi(e_2)\cdot\varphi(e_1) - \varphi(e_1) = AA^-A - A \in L(\mathcal{V}_v, \mathcal{V}_w),$$
$$\varphi_{u,v}(f_2) = \varphi_{u,v}(l(e_3e_4e_3)) - \varphi_{u,v}(l(e_3)) = \varphi(e_3)\cdot\varphi(e_4)\cdot\varphi(e_3) - \varphi(e_3) = BB^-B - B \in L(\mathcal{V}_u, \mathcal{V}_v),$$

and similarly

$$\varphi_{v,v}(f_3) = A^-ABB^-A^-ABB^- - A^-ABB^- \in L(\mathcal{V}_v, \mathcal{V}_v),$$
$$\varphi_{u,w}(f) = ABB^-A^-AB - AB \in L(\mathcal{V}_u, \mathcal{V}_w).$$

Combining all the notions defined so far, we can now finally state the main theorem of this section, which introduces a simple criterion that allows us to immediately translate statements about polynomials into statements about linear operators.

Theorem 5.12. *Let $F \subseteq K\langle X\rangle$ and $f \in (F)$. Furthermore, let Q be a labelled quiver such that f is compatible with Q and all elements of F are uniformly compatible with Q. Then, for all consistent representations $\mathcal{R} = (\mathcal{V}, \varphi)$ of Q such that every realisation of any element of F with respect to $\mathcal{R}$ is zero, we have that every realisation of f with respect to $\mathcal{R}$ is zero.*

Proof. See [RRH19, Theorem 22]. □

To illustrate how Theorem 5.12 helps us to prove operator identities, we apply it to our running example.

Example 5.13. We specify the following setting: given the operators

$$A \in L(\mathcal{V}_v, \mathcal{V}_w), \qquad A^- \in L(\mathcal{V}_w, \mathcal{V}_v), \qquad B \in L(\mathcal{V}_u, \mathcal{V}_v), \qquad B^- \in L(\mathcal{V}_v, \mathcal{V}_u)$$

mapping between the K-linear spaces $\mathcal{V}_u$, $\mathcal{V}_v$ and $\mathcal{V}_w$, where A^- and B^- are inner inverses of A and B, respectively, we want to prove that B^-A^- is an inner inverse of AB if A^-ABB^- is idempotent. In Example 5.1, we have already translated the corresponding operator identities into the following noncommutative polynomials:

$$f_1 = aa^-a - a, \qquad f_2 = bb^-b - b, \qquad f_3 = a^-abb^-a^-abb^- - a^-abb^-, \quad f = abb^-a^-ab - ab.$$

If we take the quiver and the consistent representation described in Example 5.11, uniform compatibility of these polynomials has been shown in Example 5.9. Furthermore, by our assumptions on the operators we know that the (unique) realisations

$$AA^-A - A, \qquad BB^-B - B, \qquad A^-ABB^-A^-ABB^- - A^-ABB^-$$

of f_1, f_2 and f_3 obtained in Example 5.11 are all zero. Hence, if f is in the ideal (f_1, f_2, f_3), Theorem 5.12 implies that also the realisation

$$ABB^-A^-AB - AB$$

of f is zero, which proves that B^-A^- is indeed an inner inverse of AB.

To finish this example, we have to show the ideal membership of f in (f_1, f_2, f_3). This can be done by exploiting the techniques developed in Chapter 4. We can compute the reduced Gröbner basis of (f_1, f_2, f_3) with respect to $\preceq_{deglex}$ where we order the indeterminates as $a \prec_{lex} a^- \prec_{lex} b \prec_{lex} b^-$, which turns out to be

$$G = \{f_1, f_2, f\}.$$

This immediately verifies the ideal membership of f in (f_1, f_2, f_3), and therefore, finishes the proof of our operator identity. Using the methods from Section 4.4, we can even provide an explicit cofactor representation of f in terms of f_1, f_2 and f_3:

$$f = f_1bb^-b - f_1bb^-a^-abb^-b + af_2 - abb^-a^-af_2 + af_3b.$$

In fact, when we start with actual operator identities instead of polynomials, some conditions of Theorem 5.12 become trivial. This allows us to state the following simple three step procedure for proving operator identities in a fully algebraic fashion.

1. Phrase all assumptions on the operators involved as well as the claimed properties in terms of identities involving those operators.

2. Convert these identities as well as the claimed identities to be proven into polynomials by uniformly replacing each individual operator by a unique noncommutative indeterminate from some set X in the differences of the left and right hand sides.

3. Prove that the polynomials corresponding to the claims lie in the ideal (F) generated by the set F of polynomials corresponding to the assumptions.

4. Consider different realisations $\mathcal{R}'$ of the underlying quiver, to obtain analogous statements in different settings.

Then, Theorem 5.12 yields that the claimed identities are indeed consequences of the assumptions.

Note that we do not have to explicitly check the uniform compatibility of the assumptions and the compatibility of the claims with a quiver. This relies on the observation that all polynomials arising from actual operator identities are trivially compatible with the quiver Q describing the situation of operators. In particular, since we replace each operator by an individual indeterminate, including occurring identity operators, the polynomials that we form in step 1 of this procedure do not have a constant term. Note that if an identity operator is replaced, its properties of composition with other operators have to be included in the assumptions. Furthermore, the associate quiver Q has unique labels, and therefore, all polynomials that we form

are uniformly compatible with Q. Moreover, we also do not have to construct a representation $\mathcal{R}$ of Q explicitly because $\mathcal{R}$ is already given implicitly by our translation of the actual operator identities into polynomials and as Q has unique labels, $\mathcal{R}$ is automatically consistent with Q. Additionally, we form each element in F in such a way that its unique realisation with respect to $\mathcal{R}$ is zero. Hence, also this does not have to verified explicitly. However, by considering different representations $\mathcal{R}'$ of Q, for which the realisations of the elements in F are zero, we can even obtain analogous statements for other operators in different settings without having to do any additional computations.

In some cases, however, we might not want to replace each operator by a unique indeterminate. Sometimes it can be advantageous to replace different operators by the same indeterminate. For example, the identity operator can of course act on many different spaces, but because of the very similar nature of these operators, we might want to replace all of them by the same indeterminate. In this case, the associate quiver Q does not have unique labels anymore, and therefore, we have to check the uniform compatibility of the assumptions explicitly. This means that we have to perform the following fourth step before we are able to conclude that the claimed identities indeed hold.

4. Check that all elements of F are uniformly compatible with the quiver Q describing the situation of the given operators.

In fact, the assumption that the elements of F are uniformly compatible with the quiver Q in Theorem 5.12 can be dropped. This was shown in [CHRR20]. Then, however, we have to restrict how the ideal membership of the polynomials associated to the claims in the ideal (F) is verified. In particular, the verification of the ideal membership has to be done by reducing the polynomials corresponding to the claims to zero using a set $G \subseteq (F)$ of polynomials that are Q-order compatible in the sense of the following definition.

Definition 5.14. Let $f \in K\langle X\rangle$ be compatible with Q. Then, f is *Q-order compatible* if $\sigma(\mathrm{lm}(f)) = \sigma(f)$.

Remark. Uniform compatibility implies Q-order compatibility. Additionally, if Q has unique labels, the notions of Q-order compatibility and compatibility are equivalent

We can now state the following generalisation of Theorem 5.12.

Theorem 5.15. *Let $F \subseteq K\langle X\rangle$, $G \subseteq (F)$ and $f \in K\langle X\rangle$ such that $f \to_G^* 0$. Furthermore, let Q be a labelled quiver such that f is compatible with Q and all elements of G are Q-order compatible. Then, for all consistent representations $\mathcal{R} = (\mathcal{V}, \varphi)$ of Q such that every realisation of any element of G with respect to $\mathcal{R}$ is zero, we have that every realisation of f with respect to $\mathcal{R}$ is zero.*

Proof. See [CHRR20, Corollary 1]. □

So, if the assumption of uniform compatibility of the generators of (F) is dropped, the verification of the ideal membership depends on the quiver Q. Note that this is not the case in Theorem 5.12, where ideal membership can be verified independently of any quiver. If F consists

only of Q-order compatible polynomials, then a set $G \subseteq (F)$ as required in Theorem 5.15 can be obtained by applying an adaptation of Algorithm 3 to F and Q, which is called *Q-order compatible completion* and described in [CHRR20]. Given a set G, obtained by an application of Q-order compatible completion to F and Q, and a consistent representation $\mathcal{R} = (\mathcal{V}, \varphi)$ of Q, every realisation of any element of G with respect to $\mathcal{R}$ is zero if and only if every realisation of any element of F with respect to $\mathcal{R}$ is zero. Hence, based on Theorem 5.15, we can replace the third and fourth step in our procedure for proving operator identities by the following instructions.

3. Check that all elements of F are Q-order compatible with the quiver Q describing the situation of the given operators.

4. Compute a set $G \subseteq (F)$ by applying Q-order compatible completion to F and Q, and use it to reduce the polynomials corresponding to the claims to zero.

5.2 A worked example

In this section, we apply the procedure developed in the previous section to prove the uniqueness of the Moore-Penrose inverse of a matrix. This generalised matrix inverse was introduced by Penrose in [Pen55] and is defined as the solution of a certain set of equations. To be more precise, the matrix $A^\dagger \in \mathbb{C}^{n\times m}$ is called a *Moore-Penrose inverse* of $A \in \mathbb{C}^{m\times n}$ if it satisfies the following 4 conditions:

1. $AA^\dagger A = A$,
2. $A^\dagger AA^\dagger = A^\dagger$,
3. $(AA^\dagger)^* = AA^\dagger$,
4. $(A^\dagger A)^* = A^\dagger A$,

where $M^* \in \mathbb{C}^{l\times k}$ denotes the Hermitian transpose of the matrix $M \in \mathbb{C}^{k\times l}$.

We note that a Moore-Penrose inverse exists for every matrix $A \in \mathbb{C}^{m\times n}$ (see [Pen55, Theorem 1]) and in the following, we want to prove that it is actually unique. To this end, we assume that we are given a matrix $A \in \mathbb{C}^{m\times n}$ together with two Moore-Penrose inverses $A_1^\dagger, A_2^\dagger \in \mathbb{C}^{n\times m}$ of A and we want to show that $A_1^\dagger = A_2^\dagger$.

We note that we will replace each operator by a unique indeterminate. Hence, the quiver corresponding to the situation of operators has unique labels, and consequently, we do not have to check uniform compatibility or Q-order compatibility. So, it suffices to perform the three step procedure described after Example 5.13.

Step 1: First, we have to phrase all assumptions on the operators involved in terms of identities. Our only assumptions are that $A_1^\dagger$ as well as $A_2^\dagger$ are Moore-Penrose inverses of A and according to the definition each of these assumptions can be phrased in terms of the four defining equations of the Moore-Penrose inverse. However, whenever an operator identity holds, also the adjoint statement holds. Hence, we have to add the adjoint statements of the defining equations to our assumptions. Note that the third and fourth defining

equation of the Moore-Penrose inverse are self-adjoint and of course, we have to them only once. So, we obtain the following 12 assumptions.

$$\begin{aligned} AA_1^\dagger A = A, \qquad & A_1^\dagger AA_1^\dagger = A_1^\dagger, \qquad & (AA_1^\dagger)^* = AA_1^\dagger, \qquad & (A_1^\dagger A)^* = A_1^\dagger A, \\ AA_2^\dagger A = A, \qquad & A_2^\dagger AA_2^\dagger = A_2^\dagger, \qquad & (AA_2^\dagger)^* = AA_2^\dagger, \qquad & (A_2^\dagger A)^* = A_2^\dagger A. \\ (AA_1^\dagger A)^* = A^*, \qquad & (A_1^\dagger AA_1^\dagger)^* = (A_1^\dagger)^*, \qquad & (AA_2^\dagger A)^* = A^*, \qquad & (A_2^\dagger AA_2^\dagger)^* = (A_2^\dagger)^*, \end{aligned}$$

where the last line contains the four additional adjoint statements. Our claim $A_1^\dagger = A_2^\dagger$ is already phrased as an identity.

Step 2: Next, we have to translate the operator identities into noncommutative polynomials. We do this by replacing the actual operators by the noncommutative indeterminates in $X = \{a, a^*, a_1^\dagger, (a_1^\dagger)^*, a_2^\dagger, (a_2^\dagger)^*\}$. Then, our assumptions translate into

$$\begin{aligned} & aa_1^\dagger a - a, & & a_1^\dagger aa_1^\dagger - a_1^\dagger, & & (a_1^\dagger)^* a^* - aa_1^\dagger, & & a^*(a_1^\dagger)^* - a_1^\dagger a, \\ & aa_2^\dagger a - a, & & a_2^\dagger aa_2^\dagger - a_2^\dagger, & & (a_2^\dagger)^* a^* - aa_2^\dagger, & & a^*(a_2^\dagger)^* - a_2^\dagger a, \\ & a^*(a_1^\dagger)^* a^* - a^*, & & (a_1^\dagger)^* a^*(a_1^\dagger)^* - (a_1^\dagger)^*, & & a^*(a_2^\dagger)^* a^* - a^*, & & (a_2^\dagger)^* a^*(a_2^\dagger)^* - (a_2^\dagger)^*. \end{aligned} \tag{1}$$

We collect all these polynomials in a set $F \subseteq \mathbb{Q}\langle X\rangle$. Analogously, our claim translates into $a_1^\dagger - a_2^\dagger \in \mathbb{Q}\langle X\rangle$.

Step 3: Finally, we have to show that the polynomial corresponding to our claim lies in the ideal generated by the polynomials corresponding to our assumptions, i.e. we have to verify that

$$a_1^\dagger - a_2^\dagger \in (F).$$

Using $\preceq_{deglex}$ as a monomial ordering where we order the indeterminates as

$$a \prec_{lex} a^* \prec_{lex} a_1^\dagger \prec_{lex} (a_1^\dagger)^* \prec_{lex} a_2^\dagger \prec_{lex} (a_2^\dagger)^*,$$

and applying the methods from Chapter 4, we can compute a Gröbner basis G of (F). The reduced Gröbner basis of (F) is given by

$$\begin{aligned} G = \{ & (a_1^\dagger)^* a^* - aa_1^\dagger, a_1^\dagger a - a^*(a_1^\dagger)^*, aa^*(a_1^\dagger)^* - a, a^* aa_1^\dagger - a^*, \\ & aa_1^\dagger(a_1^\dagger)^* - (a_1^\dagger)^*, a^*(a_1^\dagger)^* a_1^\dagger - a_1^\dagger, (a_2^\dagger)^* - (a_1^\dagger)^*, a_1^\dagger - a_2^\dagger\}. \end{aligned}$$

As we can see, our claim is an element of G. This verifies the ideal membership of $a_1^\dagger - a_2^\dagger$ in (F). Additionally, we can apply the techniques developed in Section 4.4 to obtain the following cofactor representation of the claim in terms of the assumptions

$$\begin{aligned} a_1^\dagger - a_2^\dagger = {} & a_1^\dagger f_1 a_2^\dagger - f_2 + a_1^\dagger f_3(aa_2^\dagger - 1) - f_4 a^*(a_2^\dagger)^* a_2^\dagger - a_1^\dagger f_5 a_2^\dagger \\ & + f_6 + a_1^\dagger(a_1^\dagger)^* a^* f_7 + (1 - a_1^\dagger a) f_8 a_2^\dagger + f_9(a_2^\dagger)^* a_2^\dagger - a_1^\dagger(a_1^\dagger)^* f_{11}, \end{aligned} \tag{2}$$

where f_i denotes to the i-th polynomial in (1). This cofactor representation can be seen as a certificate for the ideal membership $a_1^\dagger - a_2^\dagger \in (F)$.

Now, applying Theorem 5.12 yields that the claimed operator identity $A_1^\dagger = A_2^\dagger$ holds, which means that the Moore-Penrose inverse of a matrix $A \in \mathbb{C}^{n\times m}$ is indeed unique. Furthermore, by considering different representations of the quiver associated to this example, Theorem 5.12 also implies an analogous statement for linear operators over (infinite-dimensional) Hilbert spaces or for elements of C^*-algebras. Also, note that the ideal membership of f in (F) holds in $\mathbb{Z}\langle X\rangle$ as well, since all coefficients in (2) are integers. This implies that an analogous statement holds in an arbitrary ring with involution. For further details on this topic, we refer to [KDC07].

While the cofactor representation (2) in this illustrative example is rather short and the computations required to obtain it could still be done by hand, more complicated examples are often virtually impossible to do by hand as the representation of a polynomial in terms of the generators of an ideal can be very involved and hard to find. For example, in joint work in progress with Dragana Cvetković Ilić and her research group on algebraic proofs for generalised inverses [CW17], we obtained certificates for certain ideal memberships related to Hartwig's triple reverse order law [Har86] consisting of several hundred terms. In such cases, the assistance of a computer and dedicated software is very much needed. To this end, we have developed the `OperatorGB` package, which we describe in detail in the following chapter.

Chapter 6

Software

We have implemented all algorithms presented in this thesis in a package named `OperatorGB` [HRR19]. In particular, we provide a version of this package for MATHEMATICA and SAGEMATH, both of which are available at

`http://gregensburger.com/softw/OperatorGB`

along with documentation and examples.

In this chapter, we shall describe both of these versions, their main functionality and usage. In particular, in Section 6.1, we discuss the two main methods provided by the package, which allow to compute noncommutative Gröbner bases together with tracing of cofactors and to almost fully automatically certify operator identities. For further details concerning the functionality of the package, we refer to the documentation. In Section 6.2, we provide some insights concerning the design choices made when implementing these methods and explain some optimisations which helped to obtain reasonably efficient software. Our implementation of noncommutative Gröbner bases is by far not the only one publicly available. Other well-known packages that also provide functionality for computing noncommutative Gröbner bases include the MATHEMATICA package `NCAlgebra` [HS01], the SINGULAR package `Letterplace` [LSL09] and the GAP package `GBNP` [CK16]. For a more extensive survey of computer algebra systems that provide the possibility of performing computations in the noncommutative polynomial ring, we refer to [LSL09]. However, we note that at the time of writing this thesis, there is no other package publicly available that also allows to trace cofactors for arbitrary (partial) Gröbner bases computations in the noncommutative polynomial ring. To end this chapter, we compare our implementations to the other packages mentioned above using some of the benchmark examples presented in [LSL09] as well as some of the examples from Section 4.5.2.

6.1 Main functionality

In this section, we shall describe the two most important methods implemented by our package, namly `Groebner` and `Certify`. The first one allows to compute (partial) Gröbner bases of arbitrary finitely generated ideals in $\mathbb{Q}\langle X\rangle$ together with tracing of cofactors, whereas the latter

implements the procedure described after Theorem 5.12 to automatically verify operator identities. When the package manages to verify an operator identity, it also produces a certificate of this proof, which can then be checked independently. Apart from this basic functionality, the package also includes methods to check (uniform) compatibility of polynomials with quivers as well as some other auxiliary methods for which we refer to the documentation. Note that, so far, the only supported coefficient domain for noncommutative polynomials are the rationals.

The MATHEMATICA version of the package can be loaded by placing it in the current working directory and executing the following line.

```
In[1]:= << OperatorGB.m

        Package OperatorGB version 1.2.0
        Copyright 2019, Institute for Algebra, JKU
        by Clemens Hofstadler, clemens.hofstadler@jku.at
```

Similarly, the SAGEMATH version can be loaded by placing it in a directory where SAGEMATH can find it, e.g. the current working directory, and entering:

```
sage: from OperatorGB import *
```

Since the syntax of both versions of the package is very similar, we discuss the MATHEMATICA commands in more detail in the following sections and only show the usage of the corresponding counterparts in SAGEMATH briefly at the end of each section.

6.1.1 Computing noncommutative Gröbner bases

The first important application of our software is to compute noncommutative (partial) Gröbner bases of arbitrary ideals in $\mathbb{Q}\langle X\rangle$, which can then be used to certify ideal membership via cofactor representations. To illustrate how this can be achieved, we reconsider the polynomials from the running example from Section 5.1. This means, we are given

$$f_1 = aa^-a - a, \qquad f_2 = bb^-b - b, \qquad f_3 = a^-abb^-a^-abb^- - a^-abb^-$$

and we want to verify that

$$f = abb^-a^-ab - ab$$

is contained in the ideal $(F) = (f_1, f_2, f_3) \subseteq \mathbb{Q}\langle a, a^-, b, b^-\rangle$.

In MATHEMATICA, these polynomials can be entered using the built-in noncommutative multiplication.

```
In[2]:= F = {a**a^-**a - a, b** b^-**b - b,
            a^-**a**b** b^-**a^-**a**b**b^- -  a^-**a**b**b^-};
        f = a**b**b^-**a^-**a**b - a**b;
```

Then, a monomial ordering has to be specified. The package supports a degree lexicographic ordering as well as a multigraded lexicographic ordering with arbitrary many blocks. The first ordering can be defined by passing a list of all variables that should be affected by the ordering in ascending order to the function `SetUpRing`. Hence, the command

```
In[3]:= SetUpRing[{a, a⁻, b, b⁻}]

        a < a⁻ < b < b⁻
```

defines a degree lexicographic ordering on $\mathbb{Q}\langle a, a^-, b, b^-\rangle$ where $a \prec_{lex} a^- \prec_{lex} b \prec_{lex} b^-$. If several lists are passed to `SetUpRing`, a multigraded lexicographic ordering is defined, where each list defines one block of variables. So, for example, we can define a multigraded ordering consisting of two blocks, where the first block consisting of a and a^- is smaller than the second block consisting of b and b^-, as follows.

```
In[4]:= SetUpRing[{a, a⁻}, {b, b⁻}]

        a < a⁻ << b < b⁻
```

For this example, however, we stick to a degree lexicographic ordering where $a \prec_{lex} a^- \prec_{lex} b \prec_{lex} b^-$.

To now compute a (partial) Gröbner basis of the ideal generated by `F`, we call the method `Groebner`. We note that in this particular example, we are in fact able to compute a complete Gröbner basis. Besides a list of polynomials, the `Groebner` method takes an empty list (in this example named `cofactorsG`) as an additional parameter. In this list, cofactor representations of all elements in the Gröbner basis (including the generators given as input) are saved. Hence, executing

```
In[5]:= cofactorsG = {};
        G = Groebner[cofactorsG, F];
```

does not only yield a Gröbner basis `G` of the ideal generated by `F`

```
In[6]:= G

Out[6]= {-a + a**a⁻**a, -b + bb⁻**b, -a⁻**a**b**b⁻ + a⁻**a**b**b⁻**a⁻**a**b**b⁻,
        -a**b**b⁻ + a**b**b⁻**a⁻**a**b**b⁻,
        -a⁻**a**b + a⁻**a**b**b⁻**a⁻**a**b, -a**b + a**b**b⁻**a⁻**a**b}
```

but additionally for each element in `G` a cofactor representation in terms of the polynomials in `F`, which is saved in form of a list of triples in `cofactorsG`. So, for example, the first list in `cofactorsG` contains just one triple and corresponds to the first element in `G`, i.e. the first element in `F`, the fourth list in `cofactorsG` corresponds to the fourth element in `G`, which is first element that has been added during the Gröbner basis computation, and so on.

```
In[7]:= cofactorsG[[1]]

Out[7]= {{1, -a + a**a⁻**a, 1}}

In[8]:= cofactorsG[[4]]
```

```
Out[8]= {{a, -a^**a**b**b^ + a^**a**b**b^**a^**a**b**b^, 1},
         {-1, -a + a**a^**a, b**b^**a^**a**b**b^}, {1, -a + a**a^**a, b**b^}}
```

Remark. The `Groebner` method can also take several optional arguments as input. Among other things, these arguments can be used to limit the number of iterations of the outer `while` loop (Line 3 in Algorithm 3 and Algorithm 6) or to impose a degree bound on the ambiguities that are considered. We refer to the documentation for more information.

Having the Gröbner basis `G`, we can verify the ideal membership of `f` in the ideal generated by `F` by computing a normal form of `f` with respect to `G` and checking whether this normal form is zero. A normal form can be computed using the command `ReducedForm`, which, besides the polynomial `f` to be reduced and the set of reductors `G`, also takes an empty list as input. In this list, a cofactor representation of the difference of `f` and the computed normal form in terms of `G` is saved.

```
In[9]:= cofactorsF = {};
        ReducedForm[cofactorsF,G,f]

Out[9]= 0
```

As can be seen, we obtain zero, which proves the ideal membership. Additionally, `cofactorsF` contains a cofactor representation of $\texttt{f} - 0 = \texttt{f}$ in terms of `G`, which is just one triple as `f` was already an element of `G`.

```
In[10]:= cofactorsF

Out[10]= {{1, -a**b + a**b**b^**a^**a**b, 1}}
```

However, usually, we want to have a linear combination consisting of the generators of the ideal and not in terms of a Gröbner basis. This can be achieved using the method `Rewrite`, which takes the list `cofactorsF` from the reduction and the list `cofactorsG` obtained from calling `Groebner[cofactorsG,F]` as input and returns a cofactor representation of `f` in terms of `F`.

```
In[11]:= certificate = Rewrite[cofactorsF, cofactorsG]

Out[11]= {{a, -b + b**b^**b, 1}, {-a**b**b^**a^**a, -b + b**b^**b, 1},
          {a, -a^**a**b**b^ + a^**a**b**b^**a^**a**b**b^, b},
          {-1, -a + a**a^**a, b**b^**a^**a**b**b^**b}, {1, -a + a**a^**a, b**b^**b}}
```

This linear combination now serves as a certificate for the ideal membership of `f` in the ideal generated by `F`. We can check that it still yields the polynomial `f` that we initially reduced using the `MultiplyOut` command.

```
In[12]:= MultiplyOut[certificate] === f
```

```
Out[12]= True
```

A SageMath session executing the same commands looks as follows. Note that before being able to enter the polynomials, first a `FreeAlgebra` has to defined which serves as the parent for the noncommutative polynomials. We have also replaced the variables `a⁻` and `b⁻` by `aa` and `bb`, respectively. Furthermore, due to intermediate interreductions the Gröbner basis `G` computed by the SageMath version of the package looks different to the one obtained in Mathematica.

```
# defining the free algebra
sage: A = FreeAlgebra(QQ,4,['a','aa','b','bb'])
sage: a,aa,b,bb = A.gens()

# entering the polynomials
sage: F = [a*aa*a - a, b*bb*b - b, aa*a*b*bb*aa*a*b*bb - aa*a*b*bb]
sage: f = a*b*bb*aa*a*b - a*b

# defining the monomial ordering
sage: SetUpRing([a,aa,b,bb])
a < aa < b < bb

# computing a Groebner basis
sage: cofactorsG = []
sage: G = Groebner(cofactorsG,F)
sage: G
[-a + a*aa*a, -b + b*bb*b, -a*b + a*b*bb*aa*a*b]

# reducing the polynomial f
sage: cofactorsF = []
sage: ReducedForm(cofactorsF,G,f)
0
sage: cofactorsF
[(1, -a*b + a*b*bb*aa*a*b, 1)]

# rewriting the cofactors
sage: certificate = Rewrite(cofactorsF,cofactorsG)
sage: certificate
[(-1, -a + a*aa*a, b*bb*aa*a*b*bb*b),
  (1, -a + a*aa*a, b*bb*b),
  (a, -aa*a*b*bb + aa*a*b*bb*aa*a*b*bb, b),
  (a, -b + b*bb*b, 1),
  (-a*b*bb*aa*a, -b + b*bb*b, 1)]

# checking that the result is correct
sage: MultiplyOut(certificate) == f
```

```
True
```

6.1.2 Certifying operator identities

The second main application and probably the standard use-case of our package is to almost fully automatically certify operator identities using the method `Certify`. For illustrational purposes, we again consider the running example from Section 5.1.

So, we are given the operators

$$A \in L(\mathcal{V}_v, \mathcal{V}_w), \qquad A^- \in L(\mathcal{V}_w, \mathcal{V}_v), \qquad B \in L(\mathcal{V}_u, \mathcal{V}_v), \qquad B^- \in L(\mathcal{V}_v, \mathcal{V}_u)$$

mapping between the K-linear spaces $\mathcal{V}_u$, $\mathcal{V}_v$ and $\mathcal{V}_w$, where A^- and B^- are inner inverses of A and B, respectively, and we want to prove that B^-A^- is an inner inverse of AB if A^-ABB^- is idempotent.

The translation of the actual operator identities into noncommutative polynomials still has to be done by hand but from then on everything is automatised. Hence, we start by entering the polynomials corresponding to the identities above into MATHEMATICA.

```
In[1]:= assumptions = {a**a¯**a - a, b** b¯**b - b,
            a¯**a**b** b¯**a¯**a**b**b¯ -  a¯**a**b**b¯};
        claim = a**b**b¯**a¯**a**b - a**b;
```

Since our package does not assume that every operator has been replaced by a unique indeterminate, we have to provide a quiver `Q` against which compatibility of the claim and uniform compatibility of the assumptions is checked. Such a quiver can be defined by providing a list of triples. For every operator L appearing in our operator identities, we have to provide a triple of the form $\{l, s, t\}$, where l is the name of the indeterminate that we have replaced L with, s is the codomain of L and t the domain of L. Hence, in case of our example, the corresponding quiver can be set up as follows.

```
In[2]:= Q = {{a,Vᵥ,Vᵂ}, {a¯,Vᵂ,Vᵥ}, {b,Vᵤ,Vᵥ}, {b¯,Vᵥ,Vᵤ}};
```

Now, we can call the method `Certify`, which takes a set of assumptions, a claim (all in form of noncommutative polynomials) and a quiver `Q` as input and automatically executes the steps 3 and 4 of the procedure described after Theorem 5.12. The return value of `Certify` is either a cofactor representation of the claim in terms of the assumptions, which proves that the corresponding claimed operator identity is indeed a consequence of the assumed identities, or `$Failed` (respectively, `False` in SAGEMATH), indicating that the program was not able to prove that the claimed identity follows from the assumptions.

```
In[3]:= certificate = Certify[assumptions,claim,Q]

Out[3]= {{a, -b + b**b¯**b, 1}, {-a**b**b¯**a¯**a, -b + b**b¯**b, 1},
        {a, -a¯**a**b**b¯ + a¯**a**b**b¯**a¯**a**b**b¯, b},
        {-1, -a + a**a¯**a, b**b¯**a¯**a**b**b¯**b}, {1, -a + a**a¯**a, b**b¯**b}}
```

As before, we can check that this is indeed a cofactor representation of our claim by multiplying it out.

```
In[4]:= MultiplyOut[certificate] === claim

Out[4]= True
```

In SageMath the same commands can be executed as follows.

```
# defining the free algebra
sage: A = FreeAlgebra(QQ,4,['a','aa','b','bb'])
sage: a,aa,b,bb = A.gens()

# entering the assumptions and the claim
sage: assumptions = [a*aa*a - a, b*bb*b - b, aa*a*b*bb*aa*a*b*bb - aa*a*b*bb]
sage: claim = a*b*bb*aa*a*b - a*b

# defining the quiver
sage: Q = Quiver([('a','Vv','Vw'),('aa','Vw','Vv'),('b','Vu','Vv'),('bb','Vv','Vu')])

# calling Certify
sage: certificate = Certify(assumptions,claim,Q)
sage: certificate
[[(-1, -a + a*aa*a, b*bb*aa*a*b*bb*b),
  (1, -a + a*aa*a, b*bb*b),
  (a, -aa*a*b*bb + aa*a*b*bb*aa*a*b*bb, b),
  (a, -b + b*bb*b, 1),
  (-a*b*bb*aa*a, -b + b*bb*b, 1)]]

# checking that the result is correct
sage: MultiplyOut(certificate) == claim
True
```

6.2 Implementation details

In this section, we summarise some important facts about the data structures used and some code optimisations done in order to improve the performance of the `Groebner` method.

First of all, we note that the Mathematica version of the package implements the Buchberger algorithm whereas the SageMath version implements the F4 algorithm. The reason for this is that Mathematica is particularly good at pattern matching, which is the foundation for polynomial reduction as done in the Buchberger algorithm, but comparatively slow when it comes to sparse linear algebra and matrix normal form computations. SageMath, however, provides fast algorithms to efficiently compute reduced row echelon forms of sparse matrices.

Regarding the data structures used in the MATHEMATICA version of the package, there is not much to say as in MATHEMATICA everything is internally saved as a list anyway. However, we note that in our procedures we do not use MATHEMATICA's built-in noncommutative multiplication, as it is very inflexible and does not provide a lot of support. Instead, we use our own data structure to represent noncommutative monomials. This data structure automatically flattens nested products or pulls out coefficients for example. Polynomial reduction is implemented via replacement rules, which allows us to fully exploit MATHEMATICA's pattern matching capabilities. When such a replacement rule is applied, the corresponding polynomial gets reduced and simultaneously the cofactors of the reduction are saved in a separate list. In this way, we can efficiently trace cofactors throughout the whole Gröbner basis computation.

The data structures used in the SAGEMATH version of the package are a bit more intricate. For example, we represent noncommutative monomials in form of strings. This has the big advantage that divisibility checks of monomials, which have to be done very often, boil down to checking whether one string is a substring of another string and this can be done very efficiently. So, we store a monomial as a string consisting of all variables appearing in the monomial, each separated by a special character, in our case this is "`*`". Additionally, for divisibly checks to work correctly, we add this special character also at the very beginning and the very end of each monomial. As an example, the monomial *aba* gets converted into `"*a*b*a*"`. The empty word becomes `"*"`. The data structure representing a term in a noncommutative polynomial then consists of such a string and a coefficient stored as a SAGEMATH rational. A noncommutative polynomial is simply a list of such terms. At first, we tried storing a monomial as a tuple of symbolic variables but with this approach the divisibility checks were way too slow, probably due to costly equality checks of symbolic variables.

One important optimisation done in the MATHEMATICA version of the package is the parallelisation of certain parts of the computation. In particular, the computation of the ambiguities as well as the computation of the S-polynomials are parallelised. To this end, we use the already built-in MATHEMATICA commands `ParallelMap` and `Parallelize`. These functions basically take care of all things related to parallelising the computations without the user having to worry about anything. It is only recommended to set the option `DistributedContexts` to `Automatic` so that all needed variables and function definitions get shared among all kernels automatically. Surprisingly, it turned out that parallelising the reductions done in the CheckResolvability subroutine does not yield a significant improvement. So far, in the SAGEMATH version of the package nothing is parallelised but we note that the only part of the `Groebner` routine that would be worth parallelising is the computation of the reduced row echelon form or the rewriting of the cofactors at the end of the procedure (if this is even possible).

Another interesting and probably at first glance not completely obvious optimisation in MATHEMATICA was to get rid of passing large arguments to functions. The reason for this is that in MATHEMATICA function arguments get evaluated by default, which can be very time consuming in case of more complicated expressions or large lists.

We could also achieve a significant (and surprising) improvement by replacing all `Table` commands by `Map` whenever large lists have to be generated. When it comes to incrementally generating lists, we could also save time by replacing calls to `AppendTo` by `Sow & Reap`, which on the one hand simplified the code, as some complicated `Hold` statements were no longer needed,

and on the other hand avoided a lot of unnecessary internal copying. We also use the `Sow & Reap` commands to trace the cofactors during (partial) Gröbner bases computations. This is done by forming replacement rules, which do not only replace the leading term of a polynomial by its tail, but at the same time also sow the cofactors used during this reduction. When these replacement rules are then used to reduce a polynomial, all cofactors of this reduction can be collected simply by calling `Reap`.

When it comes to optimisations done in the SAGEMATH version of the package, it is worth mentioning that it is crucial to use sparse matrices and exploit SAGEMATH's efficient algorithms for such matrices. Surprisingly, it is more efficient to compute the inverse of the submatrix A in the Faugère-Lachartre elimination (see Section 4.5.3) by appending an identity matrix to the right side of A and using the `rref` method instead of the explicit `inverse` method, most likely because the latter is not optimised for sparse matrices. Apart from that, not much was done in terms of code optimisations of the SAGEMATH version of the package. So, there is probably still room for improvement, especially when it comes to rewriting large cofactor representations.

6.3 Comparison

In this section, we compare the following packages for noncommutative Gröbner bases computations.

- `GBNP` version 1.0.3 running on GAP version 4.10.1
- `Letterplace` running on SINGULAR 4.1.2
- `NCAlgebra` version 5.0.4 running on MATHEMATICA 12.0.0
- `OperatorGB` version 1.2.0 (in the following abbreviated by `OGB Mathematica`) running on MATHEMATICA 12.0.0
- `OperatorGB` version 1.2.0 (in the following abbreviated by `OGB SageMath`) running on SAGEMATH 9.0

In particular, we measure the time needed to compute a (partial) Gröbner basis for the following benchmark examples taken from [LSL09].

Example	Generators of the ideal
`braid3`	$yxy - zyz, xyx - zxy, zxz - yzx, x^3 + y^3 + z^3 + xyz$
`braid4`	$yxy - zyz, xyz - zxy, zxz - yzx, x^3 + y^3 + z^3 + xyz$
`lp1`	$z^4 + yxyx - xy^2x - 3zyxz, x^3yxy - xyx, zyx - xyz + zxz$
`lv2`	$xy + yz, x^2 + xy - yx - y^2$

As done in [LSL09], we only compute truncated Gröbner bases of these homogeneous ideals. The designated degree bounds are indicated by the number after the "`-`" in the name of each example in Table 6.1. So, for example `lp1-10` means that we compute a partial Gröbner basis

of the example `lp1` up to degree 10. For all examples, we fix $\preceq_{deglex}$ as a monomial ordering where we order the indeterminates as $x \prec_{lex} y \prec_{lex} z$ and work over the coefficient field $\mathbb{Q}$.

Since `NCAlgebra` does not provide the functionality to compute truncated Gröbner bases, we cannot perform these computations with this package. Hence, we also reconsider some of the ideals from Example 4.41 as these all admit a finite Gröbner basis. We add the following examples to our benchmark tests.

Example	Generators of the ideal
`tri2`	$a^2-1, b^3-1, (ababab^2)^3-1$
`tri3`	$a^3-1, b^3-1, (abab^2)^2-1$
`tri12`	$a^2-1, b^3-1, (ababab^2abab^2)^2-1$
`tri13`	$a^2-1, b^3-1, (abababababab^2ab^2)^2-1$

As for the other examples, we fix $\preceq_{deglex}$ as a monomial ordering where we order the indeterminates as $b \prec_{lex} a$ and work over the coefficient field $\mathbb{Q}$.

The following table gives the timings for computing a (truncated) Gröbner basis for the examples stated above in the format "minutes : seconds". A timing of the form "> xx:xx" indicates that the computation was aborted after this certain time. To be able to better compare our algorithms to the other packages, which do not trace cofactors during their Gröbner bases computations, we provide timings for our methods once with tracing of cofactors (the corresponding columns are indicated by "w/ cf") and once without tracing of cofactors (the corresponding columns are indicated by "w/o cf"). We also note that all tests were performed on a laptop equipped with a 2.7 GHz Quad-Core Intel Core i7 Processor with 16 GB of RAM running macOS.

Example	`GBNP`	`Letterplace`	`NCAlgebra`	`OGB Mathematica`		`OGB SageMath`	
				w/ cf	w/o cf	w/ cf	w/o cf
`braid3-9`	0:01	0:01	–	0:26	0:08	0:08	0:02
`braid3-10`	0:11	0:01	–	> 100:00	3:58	23:14	0:55
`braid4-10`	0:03	0:01	–	0:29	0:12	0:17	0:05
`braid4-11`	0:13	0:01	–	5:15	1:13	2:16	0:24
`lp1-10`	0:01	0:01	–	0:04	0:03	0:02	0:01
`lv2-15`	0:03	0:01	–	0:01	0:01	0:01	0:01
`tri2`	0:01	0:06	9:19	1:34	0:58	0:33	0:06
`tri3`	0:01	0:01	0:21	0:01	0:01	0:01	0:01
`tri12`	0:01	0:01	2:40	0:37	0:26	0:20	0:04
`tri13`	0:02	0:39	64:41	6:15	3:45	1:23	0:40

Table 6.1: Running times for computing (partial) Gröbner bases for several benchmark examples.

As can be seen, our algorithms clearly outperform `NCAlgebra` but cannot compete with `GBNP` and `Letterplace`. It also becomes obvious that tracing cofactors, especially in the larger homogeneous examples, can become very costly. We are working to further improve this part

of our implementation. Nevertheless, we consider the results obtained from these benchmark examples to be encouraging that our package can indeed be useful in practice; especially when it comes to verifying operator identities, where the required (partial) Gröbner basis computations seem to be quite efficient.

Bibliography

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, Cambridge, 1998.

[Ber78] George M. Bergman. *The diamond lemma for ring theory.* Advances in Mathematics 29, pp. 178–218, 1978.

[Bok76] Leonid A. Bokut'. *Imbeddings into simple associative algebras.* Algebra i Logika, 15(2), pp. 117–142, 1976

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, University of Innsbruck, Austria, 1965.

[Buc79] Bruno Buchberger. *A criterion for detecting unnecessary reductions in the construction of Gröbner bases.* In Proceedings of EUROSAM'79, pp. 3–21, 1979.

[CHRR20] Cyrille Chenavier, Clemens Hofstadler, Clemens G. Raab, and Georg Regensburger. *Compatible rewriting of noncommutative polynomials for proving operator identities.* arXiv:2002.03626, 2020.

[CK16] Arjeh M. Cohen and Jan Willem Knopper. *Documentation on the GBNP package.* Available at `http://mathdox.org/gbnp/`, 2016.

[CW17] Dragana S. Cvetković Ilić and Yimin Wei. *Algebraic properties of generalized inverses.* Springer, Singapore, 2017.

[DW05] Harm Derksen and Jerzy Weyman. *Quiver representations.* Notices of the American Mathematical Society 52, pp. 200–206, 2005.

[Fau99] Jean-Charles Faugère. *A new efficient algorithm for computing Gröbner bases (F4).* Journal of Pure and Applied Algebra 139, pp. 61–88, 1999.

[FL10] Jean-Charles Faugère and Sylvain Lachartre. *Parallel Gaussian elimination for Gröbner bases computations in finite fields.* Proceedings of the 4th International Workshop on Parallel and Symbolic Computation, pp. 89–97, 2010.

[GM88] Rüdiger Gebauer and Hans Michael Möller. *On an installation of Buchberger's algorithm.* Journal of Symbolic Computation 6, pp. 275–286, 1988.

[GMNRT91] Alessandro Giovini, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. *"One sugar cube, please" or selection strategies in the Buchberger algorithm.* Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation (ISSAC '91), Stephen M. Watt (Ed.), ACM, pp. 49–54, 1991.

[GHM19] Yves Guiraud, Eric Hoffbeck and Philippe Malbos. *Convergent presentations and polygraphic resolutions of associative algebras.* Mathematische Zeitschrift 293, pp. 113–179, 2019.

[Har86] Robert E. Hartwig. *The reverse order law revisited.* Linear Algebra and its Applications 76, pp. 241–246, 1986.

[HRR19] Clemens Hofstadler, Clemens G. Raab and Georg Regensburger. *Certifying operator identities via noncommutative Gröbner bases.* ACM Communications in Computer Algebra 53, pp. 49–52, 2019.

[Hos18] Jamal Hossein Poor. *Tensor reduction systems for rings of linear operators.* PhD thesis, Johannes Kepler University Linz, Austria, 2018.

[HRR18] Jamal Hossein Poor, Clemens G. Raab, and Georg Regensburger. *Algorithmic operator algebras via normal forms in tensor rings.* Journal of Symbolic Computation 85, pp. 247–274, 2018.

[HS99] J. William Helton and Mark Stankus. *Computer assistance for "discovering" formulas in system engineering and operator theory.* Journal of Functional Analysis 161, pp. 289–363, 1999.

[HS01] J. William Helton and Mark Stankus. *A noncommutative Gröbner basis package for* MATHEMATICA. Available at `http://math.ucsd.edu/~ncalg/`.

[HSW98] J. William Helton, Mark Stankus, and John J. Wavrik. *Computer simplification of formulas in linear systems theory.* IEEE Trans. Automat. Control 43, pp. 302–314, 1998.

[HW94] J. William Helton and John J. Wavrik. *Rules for computer simplification of the formulas in operator model theory and linear systems.* In Nonselfadjoint operators and related topics, pp. 325–354, Birkhäuser, Basel, 1994.

[KDC07] Jerry J. Koliha, Dragan Djordjević, and Dragana Cvetković. *Moore-Penrose inverse in rings with involution.* Linear Algebra and its Applications 426, pp. 371–381, 2007.

[Kro01] Frank Dell Kronewitter. *Using noncommutative Gröbner bases in solving partially prescribed matrix inverse completion problems.* Linear Algebra and its Applications 338, pp. 171–199, 2001.

[LSL09] Roberto La Scala and Viktor Levandovskyy. *Letterplace ideals and non-commutative Gröbner bases.* Journal of Symbolic Computation 44, pp. 1374–1393, 2009.

[LSL13] Roberto La Scala and Viktor Levandovskyy. *Skew polynomial rings, Gröbner bases and the letterplace embedding of the free associative algebra.* Journal of Symbolic Computation 48, pp. 110–131, 2013.

[Lev05] Viktor Levandovskyy. *Non-commutative computer algebra for polynomial algebras: Gröbner bases, applications and implementation.* PhD thesis, Technische Universität Kaiserslautern, Germany, 2005.

[MM82] Ernst W. Mayr and Albert R. Meyer *The complexity of the word problems for commutative semigroups and polynomial ideals.* Advances in Mathematics 46, pp. 305–329, 1982.

[Mor86] Ferdinando Mora. *Groebner bases for non-commutative polynomial rings.* Proceedings of the 3rd International Conference on Algebraic Algorithms and Error-Correcting Codes (AAECC-3), Jacques Calmet (Ed.), Springer-Verlag, pp. 353–362, 1986.

[Mor94] Teo Mora. *An introduction to commutative and noncommutative Gröbner Bases.* Journal of Theoretical Computer Science 134, pp. 131–173, 1994.

[Mor16] Teo Mora. *Solving polynomial equation systems IV.* Cambridge University Press, Cambridge, 2016.

[Nor01] Patrik Nordbeck. *Canonical bases for algebraic computations.* PhD thesis, Lund University, Sweden, 2001.

[Pen55] Roger Penrose. *A generalized inverse for matrices.* Proceedings of the Cambridge Philosophical Society 51, pp. 406–413, 1955.

[RRH19] Clemens G. Raab, Georg Regensburger and Jamal Hossein Poor. *Formal proofs of operator identities by a single formal computation.* arXiv:1910.06165, 2019.

[RM02] Gerhard Rosenberger and Martin Scheer. *Classification of the finite generalized tetrahedron groups.* Combinatorial and geometric group theory, pp. 207–229, AMS, Providence, 2002.

[RBE03] Markus Rosenkranz, Bruno Buchberger, and Heinz W. Engl. *Solving linear boundary value problems via non-commutative Gröbner bases.* Applicable Analysis. An International Journal 82, pp. 655–675, 2003.

[Win96] Franz Winkler. *Polynomial algorithms in computer algebra.* Springer-Verlag Wien, 1996.

[Xiu12] Xingqiang Xiu. *Non-commutative Gröbner bases and applications.* PhD thesis, University of Passau, Germany, 2012.