Über Gröbnerwalkalgorithmen

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

von
I Made Sulandra

Saarbrücken, im Jahre 2003

Tag des Kolloquiums	:	31. Oktober 2003
Dekan	:	Prof. Dr. Philipp Slusallek
Berichterstatter	:	Prof. Dr. Wolfram Decker
		Prof. Dr. Theo de Jong

Gedruckt mit Unterstützung des Deutschen Akademischen Austauschdientes

Introduction

In many applications of mathematics, problems can be described by systems of linear equations. In this case, the structure of the solution sets is well-understood and there are efficient algorithms to find the solutions, for example, Gaussian elimination.

Other problems do not lead to systems of linear equations, but to systems of polynomial equations (e.g. in robotics).

In contrast to a system of linear equations, solving a system of polynomial equations is much more difficult. Algebraic geometry provides plenty of methods to study the structure of the solutions sets. Further, there are important recent developments in computer algebra (and numerical analysis). Fast computers and the design of new algorithms, in particular for solving systems of polynomial equations, allow one to solve many complicated practical problems and to investigate theoretical problems.

To find solutions of a system of polynomial equations depends on geometrical techniques, for example projection and continuation. Algebraic geometry describes a connection between geometry and algebra. In this context, sets of generators of an ideal with certain „good "properties play an important role. These sets are called *Gröbner bases*

Gröbner bases and Buchberger's algorithm are an indispensable component of modern algebra; they are implemented in many computer algebra systems.

A Gröbner basis of an ideal is not unique and depends in particular on the choice of a monomial ordering. For most applications, any Gröbner basis will do. For those applications the reverse lexicographic order seems to be the most efficient. For some applications, however, Gröbner bases with special properties and thus special monomial orders are needed. To explicitly solve a system of polynomial equations via elimination, for instance, one typically makes use of the lexicographic order. Usually, the computation of a Gröbner basis with respect to a lexicographic ordering costs a lot of time and memory. There are two approaches to overcome this problem:

The first approach, due to Traverso (1997), uses information which arises from the Hilbert function of a quotient ring R/I (*Hilbert-driven version of Buchberger's algorithm*), where I is a homogenous ideal in the polynomial ring R. First one computes a Gröbner basis G of I with respect to the reverse lexicographic ordering to get the Hilbert function of R/I. Then one computes the desired Gröbner basis with respect to the lexicographic ordering, making use of the Hilbert function to avoid unneeded computations.

The other approach is basis conversion. One first computes a Gröbner basis with respect to the reverse lexicographic ordering and then converts it into a Gröbner basis with respect to the lexicographic ordering. There are two different methods.

The first method was introduced in 1993 by Faugère et al. It is called *FGLM algorithm* and works only if $K[x_1, \ldots, x_n]/I$ has finite dimension as a K-vector space, that is, if I is zero-dimensional. If we have a Gröbner basis for I with respect to the reverse lexicographic ordering, then we can use the structure of the K-vector space $K[x_1, \ldots, x_n]/I$ to obtain systems of linear equations from which elements of a Gröbner basis with respect to the lexicographic ordering can be found.

In 1997 Collart et al. introduced a new algorithm which makes use of the Gröbner fan of an ideal. A Gröbner basis with respect to the lexicographic ordering will be approximated step by step from a Gröbner basis with respect to the reverse lexicographic ordering. In each step, a Gröbner basis F of I with respect to a new ω-weighted monomial ordering $\succ_{(\omega,\, lex)}$ will be computed from the Gröbner basis computed in the step before. The weight vector ω is determined by the way which we follow while walking through the Gröbner fan. Thus, the cost of the total computation depends on this way. Suggestions on how to perform the *Gröbner walk* are due to Amrhein et al. (1996, 1997, 1998) and Tran (2000).

So far, the Gröbner walk algorithm and its variations were not available in the leading computer algebra systems for applications in algebraic geometry. As part of this dissertation, we implemented the algorithms of Amrhein et al. and Tran as part of the system SINGULAR which has been developed at the University of Kaiserslautern. We also came up with some alternative ideas of choosing the way through the Gröbner fan, thus reacting to some problems arizing from the way of representing polynomials in SINGULAR. We analyzed the time and memory usage of all algoritms implemented in SINGULAR in numerous examples.

This thesis is divided into five chapters. Chapter 1 gives a short introduction to algebra and geometry: polynomials, ideals, monomial orderings, the division algorithm, Gröbner bases and Gröbner fans of an ideal.

In Chapter 2 we present not only the algorithms of Faugère et al. (1993) and Traverso (1997), but also variations of Buchberger's algorithm. These variations are based on the criteria of Buchberger (1985), Gebauer and Möller (1988) and Giovini et al. (1991) respectively.

The Gröbner walk algorithm is analyzed in Chapter 3. This algorithm uses several times a „direct“ basis conversion and correspondingly different monomial orderings, depending on the current weight vector. Therefore, we present first this direct basis conversion. Then, we show how to find a weight vector for constructing a monomial ordering needed in the next conversion step.

The algorithms of Amrhein et al. (1996, 1997 and 1998) and Tran (2000) are presented in Chapter 4. We also describe our own algorithms in Chapter 4. The advantages and disadvantages of these algorithms are illustrated by some examples.

The last chapter compares the presented algorithms with respect to their practical al performance in a large number of examples. Roughly speaking, our test results are as follows:

If I is a nonhomogeneous ideal, given by its reduced Gröbner basis with respect to the reverse lexicographic ordering, typically our algorithms are the fastest.

If I is a homogeneous ideal, however, the Hilbert-driven version of Buchberger's algorithm performs best.

Einleitung

In vielen Anwendungsgebieten der Mathematik lassen sich Probleme durch lineare Gleichungssysteme beschreiben. Die Struktur der Lösungsmengen solcher Systeme hat man gut verstanden, und es existieren effiziente Algorithmen zur Bestimmung ihrer Lösung, wie etwa der Gauß'sche Algorithmus.

Andere Probleme lassen sich nicht durch lineare, aber durch polynomiale Gleichungssysteme beschreiben (zum Beispiel in der Robotik).

Im Gegensatz zu linearen Gleichungssystemen sind polynomiale Gleichungssysteme viel schwieriger zu lösen. Die Algebraische Geometrie stellt aber eine Vielzahl von Techniken zum qualitativen und quantitativen Studium der Lösungsmengen solcher Systeme bereit. Vor allem in den letzten Jahrzehnten gab es bedeutende Entwicklungen auf dem Gebiet der Computeralgebra (und der numerischen Ananlysis). Schnelle Computer und neu entwickelte Algorithmen, insbesondere solche um Lösungen polynomialer Gleichungssysteme zu finden, ermöglichen es heutzutage, schwierige praktische Probleme zu lösen und theoretische Fragestellungen zu untersuchen.

Lösungen polynomialer Gleichungssysteme zu bestimmen hängt eng mit geometrischen Techniken (Projektion, Fortsetzung von Lösungen) zusammen. Die Algebraische Geometrie stellt einen Zusammenhang zwischen Geometrie und Algebra her: Eigenschaften geometrischer Objekte werden mit denen von Polynomen und den von ihnen erzeugten Idealen in Verbindung gebracht. In diesem Fall spielen zum Beispiel Erzeugendensysteme eines Ideals mit besonders „guten“ Eigenschaften, sogenannte *Gröbnerbasen*, eine wichtige Rolle.

Gröbnerbasen und der Buchbergeralgorithmus zur Berechnung von Gröbnerbasen sind ein unverzichtbarer Bestandteil der modernen Algebra und wurden in vielen bedeutenden Computeralgebrasystemen implementiert. Mit ihrer Hilfe lassen sich auch abstrakte Fragenstellungen lösen.

Eine Gröbnerbasis eines Ideals ist nicht eindeutig bestimmt, sie hängt insbesondere von der Wahl einer monomialen Ordnung ab. Für die meisten Anwendungen spielt es keine Rolle, welche Gröbnerbasis benutzt wird. In diesem Fall wählt man aus Gründen der Effizienz typischerweise die umgekehrt lexikographische Ordnung zur Berechnung einer Gröbnerbasis. Für manche Anwendungen benötigt man aber Gröbnerbasen mit speziellen Eigenschaften und deswegen auch spezielle monomiale Ordnungen. Zum expliziten Lösen von polynomialen Gleichungssystemen, das heisst zum Eliminieren von Variablen, verwendet man etwa häufig die lexikographische Ordnung. Die Berechnung einer Gröbnerbasis bezüglich einer solchen Ordnung benötigt allerdings fast immer mehr Zeit und Speicher. Diese Schwierigkeit zu überwinden ist somit von großer Wichtigkeit. Im Wesentlichen gibt es hierfür zwei Ansätze:

Der erste Ansatz, der auf Traverso (1997) zurückgeht, benutzt die in der Hilbertfunktion eines Restklassenrings R/I enthaltene Information (*Hilbert-driven Buchbergeralgorithmus*). Zunächst berechnet man eine Gröbnerbasis des homogenen Ideals I im Polynomring R bezüglich der umgekehrt lexikographischen Ordnung $\succ_{rlex}$ und erhält hieraus die Hilbertfunktion von R/I. Dann berechnet man die gewünschte Gröbnerbasis bezüglich der lexikographischen Ordnung indem man die in der Hilbertfunktion enthaltene Information ausnutzt um überflüssige Rechnungen zu vermeiden.

Der andere Ansatz basiert auf Basiskonvertierung. Hier wird zuerst eine Gröbnerbasis bezüglich einer umgekehrt lexikographischen Ordnung ausgerechnet und anschließend in eine Gröbnerbasis bezüglich einer lexikographischen Ordnung konvertiert. Es gibt im Wesentlichen zwei verschiedene Methoden, diese Konvertierung durchzuführen.

Die zeitlich erste Methode wurde 1993 von Faugère et al. vorgestellt. Dieser sogenannte *FGLM-Algorithmus* nutzt aus, dass $K[x_1, \ldots, x_n]/I$ ein endlichdimensionaler K-Vektorraum ist, wenn I ein nulldimensionales Ideal ist. Hat man eine Gröbnerbasis von I bezüglich der umgekehrt lexikographischen Ordnung, so kann man die K-Vektorraumstruktur von $K[x_1, \ldots, x_n]/I$ ausnutzen, um lineare Gleichungssysteme zu gewinnen, mit deren Hilfe die Elemente einer Gröbnerbasis bezüglich der lexikographischen Ordnung bestimmt werden können.

Collart et al. stellten 1997 eine andere Methode vor, welche die Eigenschaften des Gröbnerfans eines Ideals benutzt. Ausgehend von einer Gröbnerbasis bezüglich der umgekehrt lexikographischen Ordnung nähert man sich schrittweise einer Gröbnerbasis bezüglich der lexikographischen Ordnung. In jedem Schritt wird aus der im vorherigen Schritt berechneten Gröbnerbasis G eine neue Gröbnerbasis F von I bezüglich einer neuen ω-gewichteten monomialen Ordnung $\succ_{(\omega, lex)}$ berechnet. Der Gewichtsvektor ω wird durch den Gröbnerfan bestimmt. Wählt man solche Vektoren geschickt, so kann die neue Gröbnerbasis leicht berechnet werden. Der Rechenaufwand hängt entscheidend vom gewählten „Weg“ durch den Gröbnerfan ab, weswegen dieser Algorithmus *Gröbnerwalk* genannt wird. Die Frage nach der “besten” Wahl des Wegs ist ein aktuelles Forschungsthema in der Computeralgebra. Die bekanntesten Lösungsvorschläge stammen von Amrhein et al. (1996, 1997, 1998) und Tran (2000).

Die bisher bekannten Algorithmen zur Konvertierung wurden, abgesehen von FGLM, noch nicht im Computeralgebrasystem SINGULAR implementiert, weswegen wir dies im Rahmen dieser Dissertation vornahmen (SINGULAR wird an der Universität Kaiserslautern entwickelt und ist eines von zwei führenden Computeralgebrasystemen für die Bereiche Kommutative Algebra, Algebraische Geometrie und Singularitätentheorie). Die Dissertation stellt aber auch neue Ideen vor, mit denen insbesondere auf Probleme reagiert wird, die sich aus der Polynomdarstellung im Kern von SINGULAR ergeben. Den Zeit- und Speicherverbrauch aller implementierten Algorithmen untersuchten wir, indem wir zahlreiche Beispiele berechneten.

Diese Arbeit ist in fünf Kapitel aufgeteilt. In Kapitel 1 bieten wir einen Überblick über Grundlagen aus der Algebra und Geometrie. Es werden Begriffe wie Polynom, Ideal, monomiale Ordnung, Divisionsalgorithmus, Gröbnerbasis und Gröbnerfan eines Ideals eingeführt.

In Kapitel 2 werden nicht nur Varianten des Buchbergeralgorithmus, die auf den Kriterien von Buchberger (1985), Gebauer und Möller (1988) bzw. Giovini et al.

(1991) basieren, vorgestellt, sondern auch der Algorithmus von Faugère et al. (1993) bzw. Traverso (1996).

Der Gröbnerwalkalgorithmus wird in Kapitel 3 ausführlich analysiert. Im Gröbnerwalkalgorithmus wird eine „direkte“ Basiskonvertierung mehrfach genutzt und mehrere monomiale Ordnungen gebraucht (diese hängen von dem aktuell gewählten Gewichtsvektor ab). Deswegen wird die direkte Konvertierung zuerst erläutert. Danach stellen wir vor, wie man einen Gewichtsvektor zur Konstruktion einer monomialen Ordnung findet, der in der nächsten Konvertierung genutzt wird.

Die Algorithmen von Amrhein et al. (1996, 1997, 1998) und der Algorithmus von Tran (2000) werden in Kapitel 4 vorgestellt. Darüber hinaus werden eigene Algorithmen beschrieben. Außerdem werden Vor- und Nachteile der Algorithmen anhand einiger Beispiele erläutert.

Im letzten Kapitel werden alle vorgestellte Algorithmen zur Basiskonvertierung miteinander verglichen. Da eine theoretische Analyse der Algorithmen kaum möglich ist und wenig über die praktische Eignung des jeweiligen Algorithmus aussagt, wurden umfangreiche Tests mit in der Literatur vorkommenden Beispielen durchgeführt. Dabei stellte sich folgendes heraus:

Ist I ein inhomogenes Ideal, dessen Gröbnerbasis bezüglich der umgekehrt lexikographischen Ordnung gegeben ist, so kann man im Allgemeinen mit unseren Algorithmen am schnellsten eine Basiskonvertierung durchführen.

Ist I hingegen ein homogenes Ideal, so ist der Hilbert-driven Buchbergeralgorithmus immer schneller als alle auf dem Gröbnerwalk basierenden Algorithmen.

Danksagung

Für die Erstellung dieser Arbeit danke ich Gott!

Während meiner Arbeit habe ich hilfreiche Anregungen von mehreren Leuten bekommen:

Prof. Dr. Wolfram Decker hat mir das Thema dieser Arbeit gegeben und war auch immer bereit, darüber zu diskutieren und mir zu helfen. Deswegen bedanke ich mich sehr bei ihm, auch für seine Geduld.

Dank gebührt Olaf Bachmann und Hans Schönemann für ihre hilfreichen Vorschläge und für die Anbindung an SINGULAR. Mein herzlicher Dank geht an Holger Cröni und Carsten Kühn für ihre zahlreichen Kommentare zur sprachlichen Verbesserung dieser Arbeit. Danken möchte ich ebenfalls Christine Theis und Emanuel Herrmann für ihre Vorschläge.

Mein Dank geht auch an das PGSM-Projekt (Indonesien) für die finanzielle Unterstützung, an den DAAD (Deutschland) für die Organisation meines Aufenthaltes und an meine Kollegen, die an der staatlichen Universität von Malang (Indonesien) arbeiten und meine beruflichen Verpflichtungen übernommen haben.

Schließlich bedanke ich mich sehr bei meiner geliebten Frau für die moralische und praktische Unterstützung, vor allem bei der Erziehung unserer Kinder.

Für meine geliebte Familie:
Rachwati,
Ni Putu Adi Nirmala Putri,
I Kadek Adi Satya Putra,
I Komang Adi Satrya Putra.

Inhaltsverzeichnis

Kapitel 1

Grundlagen aus der Algebra und Geometrie

In diesem Kapitel werden wir einen Überblick über die Grundlagen aus der Algebra und der Geometrie geben, die wichtig für diese Arbeit sind: Wir werden Polynome, Ideale, monomiale Ordnungen, den Divisionsalgorithmus, Gröbnerbasen und Gröbnerfans betrachten.

In dieser Arbeit bezeichnen wir mit $\mathbb{Z}$ die Menge der ganzen Zahlen, mit $\mathbb{N}_0$ die Menge der nichtnegativen ganzen Zahlen, mit $\mathbb{R}$ die Menge der reellen Zahlen, mit $\mathbb{R}_{\geq 0}$ die Menge der nichtnegativen reellen Zahlen, und mit $\mathbb{Q}_{\geq 0}$ die Menge der nichtnegativen rationalen Zahlen. Falls nicht anders erwähnt, bezeichnen wir mit K einen beliebigen Körper.

1.1 Polynome und Ideale

In diesem Abschnitt führen wir einige Bezeichnungen ein und wiederholen den Hilbertschen Basissatz, der besagt: jedes Ideal im Polynomring $K[x_1, \ldots, x_n]$ mit n Variablen x_i über dem Körper K ist von endlich vielen Polynomen erzeugt.

Bemerkung und Definition 1.1.1. *Ist $\alpha := (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$, so nennt man das Potenzprodukt*

$$\mathbf{x}^\alpha := x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$$

*ein **Monom** in den Variablen $x_1, \ldots, x_n$ vom **Grad***

$$|\alpha| := \alpha_1 + \cdots + \alpha_n.$$

Jeder Ausdruck der Form

$$a_\alpha \mathbf{x}^\alpha, a_\alpha \in K \backslash \{0\}$$

*heißt **Term** vom Grad $|\alpha|$. Jedes Polynom $f \in K[x_1, \ldots, x_n] \backslash \{0\}$ läßt sich als Summe endlich vieler Terme schreiben. Haben alle diese Terme den gleichen Grad d, so*

heißt f ***homogen*** *(vom Grad d). Fassen wir in der Darstellung eines beliebigen Polynoms $f \in K[x_1, \ldots, x_n]\backslash\{0\}$ die Terme gleichen Totalgrads zusammen, so erhalten wir eine eindeutige Zerlegung*

$$f = f_{i_1} + f_{i_2} + \cdots + f_{i_t}$$

von f in homogene Komponenten $f_{i_s} \neq 0$ vom Grad $i_1 > i_2 > \cdots > i_t$. Dann heißt i_1 auch der ***Totalgrad*** $\deg(f)$ *von f.*

Bemerkung und Definition 1.1.2. *Jeder Vektor $\omega := (\omega_1, \ldots, \omega_n)$ aus*

$$\mathbb{R}^n_{\geq 0} := \{(\alpha_1, \ldots, \alpha_n) : \alpha_i \in \mathbb{R}_{\geq 0},\ 0 \leq i \leq n\}$$

heißt ***Gewichtsvektor*** *und der ω-**gewichtete Grad** des Monoms $\mathbf{x}^\alpha$ ist definiert als*

$$\deg_\omega(\mathbf{x}^\alpha) := \omega \cdot \alpha := \omega_1\alpha_1 + \cdots + \omega_n\alpha_n.$$

*Haben alle Terme eines Polynoms $f \in K[x_1, \ldots, x_n]\backslash\{0\}$ den gleichen ω-gewichteten Grad, so heißt f ω-**homogen**. Ist f beliebig, so heißt der maximale ω-homogene Grad der Terme von f auch ω-**gewichteter Totalgrad** $\deg_\omega(f)$ von f. Fassen wir in der Darstellung von f die Terme gleichen ω-gewichteten Totalgrads zusammen, so erhalten wir eine eindeutige Zerlegung*

$$f = f_{i_1} + f_{i_2} + \cdots + f_{i_t}$$

von f in ω-homogene Komponenten $f_{i_s} \neq 0$ vom ω-gewichteten Grad $i_1 > i_2 > \cdots > i_t$. Dann heißt f_{i_1} auch ***Initialform*** $in_\omega(f)$ *von f bzgl. ω.*

Für zwei Monome $\mathbf{x}^\alpha, \mathbf{x}^\beta$ hat man

$$\deg_\omega(\mathbf{x}^\alpha\mathbf{x}^\beta) = \deg_\omega(\mathbf{x}^\alpha) + \deg_\omega(\mathbf{x}^\beta).$$

Definition 1.1.3. *Eine nicht leere Teilmenge I von $K[x_1, \ldots, x_n]$ heißt* ***Ideal****, falls gilt:*

(i) Sind $f, g \in I$, so ist $f + g \in I$.

(ii) Ist $f \in I$ und $h \in K[x_1, \ldots, x_n]$, so ist $fh \in I$.

Definition 1.1.4. *Sei $I \subset K[x_1, \ldots, x_n]$ ein Ideal.*

(i) I heißt ***homogen****, falls für alle $0 \neq f \in I$ seine homogenen Komponenten in I sind.*

*(ii) Sei $\omega \in \mathbb{R}^n_{\geq 0}$ ein Gewichtsvektor. I heißt ω-**homogen** , falls für alle $0 \neq f \in I$ seine ω-homogenen Komponenten in I sind.*

Definition 1.1.5. *Sei R ein kommutativer Ring mit 1. Dann heißt R* ***noethersch****, wenn jedes Ideal I in R endlich erzeugt ist, d.h. es gibt $a_1, \ldots, a_s \in I$ mit*

$$I = \{b_1a_1 + \cdots + b_sa_s : b_i \in R, 0 \leq i \leq s\} =: \langle a_1, \ldots a_s\rangle.$$

Die Elemente $a_1, \ldots, a_s$ heißen ***Erzeuger*** *von I und die Menge dieser Elemente heißt* ***Erzeugendensystem*** *von I.*

Beispiel 1.1.6. *(i) Da jeder Körper K nur zwei Ideale, nämlich $\{0\}$ und K hat, ist K noethersch.*

(ii) Der Ring $\mathbb{Z}$ ist noethersch, da $\mathbb{Z}$ sogar ein Hauptidealring ist: Jedes Ideal $I \subset \mathbb{Z}$ hat die Gestalt $I = \langle \nu \rangle$ für ein geeignetes $\nu \in \mathbb{N}_0$.

Satz 1.1.7. *Für einen kommutativen Ring R mit 1 sind äquivalent:*

(i) R ist noethersch.

(ii) Jede aufsteigende Kette von Idealen

$$I_1 \subset I_2 \subset \cdots \subset I_s \subset \cdots \subset R$$

*wird **stationär**, d.h. es gibt ein $s_0 \in \mathbb{N}_0$ mit $I_s = I_{s_0}, \forall s \geq s_0$.*

Beweis. Siehe Gröbner (1968). □

Satz 1.1.8. ***(Hilbertscher Basissatz)*** *Sei R noethersch. Dann ist auch der Polynomring $R[x]$ in einer Variablen x noethersch.*

Beweis. Siehe Gröbner (1968). □

Da K noethersch ist, ist auch $K[x]$ noethersch.

Wegen $K[x_1, \ldots, x_n] = K[x_1, \ldots, x_{n-1}][x_n] = \cdots = K[x_1] \cdots [x_{n-1}]\ [x_n]$ erhalten wir aus dem Hilbertschen Basissatz das folgende Korollar.

Korollar 1.1.9. *Der Polynomring $K[x_1, \ldots, x_n]$ ist noethersch* [1].

Nach diesem Korollar ist jedes Ideal I in $K[x_1, \ldots, x_n]$ endlich erzeugt, d.h. von der Form

$$I = \langle f_1, \ldots, f_s \rangle \text{ mit } f_1, \ldots, f_s \in K[x_1, \ldots, x_n].$$

Jedes Element von I läßt sich als Linearkombination

$$h_1 f_1 + \cdots + h_s f_s \text{ mit } h_1, \ldots, h_s \in K[x_1, \ldots, x_n]$$

schreiben. Dies erinnert an die Situation eines Vektorraums über einem Körper (mit $K[x_1, \ldots, x_n]$ in der Rolle des Körpers und mit I in der Rolle des Vektorraums). Allerdings bildet $f_1, \ldots, f_s$ keine „Basis“ im Sinne der linearen Algebra, da es zwischen $f_1, \ldots, f_s$ nicht-triviale Relationen

$$h_1 f_1 + \cdots + h_s f_s = 0$$

gibt (z.B. $f_2 f_1 - f_1 f_2 = 0$).

Bemerkung und Definition 1.1.10. *Ein von Null verschiedenes Ideal $I \subset K[x_1, \ldots, x_n]$ heißt **monomiales Ideal**, wenn es ein Erzeugendensystem von I gibt, das aus Monomen besteht (dann gibt es auch endlich viele Monome, die I erzeugen). Sei also I ein monomiales Ideal, etwa*

$$I = \langle m_1, \ldots, m_s \rangle$$

mit Monomen m_i. Wird etwa m_j durch ein m_i mit $i \neq j$ geteilt, so gilt

$$I = \langle m_1, \ldots, \hat{m}_j, \ldots, m_s \rangle.$$

*Fahren wir so fort, so erhalten wir ein eindeutig bestimmtes **minimales Erzeugendensystem** [2] von I, das aus Monomen besteht.*

[1] Das Korollar 1.1.9 garantiert, dass der später vorgestellte „Buchbergeralgorithmus“ terminiert.

[2] Ist M das minimale Erzeugendensystem von I, so ist $M \subset N$ für alle monomiale Erzeugendensysteme N von I.

1.2 Monomiale Ordnungen

Im Polynomring $K[x]$ in einer Variablen x sind die Monome in $K[x]$ geordnet:

$$\cdots > x^i > \cdots > x^2 > x > 1.$$

Deswegen kann man ein Polynom $f \in K[x]$ durch ein von Null verschiedenes anderes Polynom $g \in K[x]$ dividieren. Division mit Rest funktioniert, in dem man zunächst den Leitterm von f, d.h. den Term höchster Ordnung von f, durch den Leitterm von g dividiert. Wir ersetzen dann f durch

$$f_1 := f - \frac{lc(f)}{lc(g)} x^{\deg(f)-\deg(g)} g$$

(falls $\deg(f) \geq \deg(g)$). Diese Vorgehensweise wird wiederholt, bis der Leitterm von f_1 durch den Leitterm von g nicht geteilt wird, d.h. $\deg(f_1) < \deg(g)$.

Diese Vorgehensweise wollen wir auf Polynome in mehreren Variablen verallgemeinern. Deswegen beschäftigen wir uns nun mit monomialen Ordnungen auf dem Polynomring $K[x_1, \ldots, x_n]$. Wir erinnern uns zunächst an den Begriff Ordnung.

Definition 1.2.1. *Sei* X *eine nicht leere Menge. Eine Relation* $\mathrm{T} \subset \mathrm{X} \times \mathrm{X}$ *heißt* ***Teilordnung*** *auf* X*, falls gilt:*

(i) $(x,x) \in \mathrm{T}, \forall x \in \mathrm{X}$ *(Reflexivität),*

(ii) $(x,y) \in \mathrm{T} \wedge (y,x) \in \mathrm{T} \Longrightarrow x = y$ *(Antisymmetrie),*

(iii) $(x,y) \in \mathrm{T} \wedge (y,z) \in \mathrm{T} \Longrightarrow (x,z) \in \mathrm{T}$ *(Transitivität).*

Man schreibt dann auch $x \succeq y$ *oder* $y \preceq x$*, falls* $(x,y) \in \mathrm{T}$*, und* $x \succ y$ *oder* $y \prec x$*, falls* $(x,y) \in \mathrm{T}$ *und* $x \neq y$*.*

Eine Teilordnung $\succ$ *auf* X *heißt* ***Totalordnung*** *auf* X*, falls sich je zwei Elemente von* X *bzgl.* $\succ$ *vergleichen lassen*

$$x, y \in \mathrm{X} \Longrightarrow x \succ y \text{ oder } x = y \text{ oder } y \succ x.$$

Sind $\succ_1$ *und* $\succ_2$ *Teilordnungen auf* X*, so heißt* $\succ_2$ ***feiner*** *als* $\succ_1$*, falls gilt*

$$x \succ_1 y \Longrightarrow x \succ_2 y.$$

Beispiel 1.2.2. *Auf* $\mathbb{N}_0^n$ *haben wir eine Teilordnung* $\preceq$*, die wie folgt definiert ist:*

$$\alpha \preceq \beta :\Longleftrightarrow \alpha_i \leq \beta_i, \forall 1 \leq i \leq n$$

für alle $\alpha := (\alpha_1, \ldots, \alpha_n),\ \beta := (\beta_1, \ldots, \beta_n) \in \mathbb{N}_0^n$*.*

Teilbarkeit definiert eine teilweise Ordnung auf der Menge aller Monome in $K[x_1, \ldots, x_n]$*. Diese entspricht der obigen Teilordnung* $\preceq$ *auf* $\mathbb{N}_0^n$*:*

$$\mathbf{x}^\alpha | \mathbf{x}^\beta \Longleftrightarrow \alpha \preceq \beta$$

Aber diese Teilordnungen sind keine Totalordnungen.

Definition 1.2.3. *Eine Totalordnung $\succ$ auf einer Menge X heißt* ***Wohlordnung****, wenn jede nicht leere Teilmenge von X ein kleinstes Element bzgl. $\succ$ hat.*

Definition 1.2.4. *Eine* ***monomiale Ordnung*** *oder* ***Termordnung*** *auf dem Polynomring $K[x_1, \ldots, x_n]$ ist eine Totalordnung $\succ$ auf der Menge aller Monome in $K[x_1, \ldots, x_n]$ mit den folgenden Eigenschaften:*

(i) $\succ$ verfeinert die durch Teilbarkeit definierte Teilordnung, d.h. für je zwei Monome $m_1 \neq m_2$ in $K[x_1, \ldots, x_n]$ gilt:

$$m_2 \text{ teilbar durch } m_1 \Longrightarrow m_2 \succ m_1,$$

(ii) $\succ$ ist multiplikativ, d.h. für je drei Monome m_1, m_2, n gilt:

$$m_1 \succ m_2 \Longrightarrow nm_1 \succ nm_2.$$

Lemma 1.2.5. *Jede monomiale Ordnung $\succ$ auf $K[x_1, \ldots, x_n]$ ist eine Wohlordnung.*

Beweis. Sei M eine beliebige nicht leere Teilmenge der Menge aller Monome in $K[x_1, \ldots, x_n]$. Wir betrachten das monomiale Ideal $\langle M \rangle$, das durch M erzeugt wird. Da $K[x_1, \ldots, x_n]$ noethersch ist, existieren endlich viele Monome in M, etwa $m_1, \ldots, m_s$, so dass $\langle M \rangle = \langle m_1, \ldots, m_s \rangle$ ist.

Dann ist jedes Monom von $\langle M \rangle$ durch eines der m_i teilbar. Also ist das kleinste Monom von $\{m_1, \ldots, m_s\}$ auch das kleinste Monom in M. □

Sei M_R die Menge aller Monome in $R := K[x_1, \ldots, x_n]$. Durch die Abbildung

$$exp : M_R \longrightarrow \mathbb{N}_0^n \text{ mit } exp(\mathbf{x}^\alpha) := \alpha,$$

die ein Isomorphismus zwischen den zwei Monoiden $(M_R, \cdot)$ und $(\mathbb{N}_0^n, +)$ ist, kann eine monomiale Ordnung $\succ$ auf $K[x_1, \ldots, x_n]$ als eine Ordnung auf $\mathbb{N}_0^n$ aufgefaßt werden: Für $\alpha, \beta \in \mathbb{N}_0^n$ setze

$$\alpha \succ \beta :\Longleftrightarrow \mathbf{x}^\alpha \succ \mathbf{x}^\beta.$$

Im folgenden wird zwischen diesen beiden Interpretationen nicht unterschieden.

Beispiel 1.2.6. *Die folgenden Festlegungen definieren monomiale Ordnungen auf $K[x_1, \ldots, x_n]$:*

(i) ***Lexikographische Ordnung*** *($\succ_{lex}$) :*

$$\alpha \succ_{lex} \beta \;:\Longleftrightarrow\; \exists\, i \in \{1, \ldots, n\} \text{ mit } \alpha_i > \beta_i \text{ und } \alpha_j = \beta_j, \forall j < i$$

(ii) ***Graduierte lexikographische Ordnung*** *($\succ_{glex}$) :*

$$\begin{aligned} \alpha \succ_{glex} \beta \;:\Longleftrightarrow\; & |\alpha| > |\beta| \text{ oder} \\ & \big(|\alpha| = |\beta| \text{ und } \alpha \succ_{lex} \beta\big) \end{aligned}$$

(iii) ***Umgekehrte (graduierte) lexikographische Ordnung*** *oder* ***reverslexikographische Ordnung*** *($\succ_{rlex}$) :*

$$\begin{aligned} \alpha \succ_{rlex} \beta \;:\Longleftrightarrow\; & |\alpha| > |\beta| \text{ oder} \\ & \big(|\alpha| = |\beta| \text{ und } \exists\, i \in \{1, \ldots, n\} \text{ mit } \alpha_i < \beta_i \\ & \quad \text{und } \alpha_j = \beta_j, \forall j > i\big) \end{aligned}$$

(iv) Sei $\omega \in \mathbb{R}^n_{\geq 0}$. ***Die ω-gewichtete lexikographische Ordnung:***

$$\alpha \succ_{(\omega,\ lex)} \beta \ :\Longleftrightarrow\ \omega \cdot \alpha > \omega \cdot \beta \text{ oder } \big(\omega \cdot \alpha = \omega \cdot \beta \text{ und } \alpha \succ_{lex} \beta\big).$$

Beispiel 1.2.7. *Sei A eine $(n \times n)$-Matrix mit reellen Einträgen. Für jeden Multiindex $\alpha \in \mathbb{N}_0^n$ ist dann $A \cdot \alpha$ definiert, wenn man α als Spaltenvektor auffaßt. $A \cdot \alpha$ ist ein Vektor in $\mathbb{R}^n$. Für die Vektoren des $\mathbb{R}^n$ ist $\succ_{lex}$ analog zu oben erklärt. Wir definieren für $\alpha, \beta \in \mathbb{N}_0^n$*

$$\alpha \succ_A \beta :\Longleftrightarrow A \cdot \alpha \succ_{lex} A \cdot \beta.$$

Setzen wir voraus, dass gilt

(i) A ist invertierbar.

(ii) In jeder Spalte von A ist der erste von 0 verschiedene Eintrag positiv,

so definiert $\succ_A$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$ (die durch A definierte monomiale Ordnung).

Satz 1.2.8. *Sei $\succ$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$. Dann existiert eine Matrix A wie im Beispiel 1.2.7, so dass die Ordnung $\succ_A$ gleich $\succ$ ist.*

Beweis. Siehe Robbiano (1985). □

Beispiel 1.2.9. *(i) Die lexikographische Ordnung auf $K[x, y, z]$ wird durch die 3×3-Einheitsmatrix $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ definiert.*

(ii) Sei $\omega = (1, 2, 3)$. Die ω-gewichtete lexikographische Ordnung $\succ_{(\omega,\ lex)}$ wird durch $A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ definiert.

(iii) Allgemein gilt: Ist $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n_{>0}$, so ist die durch

$$A = \begin{pmatrix} \omega_1 & \omega_2 & \cdots & \omega_{n-1} & \omega_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & & \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}_{n \times n}$$

definierte monomiale Ordnung $\succ_A$ auf $K[x_1, \ldots, x_n]$ auch die ω-gewichtete lexikographische Ordnung auf $K[x_1, \ldots, x_n]$.

Definition 1.2.10. *Sei $\succ$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$. Ist $0 \neq f = \sum_{\alpha \in \mathbb{N}_0^n} a_\alpha \mathbf{x}^\alpha \in K[x_1, \ldots, x_n]$, so heißt*

*(i) $exp_\succ(f) := max_\succ\{\alpha : a_\alpha \neq 0\}$ der **Leitexponent** von f,*

*(ii) $lc_\succ(f) := a_{exp_\succ(f)}$ der **Leitkoeffizient** von f,*

(iii) $lm_\succ(f) := \mathbf{x}^{exp_\succ(f)}$ *das* ***Initialmonom (Leitmonom)*** *von* f,

(iv) $in_\succ(f) := lc_\succ(f) lm_\succ(f)$ *der* ***Initialterm (Leitterm)*** *von* f.

Wir definieren $in_\succ(0) = 0$. *Ist* $I \subset K[x_1, \ldots, x_n]$ *ein Ideal und* $\omega \in \mathbb{R}^n_{\geq 0}$ *ein Gewichtsvektor, so heißt*

$$exp_\succ(I) := \{exp_\succ(f) : f \in I\}$$

die ***Menge der Exponenten*** *von* I,

$$in_\succ(I) := \langle \{in_\succ(f) : f \in I\} \rangle$$

das ***Initialideal*** *von* I *bzgl.* $\succ$ *und*

$$in_\omega(I) := \langle \{in_\omega(f) : f \in I\} \rangle$$

das ***Initialformenideal*** *von* I *bzgl.* ω.

Bemerkung 1.2.11. *In obiger Situation gilt:*

(i) Das Initialideal $in_\succ(I)$ *von* I *ist tatsächlich ein monomiales Ideal, aber das Initialformenideal* $in_\omega(I)$ *von* I *ist nicht notwendigerweise ein monomiales Ideal.*

(ii) $in_\omega(I)$ *ist ein monomiales Ideal genau dann, wenn* ω *im Innern eines Gröbnerkegels von* I *bzgl. einer monomialen Ordnung liegt (siehe Kapitel 3).*

(iii) Für jede monomiale Ordnung $\succ$ *und jedes Ideal* I *existiert ein nichtnegativer ganzer Gewichtsvektor* $\omega \in \mathbb{N}^n_0$, *so dass* $in_\omega(I) = in_\succ(I)$ *ist (siehe Sturmfels (1996)).*

(iv) Seien $f, g \in K[x_1, \ldots, x_n] \backslash \{0\}$ *und sei* m *der eindeutig bestimmte Term von* g, *so dass* $m \cdot in_\succ(f)$ *maximal ist. Dann gilt*

$$in_\succ(g \cdot f) = m \cdot in_\succ(f).$$

Denn: Ist $m_f \neq in_\succ(f)$ *ein Term von* f *und* $m_g \neq m$ *ein Term von* g, *so gilt*

$$m \cdot in_\succ(f) \succ m_g \cdot in_\succ(f) \succ m_g \cdot m_f.$$

1.3 Divisionsalgorithmus

In diesem Abschnitt betrachten wir wieder den Polynomring $K[x_1, \ldots, x_n]$ über einem beliebigen Körper K und eine monomiale Ordnung $\succ$ auf einem solchen Ring. Wir beschäftigen uns mit einem Algorithmus zur Berechnung von Division mit Rest in $K[x_1, \ldots, x_n]$.

Bemerkung und Definition 1.3.1. *Seien* $f, g \in K[x_1, \ldots, x_n]$ *mit* $g \neq 0$. *Dann sagen wir* f ***reduziert*** *sich auf* $f_1 \in K[x_1, \ldots, x_n]$ *modulo* g *in einem Schritt und schreiben*

$$f \xrightarrow{g} f_1,$$

falls es einen Term m *von* f *gibt, der durch* $in_\succ(g)$ *teilbar ist, so dass*

$$f_1 = f - \frac{m}{in_\succ(g)} g$$

gilt. Wir nennen f_1 eine ***Reduktion*** *von f bzgl. g. Da $\succ$ eine Wohlordnung ist, gibt es eine endliche Folge von Reduktionen*

$$f \xrightarrow{g} f_1 \xrightarrow{g} f_2 \xrightarrow{g} \cdots \xrightarrow{g} f_s := r,$$

so dass entweder $r = 0$ gilt oder kein Term von r durch $in_\succ(g)$ teilbar ist. Wir nennen r auch einen ***Rest*** *von f bei Division durch g und schreiben*

$$f \xrightarrow{g}_+ r.$$

Beispiel 1.3.2. *Wir betrachten zwei Polynome in $\mathbb{Q}[x, y]$:*

$$f := 6x^2y - x + 4y^3 - 1,$$
$$g := 2xy + y^3.$$

Ist $\succ$ die lexikographische Ordnung auf $\mathbb{Q}[x, y]$ mit $x > y$, so gilt $f \xrightarrow{g} f_1$, wobei $f_1 := -3xy^3 - x + 4y^3 - 1$ ist; in diesem Fall ist $m := 6x^2y = in_\succ(f)$. Schließlich erhält man

$$f \xrightarrow{g} -3xy^3 - x + 4y^3 - 1 \xrightarrow{\boldsymbol{2xy}+y^3} -x + 3/2y^5 + 4y^3 - 1 =: r.$$

Kein Monom von r ist durch $2xy = in_\succ(g)$ teilbar. Also ist $-x + 3/2y^5 + 4y^3 - 1$ ein Rest von f bei Division durch g.

Definition 1.3.3. *Seien $f, f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ mit $f_i \neq 0$ $(1 \leq i \leq s)$. Wir sagen: f* ***reduziert*** *sich auf $r \in K[x_1, \ldots, x_n]$ modulo $f_1, \ldots, f_s$ und schreiben*

$$f \xrightarrow{f_1,\ldots,f_s}_+ r,$$

falls Indizes $i_1, \ldots, i_t \in \{1, \ldots, s\}$ und Polynome $h_1, \ldots, h_{t-1} \in K[x_1, \ldots, x_n]$ existieren, so dass gilt

$$f \xrightarrow{f_{i_1}} h_1 \xrightarrow{f_{i_2}} h_2 \xrightarrow{f_{i_3}} \cdots \xrightarrow{f_{i_{t-1}}} h_{t-1} \xrightarrow{f_{i_t}} r.$$

Ist entweder $r = 0$ oder kein Monom von r durch einen der Initialterme $in_\succ(f_i)$, $i = 1, \ldots, s$ teilbar, so nennen wir r ***reduziert*** *bzgl. $f_1, \ldots, f_s$. In diesem Fall schreiben wir*

$$\overline{f}^{f_1,\ldots,f_s} = r$$

und nennen r auch einen ***Rest*** *von f bei Division durch $f_1, \ldots, f_s$.*

In der Tat liefert der Reduktionsprozess einen **Divisionsalgorithmus**, der den Euklidischen Divisionsalgorithmus für Polynome in einer Variablen verallgemeinert. In Algorithmus 1.3.4 beschreiben wir eine Variante dieses Algorithmus, bei der die Reihenfolge in der $f_1, \ldots, f_s$ angeordnet sind eine Rolle spielt.

Algorithmus 1.3.4. ***DivAlg***$(f, f_1, \ldots, f_s)$;

Eingabe: $f, f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ *mit* $f_i \neq 0$, $(1 \leq i \leq s)$.

Ausgabe: $u_1, \ldots, u_s, r \in K[x_1, \ldots, x_n]$ *wie in Satz 1.3.5.*

Setze: $u_1 = \cdots = u_s = r := 0$, $h := f$;

while $(h \neq 0)$ ***do***

if *existiert i, so dass* $in_\succ(h)$ *durch* $in_\succ(f_i)$ *teilbar ist* ***then***

wähle die kleinste Zahl i, so dass $in_\succ(h)$ *durch* $in_\succ(f_i)$ *teilbar ist,*

$u_i := u_i + \frac{in_\succ(h)}{in_\succ(f_i)}$;

$h := h - \frac{in_\succ(h)}{in_\succ(f_i)} f_i$;

else

$r := r + in_\succ(h)$;

$h := h - in_\succ(h)$;

return$(u_1, \ldots, u_s, r)$.

Algorithmus 1.3.4: *Divisionsalgorithmus.*

Satz 1.3.5. *Sind* $f, f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ *mit* $f_i \neq 0$ $(1 \leq i \leq s)$, *so liefert der Divisionsalgorithmus* $u_1, \ldots, u_s, r \in K[x_1, \ldots, x_n]$, *so dass gilt*

$$f = u_1 f_1 + \cdots + u_s f_s + r,$$

wobei r bzgl. $\{f_1, \ldots, f_s\}$ *reduziert ist, und*

$$in_\succ(f) = max\Big(max_{1 \leq i \leq s}\big(in_\succ(u_i) in_\succ(f_i)\big),\, in_\succ(r)\Big).$$

Beweis. Siehe Adams & Loustaunau (1994). □

Beispiel 1.3.6. *Wir betrachten* $R := \mathbb{R}[x, y]$ *mit* $\succ_{lex}$,

$$f = xy^2 - x \in R$$

sowie

$$f_1 = xy + 1,\ f_2 = y^2 - 1 \in R.$$

(i) Wir dividieren zunächst f durch f_1, f_2 *(die Reihenfolge spielt eine Rolle bei dem Divisionsalgorithmus).*

Setze: $u_1 := 0, u_2 := 0, r := 0, h := xy^2 - x$.

Am Ende der ersten Schleife hat man

$$u_1 := u_1 + \frac{in_{\succ_{lex}}(h)}{in_{\succ_{lex}}(f_1)} = y$$

$h := h - \frac{in_{\succ_{lex}}(h)}{in_{\succ_{lex}}(f_1)} f_1 = xy^2 - x - \frac{xy^2}{xy}(xy+1) = -x - y.$

Nach zwei weiteren Schleifendurchläufen hat man

$r := r + in_{\succ_{lex}}(h) = -x - y$

$h := h - in_{\succ_{lex}}(h) = 0.$

Wegen $h = 0$ *verlassen wir die Schleife und sehen*

$u_1 = y, u_2 = 0, r = -x - y,$

d.h. $f = y \cdot f_1 + 0 \cdot f_2 + (-x - y).$

(ii) Wir vertauschen f_1 *und* f_2*:*

$$g_1 := f_2,\ g_2 := f_1.$$

Setze: $u_1 := 0, u_2 := 0, r := 0, h := xy^2 - x.$

Am Ende der ersten Schleife hat man

$u_1 := u_1 + \frac{in_{\succ_{lex}}(h)}{in_{\succ_{lex}}(g_1)} = x$

$h := h - \frac{in_{\succ_{lex}}(h)}{in_{\succ_{lex}}(g_1)} g_1 = xy^2 - x - \frac{xy^2}{y^2}(y^2 - 1) = 0.$

Wegen $h = 0$ *verlassen wir die Schleife und sehen*

$u_1 = x, u_2 = 0, r = 0,$

d.h. $f = x \cdot g_1 + 0 \cdot g_2 + 0.$

Dieses Beispiel zeigt, dass der Rest bei Division im Polynomring mit mehreren Unbestimmten im Allgemeinen *nicht* eindeutig ist, d.h. der Rest von f bei Division durch $f_1, \ldots, f_s$ hängt also im Allgemeinen von der Reihenfolge der f_i ab. Diese Reihenfolge spielt keine Rolle mehr, wenn die Menge $\{f_1, \ldots, f_s\}$ eine Gröbnerbasis ist (siehe Korollar 1.4.7).

1.4 Gröbnerbasen

In diesem Abschnitt betrachten wir wieder den Polynomring

$$R := K[x_1, \ldots, x_n]$$

über einem beliebigen Körper K. Wir beschäftigen uns mit algebraischen Fragen im Zusammenhang mit dem Studium von Idealen in R. Nach dem Hilbertschen Basissatz ist jedes Ideal $I \subset R$ endlich erzeugt, d.h. von der Form

$$I = \langle f_1, \ldots, f_s \rangle, \text{ mit } f_1, \ldots, f_s \in R.$$

Im folgenden seien $\emptyset \neq I \subset K[x_1, \ldots, x_n]$ ein Ideal und $\succ$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$.

Definition 1.4.1. *Eine endliche Menge* $\{f_1, \ldots, f_s\} \subset I \backslash \{0\}$ *heißt* ***Gröbnerbasis*** *von* I *bzgl.* $\succ$*, falls gilt*

$$in_{\succ}(I) = \langle in_{\succ}(f_1), \ldots, in_{\succ}(f_s) \rangle.$$

Bemerkung 1.4.2. *Ist $F := \{f_1, \ldots, f_s\}$ eine Gröbnerbasis von I bzgl. $\succ_1$, so nicht notwendigerweise auch bzgl. $\succ_2$. Aber F ist auch eine Gröbnerbasis von I bzgl. $\succ_2$, wenn alle Initialterme der Elemente von F bzgl. der beiden monomialen Ordnungen gleich sind, wie das Lemma 1.4.8 zeigen wird.*

Das folgende Beispiel zeigt, dass Gröbnerbasen eines Ideals bzgl. einer monomialen Ordnung *nicht* eindeutig sind und ein Erzeugendensystem von I *nicht* immer eine Gröbnerbasis von I ist.

Beispiel 1.4.3. *Sei*

$$I := \langle x^2 + y, y \rangle = \langle x^2, y \rangle \subset \mathbb{R}[x, y].$$

$F := \{x^2 + y, y\}$ und $G := \{x^2, y\}$ sind zwei Gröbnerbasen von I bzgl. der lexikographischen Ordnung $\succ_1$ mit $x > y$. Die Menge G ist auch eine Gröbnerbasis von I bzgl. der lexikographischen Ordnung $\succ_2$ mit $y > x$. Aber F ist keine Gröbnerbasis von I bzgl. $\succ_2$, da

$$\langle in_{\succ_2}(x^2 + y), in_{\succ_2}(y) \rangle = \langle y \rangle \neq \langle x^2, y \rangle = in_{\succ_2}(I)$$

ist.

Lemma 1.4.4. *Ein von Null verschiedenes Ideal I besitzt eine Gröbnerbasis bzgl. $\succ$.*

Beweis. Siehe Cox et al. (1997). □

Lemma 1.4.5. *Seien I und J zwei Ideale in $K[x_1, \ldots, x_n]$. Ist $J \subset I$ mit $in_\succ(J) = in_\succ(I)$, so ist $J = I$.*

Beweis. **Annahme:** $J \neq I$. Nach Lemma 1.2.5 können wir dann $f \in I \backslash J$ wählen, so dass $in_\succ(f)$ minimal ist unter allen $in_\succ(g)$, $g \in I \backslash J$. Da

$$in_\succ(f) \in in_\succ(I) = in_\succ(J)$$

ist, so existiert $h \in J$ mit $in_\succ(f) = m \cdot in_\succ(h)$ für einen Term m. Dann ist $g := f - m \cdot h \in I \backslash J$ mit $in_\succ(f) \succ in_\succ(g)$. Dies ist ein Widerspruch zur Minimalität von $in_\succ(f)$. □

Im folgenden Satz zeigen wir, dass Gröbnerbasen tatsächlich die angekündigte Eigenschaft bzgl. Division mit Rest haben.

Satz 1.4.6. *Für $\{f_1, \ldots, f_s\} \subset I$ sind äquivalent:*

(i) $\{f_1, \ldots, f_s\}$ ist eine Gröbnerbasis von I.

(ii) Für alle $f \in K[x_1, \ldots, x_n]$ gilt $f \in I$ genau dann wenn $f \xrightarrow{f_1, \ldots, f_s}_+ 0$.

Beweis. Siehe Adams & Loustaunau (1994). □

Aus dem Satz 1.4.6 erhalten wir:

Korollar 1.4.7. *Sei $\{f_1, \ldots, f_s\}$ eine Gröbnerbasis von $I \subset K[x_1, \ldots, x_n]$ bzgl. $\succ$. Dann gilt:*

(i) $I = \langle f_1, \ldots, f_s \rangle$.

(ii) Der Rest von $f \in K[x_1, \ldots, x_n]$ bei Division durch $f_1, \ldots, f_s$ hängt nur von I und $\succ$, aber nicht von der Gröbnerbasis ab.

Korollar 1.4.7 (ii) zeigt, dass in unserem Divisionsalgorithmus die Reihenfolge von $f_1, \ldots, f_s$ keine Rolle spielt, wenn $\{f_1, \ldots, f_s\}$ eine Gröbnerbasis ist.

Lemma 1.4.8. *Seien $\succ$ und $\gg$ zwei verschiedene monomiale Ordnungen. Ist $F := \{f_1, \ldots, f_s\}$ eine Gröbnerbasis von I bzgl. $\gg$ mit $in_{\gg}(f_i) = in_{\succ}(f_i)$ für alle $i = 1, \ldots, s$, so ist F auch eine Gröbnerbasis bzgl. $\succ$.*

Beweis. Sei $f \in I$ beliebig. Division mit Rest bzgl. $\succ$ liefert eine Darstellung

$$f = a_1 f_1 + \cdots + a_s f_s + r, \tag{1.1}$$

wobei kein im Polynom r auftretendes Monom durch eines der

$$\{in_{\succ}(f_i) : 1 \leq i \leq s\} = \{in_{\gg}(f_i) : 1 \leq i \leq s\}$$

geteilt wird. Da F eine Gröbnerbasis von I bzgl. $\gg$ ist, so muss $r = 0$ sein. Aus dem Satz 1.3.5 folgt $in_{\succ}(f) \in \langle in_{\succ}(f_1), \ldots, in_{\succ}(f_s) \rangle$. Also ist F eine Gröbnerbasis von I bzgl. $\succ$. □

Beispiel 1.4.9. *Sei*

$$I := \langle x^2 - xy + z,\ -x^2 + y^2 + xz,\ xz + yz \rangle \subset \mathbb{R}[x, y, z].$$

Dann ist die Menge

$$G := \{z^2,\ xz + yz,\ xy - y^2 + yz - z,\ x^2 - y^2 + yz,\ 2y^2z - yz^2 + z^2\}$$

eine Gröbnerbasis von I bzgl. $\succ_{rlex}$ mit $x > y > z$. G ist auch eine Gröbnerbasis von I bzgl. $\succ_{lex}$ mit $x > y > z$, weil $z^2, xz, xy, x^2, 2y^2z$ die Initialterme der Elemente von G bzgl. $\succ_{lex}$ sind.

Definition 1.4.10. *Eine Gröbnerbasis $\{f_1, \ldots, f_s\}$ von I bzgl. $\succ$ heißt* ***minimal****, wenn gilt:*

(i) $lc_{\succ}(f_i) = 1$ *für alle* $i = 1, 2, \ldots, s$.

(ii) Für alle $i = 1, 2, \ldots, s$ gilt $in_{\succ}(f_i) \notin \langle in_{\succ}(f_j) : j = 1, \ldots, s,\ j \neq i \rangle$.

Da jedes Ideal eine Gröbnerbasis besitzt (siehe Lemma 1.4.4), zeigt man durch Reduktion jedes Polynoms f_i einer solchen Gröbnerbasis bzgl. der Menge $\{f_j : j = 1, \ldots, s,\ j \neq i\}$ das folgende Lemma.

Lemma 1.4.11. *Sei $\{0\} \neq I$ ein Ideal und $\succ$ eine monomiale Ordnung. Dann gilt:*

(i) I besitzt eine minimale Gröbnerbasis bzgl. $\succ$.

(ii) Sind $\{f_1, \ldots, f_s\}$ und $\{g_1, \ldots, g_t\}$ zwei minimale Gröbnerbasen, so ist

$$\{in_\succ(f_1), \ldots, in_\succ(f_s)\} = \{in_\succ(g_1), \ldots, in_\succ(g_t)\}.$$

Insbesondere haben je zwei minimale Gröbnerbasen dieselbe Anzahl von Elementen.

Definition 1.4.12. *Eine Gröbnerbasis $\{f_1, \ldots, f_s\}$ von I bzgl. $\succ$ heißt* ***reduziert****, falls gilt:*

(i) $lc(f_i) = 1$ *für alle* $i = 1, \ldots, s$.

(ii) Für alle $i = 1, \ldots, s$ *enthält* $\langle in_\succ(f_j) : j = 1, \ldots, s,\ j \neq i\rangle$ *kein Monom von* f_i, *d.h.* f_i *ist reduziert bzgl.* $\{f_j : j = 1, \ldots, s,\ j \neq i\}$.

Durch Division mit Rest und aus dem Lemma 1.4.11 folgt:

Korollar 1.4.13. *Ein von Null verschiedenes Ideal I besitzt eine eindeutig bestimmte reduzierte Gröbnerbasis bzgl. $\succ$.*

Beispiel 1.4.14. *Betrachte die folgenden Ideale in* $\mathbb{R}[x, y]$:

$$I_1 := \langle x^2 + y^2, x\rangle,\ I_2 := \langle x^2 + y^2, x, y^2\rangle,\ I_3 := \langle y^2 + x, x\rangle \text{ und } I_4 := \langle y^2, x\rangle.$$

Es ist klar, dass alle diese Ideale gleich sind, d.h. $I := I_1 = I_2 = I_3 = I_4$. $F_1 := \{x^2 + y^2, x\}$ *ist keine Gröbnerbasis von I bzgl.* $\succ_{glex}$, *weil* $y^2 \in I$ *ist, aber*

$$in_{\succ_{glex}}(y^2) = y^2 \notin \langle x\rangle = \langle x^2, x\rangle = \langle in_{\succ_{glex}}(F_1)\rangle.$$

Die Mengen

$$F_2 := \{x^2 + y^2, x, y^2\},\ F_3 := \{y^2 + x, x\} \text{ und } F_4 := \{y^2, x\}$$

sind Gröbnerbasen von I bzgl. $\succ_{glex}$. F_2 *ist keine minimale Gröbnerbasis, weil der Initialterm* x^2 *von* $x^2 + y^2$ *durch den Initialterm von* $x \in F_2$ *teilbar ist.* F_3 *ist keine reduzierte Gröbnerbasis, weil das Monom x von* $y^2 + x$ *durch den Initialterm von* $x \in F_3$ *teilbar ist.* F_4 *ist die reduzierte Gröbnerbasis.*

Zum Schluß dieses Abschnittes beschäftigen wir uns mit dem Buchbergeralgorithmus, um eine Gröbnerbasis eines Ideals zu berechnen.

Theoretisch könnten wir Satz 1.4.6 anwenden, um zu entscheiden, ob eine endliche Menge von Polynomen eine Gröbnerbasis eines Ideals ist. Aber in der Praxis können wir natürlich *nicht* alle Polynome aus dem Ideal I durch $f_1, \ldots, f_s$ dividieren, um die Entscheidung zu treffen, ob die gegebene endliche Menge $\{f_1, \ldots, f_s\} \subset I$ eine Gröbnerbasis von I bildet. Wir zeigen, dass man tatsächlich mit endlich vielen Testelementen aus I auskommt.

Sei $F := \{f_1, \ldots, f_s\} \subset K[x_1, \ldots, x_n]\backslash\{0\}$ und $\succ$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$. Wir setzen für $1 \leq i \neq j \leq s$

$$m_{ij} := \frac{in_\succ(f_i)}{\gcd(in_\succ(f_i), in_\succ(f_j))},$$
$$\mathrm{Spoly}(f_i, f_j) := m_{ji} f_i - m_{ij} f_j,$$

wobei gcd für den normierten größten gemeinsamen Teiler steht. Das Polynom $\mathrm{Spoly}(f_i, f_j)$ nennt man **S-Polynom** oder **S-Paar** von f_i und f_j.

Satz 1.4.15. ***(Buchberger-Kriterium)*** *Mit obiger Notation gilt: Für ein Ideal $I \subset K[x_1, \ldots, x_n]$ und Erzeuger $f_1, \ldots, f_s$ von I sind äquivalent:*

(i) $\{f_1, \ldots, f_s\}$ *ist eine Gröbnerbasis von* I.

(ii) $\mathrm{Spoly}(f_i, f_j) \xrightarrow{f_1,\ldots,f_s}_+ 0$ *für alle* $1 \leq i < j \leq s$.

Beweis. Siehe Adams & Loustaunau (1994). □

Auf diesem Satz basiert der von Bruno Buchberger entwickelte Algorithmus zur Berechnung einer Gröbnerbasis eines Ideals. Dieser wird **Buchbergeralgorithmus** genannt.

Wir besprechen nun den Buchbergeralgorithmus. Dabei setzen wir nur voraus, dass I durch endlich viele erzeugenden Polynome gegeben ist.

Sei $\emptyset \neq I := \langle f_1, \ldots, f_s \rangle \subset K[x_1, \ldots, x_n]$ ein Ideal mit $f_i \neq 0$ für alle $1 \leq i \leq s$ und $\succ$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$. Dann kann eine Gröbnerbasis von I bzgl. $\succ$ mit endlich vielen Schritten durch den Algorithmus 1.4.16 berechnet werden.

Beispiel 1.4.17. *Seien* $f_1 := x + z$, $f_2 := x^2 - y$ *und* $I := \langle f_1, f_2 \rangle \subset \mathbb{R}[x, y, z]$. *Durch den Buchbergeralgorithmus wollen wir eine Gröbnerbasis von* I *bzgl.* $\succ_{glex}$ *ausrechnen. Zuerst definieren wir*

$$G := \{f_1, f_2\} \ , \ m := 3 \text{ und } B := \{(f_1, f_2)\}.$$

Am Ende der ersten Schleife hat man

$$f_3 := \overline{x(x+z) - 1(x^2 - y)}^G = z^2 - y,$$
$$B := \{(f_1, f_3), (f_2, f_3)\},$$
$$G := \{f_1, f_2, f_3\},$$
$$m := 4.$$

Am Ende der zweiten Schleife hat man

$$B := \{(f_2, f_3)\},$$
$$f_4 := \overline{z^2(x+z) - x(z^2 - y)}^G = 0,$$
$$G := \{f_1, f_2, f_3\},$$
$$m := 4.$$

Am Ende der dritten Schleife hat man

$$B := \emptyset,$$
$$f_4 := \overline{z^2(x^2 - y) - x^2(z^2 - y)}^G = 0,$$
$$G := \{f_1, f_2, f_3\},$$
$$m := 4.$$

Wegen $B = \emptyset$ *verlassen wir die Schleife und erhalten die Menge* $G := \{f_1, f_2, f_3\}$, *die eine Gröbnerbasis von* I *bzgl.* $\succ_{glex}$ *ist.*

Algorithmus 1.4.16. ***BuchAlg***$(f_1, \ldots, f_s)$;

Eingabe: $F := \{f_1, \ldots, f_s\} \subset K[x_1, \ldots, x_n] \backslash \{0\}$.

Ausgabe: *eine Gröbnerbasis* G *von* $I = \langle F \rangle$ *bzgl.* $\succ$ *mit* $G \supset F$.

Initialisierung:

$G := F$;

$m := s + 1$;

$B := \{(f_i, f_j) : f_i, f_j \in G, 1 \leq i < j \leq s\}$;

while $(B \neq \emptyset)$ ***do***

wähle ein Paar$(f_i, f_j) \in B$;

$B := B \backslash \{(f_i, f_j)\}$;

$f_m := \overline{\mathrm{Spoly}(f_i, f_j)}^{G}$;

if $(f_m \neq 0)$ ***then***

$B := B \cup \{(f_i, f_m) : 1 \leq i < m\}$;
$G := G \cup \{f_m\}$;
$m := m + 1$;

return(G).

Algorithmus 1.4.16: *Buchbergeralgorithmus.*

Beweis. (der Korrektheit) Siehe z.B. Buchberger (1985). □

1.5 Gröbnerfan

Die Menge

$$\mathbb{R}^n_{\geq 0} := \{(a_1, \ldots, a_n) : a_i \geq 0,\ 0 \leq i \leq n\} \subset \mathbb{R}^n$$

heißt der **positive Orthant** in $\mathbb{R}^n$.

Eine Teilmenge $E \subset \mathbb{R}^n$ heißt **konvex**, wenn

$$\overline{\beta\gamma} := \{(1 - r)\beta + r\gamma : 0 \leq r \leq 1\} \subset E$$

für je zwei Elemente $\beta, \gamma \in E$ gilt.

Sei $\alpha \in \mathbb{R}^n$. Dann heißt

$$H_\alpha := \{\beta \in \mathbb{R}^n : \beta \cdot \alpha = 0\}$$

eine **Hyperebene** in $\mathbb{R}^n$. Die Mengen

$$H_\alpha^+ := \{\beta \in \mathbb{R}^n : \beta \cdot \alpha \geq 0\} \text{ und}$$
$$H_\alpha^- := \{\beta \in \mathbb{R}^n : \beta \cdot \alpha \leq 0\}$$

heißen **abgeschlossene Halbräume** in $\mathbb{R}^n$, die von H_α begrenzt sind. Diese Halbräume sind eigentlich **konvexe Kegel**, weil gilt:

(i) $r \in \mathbb{R}, \beta \in H_\alpha^+$ bzw. $H_\alpha^- \Longrightarrow r\beta \in H_\alpha^+$ bzw. H_α^-,

(ii) $\beta, \gamma \in H_\alpha^+$ bzw. $H_\alpha^- \Longrightarrow \beta + \gamma \in H_\alpha^+$ bzw. H_α^-.

Bemerkung und Definition 1.5.1. *Sei $W = \{\omega_1, \ldots, \omega_s\}$ eine endliche Teilmenge des $\mathbb{R}^n$. Die Menge*

$$\langle W \rangle^+ := \Big\{ \sum_{i=1}^{s} r_i \omega_i : r_i \in \mathbb{R}_{\geq 0} \Big\},$$

*die durch die Elemente von W über $\mathbb{R}_{\geq 0}$ erzeugt wird, nennt man den **durch W definierten Polyederkegel**.*

Die Menge

$$\langle W \rangle^* := \{\beta \in \mathbb{R}^n : \beta \cdot \omega_i \geq 0,\ 1 \leq i \leq s\}$$

*heißt **Polarkegel** von W, und kann auch folgendermaßen beschrieben werden*

$$\langle W \rangle^* = \bigcap_{i=1}^{s} \{\beta \in \mathbb{R}^n : \beta \cdot \omega_i \geq 0\},$$

d.h. $\langle W \rangle^*$ ist ein Durchschnitt von endlich vielen abgeschlossenen Halbräumen in $\mathbb{R}^n$.

Beispiel 1.5.2. *Sei $W = \{(1,2),(2,1)\} \subset \mathbb{R}^2$. Die Abbildung 1.1 zeigt den Polyeder- und Polarkegel von W in $\mathbb{R}^2$. Der Polyederkegel von W wird durch die Strahlen $\overline{OC}^>$ und $\overline{OB}^>$ begrenzt und der Polarkegel von W durch die Strahlen $\overline{OA}^>$ und $\overline{OD}^>$.*

Für alle $\beta, \gamma \in \langle W \rangle^*$, $\omega_i \in W$ und $r \in \mathbb{R}$ mit $0 \leq r \leq 1$ gilt

$$((1-r)\beta + r\gamma) \cdot \omega_i = (1-r)\beta \cdot \omega_i + r(\gamma \cdot \omega_i) \geq 0,$$

d.h. $\langle W \rangle^*$ ist konvex.

Aus dem obigen Paragraph und den Eigenschaften einer abgeschlossenen Menge folgt:

Lemma 1.5.3. *Seien $\langle W \rangle^*$ und $\mathbb{R}^n_{\geq 0}$ wie oben. Dann ist die Menge*

$$\langle W \rangle^* \cap \mathbb{R}^n_{\geq 0}$$

ein abgeschlossener konvexer Polyederkegel, der sich im positiven Orthant befindet.

Wir betrachten wieder ein von Null verschiedenes Ideal I im Polynomring $K[\mathbf{x}] := K[x_1, \ldots, x_n]$ und eine monomiale Ordnung $\succ$ auf $K[\mathbf{x}]$. Dann setzen wir

$$B(I) := \{\mathbf{x}^\alpha \ : \ \mathbf{x}^\alpha \notin in_\succ(I)\}.$$

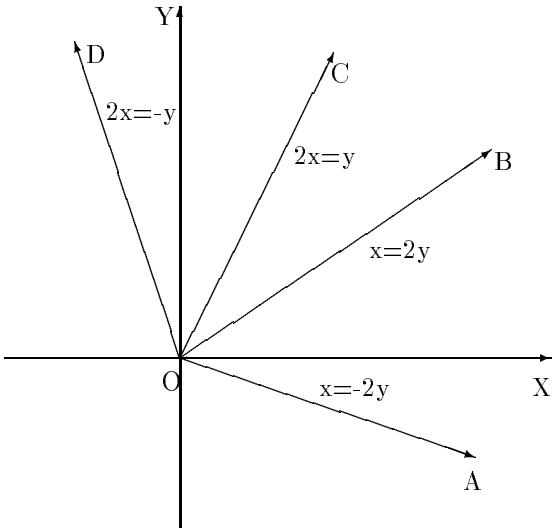

Abbildung 1.1: Polyeder- und Polarkegel von $W = \{(1,2),(2,1)\}$

Sei $f \in K[\mathbf{x}]\backslash I$ beliebig. Bei Division mit Rest von f durch eine Gröbnerbasis von I bzgl. $\succ$ erhält man

$$f = q + r \text{ mit } q \in I \text{ und } r = \sum_{\mathbf{x}^\alpha \in B(I)} a_\alpha \mathbf{x}^\alpha,$$

wobei fast alle $a_\alpha = 0$ sind. Daraus folgt, dass

$$\{\mathbf{x}^\alpha + I \ : \ \mathbf{x}^\alpha \notin in_\succ(I)\}$$

ein Erzeugendensystem von $K[\mathbf{x}]/I$ über K ist. Für eine endliche Teilmenge $S \subset B(I)$ gilt

$$\sum_{\mathbf{x}^\alpha \in S} a_\alpha \mathbf{x}^\alpha \equiv 0 \bmod I$$

genau dann, wenn alle $a_\alpha = 0$ sind (siehe Cox et al. (1997)).

Mit anderen Worten:

$$\{\mathbf{x}^\alpha + I \ : \ \mathbf{x}^\alpha \notin in_\succ(I)\}$$

ist eine K-Vektorraumbasis von $K[\mathbf{x}]/I$ (siehe Sturmfels (1996)).

Mit Hilfe dieser Tatsache, und da $K[\mathbf{x}]$ noethersch ist, kann man das folgende Lemma beweisen.

Lemma 1.5.4. *Die Menge*

$$\mathrm{Mon}(I) := \{in_\succ(I) \ : \succ \ \textit{eine monomiale Ordnung auf } K[\mathbf{x}]\}.$$

ist endlich.

Die Elemente von $\mathrm{Mon}(I)$ sind Initialideale von I bzgl. verschiedener monomialer Ordnungen auf $K[\mathbf{x}]$.

Beweis. **Annahme:** Die Menge $\mathrm{Mon}(I)$ ist *unendlich*.
Sei $0 \neq f_1 = \sum_{\alpha \in \Lambda_1} a_\alpha^{(1)} \mathbf{x}^\alpha \in I$ beliebig mit von Null verschiedenen Koeffizienten

$a_\alpha^{(1)}, \alpha \in \Lambda_1$. Für jedes $\alpha \in \Lambda_1$ existiert $M_\alpha \in \mathrm{Mon}(I)$ mit $\mathbf{x}^\alpha \in M_\alpha$. Da Λ_1 endlich und $\mathrm{Mon}(I)$ unendlich ist, erhält man eine unendliche Teilmenge

$$\Sigma_1 := \{M \in \mathrm{Mon}(I) \ : \ \mathbf{x}^{\alpha^{(1)}} \in M\}$$

von $\mathrm{Mon}(I)$ für ein $\alpha^{(1)} \in \Lambda_1$. Also ist $\langle \mathbf{x}^{\alpha^{(1)}} \rangle \subsetneq in_{\succ_1}(I)$ für eine monomiale Ordnung $\succ_1$.

Da $\{\mathbf{x}^\alpha + I \ : \ \mathbf{x}^\alpha \notin in_{\succ_1}(I)\}$ eine K-Vektorraumbasis von $K[\mathbf{x}]/I$ ist und $\langle \mathbf{x}^{\alpha^{(1)}} \rangle \subsetneq in_{\succ_1}(I)$ gilt, ist die Menge

$$\{\mathbf{x}^\alpha \ : \ \mathbf{x}^\alpha \notin \langle \mathbf{x}^{\alpha^{(1)}} \rangle\}$$

K-linear abhängig modulo I, und es existiert ein Polynom

$$0 \neq f_2 = \sum_{\alpha \in \Lambda_2} a_\alpha^{(2)} \mathbf{x}^\alpha \in I \text{ mit } \mathbf{x}^\alpha \notin \langle \mathbf{x}^{\alpha^{(1)}} \rangle \text{ für alle } \alpha \in \Lambda_2,$$

wobei Λ_2 die Multiindices der von Null verschiedenen Koeffizienten von f_2 enthält. Da Λ_2 endlich und Σ_1 unendlich ist, existiert eine unendliche Teilmenge

$$\Sigma_2 := \{M \in \Sigma_1 \ : \ \mathbf{x}^{\alpha^{(2)}} \in M\}$$

von Σ_1 für ein $\alpha^{(2)} \in \Lambda_2$. Dann gilt

$$\langle \mathbf{x}^{\alpha^{(1)}} \rangle \subsetneq \langle \mathbf{x}^{\alpha^{(1)}}, \ \mathbf{x}^{\alpha^{(2)}} \rangle \subsetneq in_{\succ_2}(I)$$

für eine monomiale Ordnung $\succ_2$.

Aus unendlich vielen Wiederholungen dieser Verfahrensweise ergibt sich eine unendliche Kette von monomialen Idealen in $K[\mathbf{x}]$:

$$\langle \mathbf{x}^{\alpha_1} \rangle \subsetneq \langle \mathbf{x}^{\alpha_1}, \ \mathbf{x}^{\alpha_2} \rangle \subsetneq \langle \mathbf{x}^{\alpha_1}, \ \mathbf{x}^{\alpha_2}, \ \mathbf{x}^{\alpha_3} \rangle \subsetneq \cdots$$

Dies ist ein Widerspruch zur Tatsache, dass der Polynomring $K[\mathbf{x}]$ noethersch ist. Also ist die Aussage richtig. □

Nach Lemma 1.5.4 ist $\mathrm{Mon}(I)$ endlich, etwa

$$\mathrm{Mon}(I) = \{M_1, \ldots, M_r\},$$

mit $M_i = in_{\succ_i}(I)$ für geignete monomiale Ordnungen $\succ_i$, $i = 1, 2, \ldots, r$.

Für alle $i = 1, 2, \ldots, r$ sei $G_i := \{g_{i1}, \ldots, g_{in_i}\}$ die reduzierte Gröbnerbasis von I bzgl. $\succ_i$, wobei

$$g_{ij} := \mathbf{x}^{\alpha_{ij}^{(0)}} + a_{\alpha_{ij}^{(1)}} \mathbf{x}^{\alpha_{ij}^{(1)}} + \cdots + a_{\alpha_{ij}^{(m_{ij})}} \mathbf{x}^{\alpha_{ij}^{(m_{ij})}} \tag{1.2}$$

mit $in_{\succ_i}(g_{ij}) = \mathbf{x}^{\alpha_{ij}^{(0)}}$ für alle $1 \leq i \leq r$ und $1 \leq j \leq n_i$ sind.

Für alle $i = 1, 2, \ldots, r$ setzen wir

$$D_i := \bigcup_{j=1}^{n_i} \{\alpha_{ij}^{(0)} - \alpha_{ij}^{(k)} : 1 \leq k \leq m_{ij}\} \tag{1.3}$$

und schreiben $D_i =: \{\delta_1^{(i)}, \ldots, \delta_{d_i}^{(i)}\}$. Der Polarkegel des durch D_i definierten Polyederkegels

$$\langle D_i\rangle^+ = \left\{\sum_{l=1}^{d_i} r_l\delta_l^{(i)} \;:\; r_l \in \mathbb{R}_{\geq 0}\right\}$$

ist dann

$$D_i^* := \langle D_i\rangle^* = \{\omega \in \mathbb{R}^n : \omega\cdot\delta \geq 0,\ \delta \in D_i\}. \tag{1.4}$$

Aus dem Lemma 1.5.3 folgt, dass die Menge $D_i^* \cap \mathbb{R}^n_{\geq 0}$ ein abgeschlossener konvexer Polyederkegel für alle $i = 1,\ldots,r$ ist.

Definition 1.5.5. *Die Menge der obigen Polyederkegel*

$$GF(I) := \{D_1^* \cap \mathbb{R}^n_{\geq 0}, \ldots, D_r^* \cap \mathbb{R}^n_{\geq 0}\}$$

heißt der ***Gröbnerfan*** *von I.*

Bemerkung 1.5.6. *Der Gröbnerfan eines Ideals ist endlich.*

Im Rest dieses Abschnittes beschäftigen wir uns mit dem Beweis des folgenden Satzes, der sich mit den Eigenschaften des Gröbnerfans beschäftigt.

Satz 1.5.7. *Für alle $i = 1,\ldots,r$ seien D_i^*, G_i wie oben. Dann gilt:*

(i) $\mathbb{R}^n_{\geq 0} = \bigcup_{i=1}^r \left(D_i^* \cap \mathbb{R}^n_{\geq 0}\right)$.

(ii) Sei A eine Matrix mit Zeilen $\omega_1,\ldots,\omega_n$, die die monomiale Ordnung $\succ_i$ definiert. Dann gibt es eine positive Zahl $\epsilon > 0$ mit

$$\sum_{l=1}^{n} \epsilon^{l-1}\omega_l \in \left(D_i^* \cap \mathbb{R}^n_{\geq 0}\right)^o := \{\omega \in \mathbb{R}^n_{\geq 0} : \omega\cdot\delta > 0,\ \delta \in D_i\}. \tag{1.5}$$

Beweis. (i) Es ist ausreichend zu zeigen, dass $\mathbb{R}^n_{\geq 0} \subset \bigcup_{i=1}^r \left(D_i^* \cap \mathbb{R}^n_{\geq 0}\right)$ ist, weil $\mathbb{R}^n_{\geq 0} \supset \bigcup_{i=1}^r \left(D_i^* \cap \mathbb{R}^n_{\geq 0}\right)$ trivialerweise erfüllt ist.

Sei $\omega \in \mathbb{R}^n_{\geq 0}$ beliebig. Dann gibt es eine monomiale Ordnung $\succ_B$, die durch eine Matrix B definiert wird, deren erste Zeile ω ist. Aus der Definition von $\mathrm{Mon}(I)$ erhalten wir ein monomiales Ideal M_i in $\mathrm{Mon}(I)$, so dass $in_{\succ_B}(I) = M_i$ ist. Aus dem Lemma (1.4.8) folgt, dass G_i auch eine Gröbnerbasis von I bzgl. $\succ_B$ ist. Dann gilt

$$\omega\cdot\alpha_{ij}^{(0)} \geq \omega\cdot\alpha_{ij}^{(k)}, \forall j = 1,2,\ldots,n_i,\ k = 1,2,\ldots,m_{ij}.$$

Also sind $\omega\cdot(\alpha_{ij}^{(0)} - \alpha_{ij}^{(k)}) \geq 0$ für alle $j = 1,2,\ldots,n_i$, $k = 1,2,\ldots,m_{ij}$, d.h. $\omega \in D_i^* \cap \mathbb{R}^n_{\geq 0}$.

(ii) Da $\succ_i$ durch die Matrix A definiert wird und $in_{\succ_i}(g_{ij}) = \mathbf{x}^{\alpha_{ij}^{(0)}}$ für alle $g_{ij} \in G_i$ wie in (1.2) ist, gilt

$$A\cdot(\alpha_{ij}^{(0)} - \alpha_{ij}^{(k)}) \succ_{lex} (0,\ldots,0) \tag{1.6}$$

für alle $j = 1,\ldots,n_i$ und $k = 1,2,\ldots,m_{ij}$, d.h.

$$A\cdot\delta \succ_{lex} (0,\ldots,0) \tag{1.7}$$

für alle $\delta \in D_i$. Also ist für alle $\delta \in D_i$ der erste von Null verschiedene Eintrag von $A\cdot\delta$ strikt positiv und die Behauptung folgt wenn wir ϵ klein genug wählen. □

Nach Lemma 1.5.7 (i) folgt:

Bemerkung 1.5.8. *Die Vereinigung aller Elemente des Gröbnerfans eines Ideals ist* $\mathbb{R}^n_{\geq 0}$.

In den folgenden Beispielen rechnen wir den Gröbnerfan eines Ideals aus.

Beispiel 1.5.9. *Sei* $I := \langle f_1, f_2, f_3\rangle \subset \mathbb{R}[x,y]$ *ein Ideal, wobei*

$$f_1 := x^3 - xy^2, \quad f_2 := y^2 - xy, \quad f_3 := xy^2 + xy + y$$

sind.

Die reduzierte Gröbnerbasis von I *bzgl.* $\succ_{lex}$ *ist*

$$F_1 := \{xy - y^2,\ y^3 + y^2 + y,\ x^3 + y^2 + y\}.$$

Aus (1.3) [3] *und (1.4)* [4] *ergibt sich*

$$D_1 := \{(1,-1),(0,1),(0,2),(3,-2),(3,-1)\},$$
$$D_1^* := \{(x,y) \in \mathbb{R}^2 : x \geq y \geq 0\}.$$

Da der Vektor $\omega_1 := (1,2) \notin D_1^*$ *ist, wählen wir die* ω_1*-gewichtete lexikographische Ordnung* $\succ_{(\omega_1, lex)}$. *Dann ist die reduzierte Gröbnerbasis von* I *bzgl.* $\succ_{(\omega_1, lex)}$*:*

$$F_2 := \{x^3 + xy + y,\ y^2 - xy,\ x^2y + xy + y\}.$$

Dann ergibt sich

$$D_2 := \{(2,-1),(3,-1),(-1,1),(1,0),(2,0)\},$$
$$D_2^* := \{(x,y) \in \mathbb{R}^2 : 0 \leq \frac{1}{2}y \leq x \leq y\}.$$

Da der Vektor $\omega_2 := (1,3) \notin D_1^* \cup D_2^*$ *ist, wählen wir die* ω_2*-gewichtete lexikographische Ordnung* $\succ_{(\omega_2, lex)}$. *Dann ist die reduzierte Gröbnerbasis von* I *bzgl.* $\succ_{(\omega_2, lex)}$*:*

$$F_3 := \{xy + x^3 + y,\ x^4 - y,\ y^2 + x^3 + y\}.$$

Dann ergibt sich

$$D_3 := \{(-2,1),(1,0),(4,-1),(-3,2),(0,1)\},$$
$$D_3^* := \{(x,y) \in \mathbb{R}^2 : 0 \leq \frac{1}{4}y \leq x \leq \frac{1}{2}y\}.$$

---

[3] Seien $g_1 := xy - y^2$, $g_2 := y^3 + y^2 + y$ und $g_3 := x^3 + y^2 + y$. Der Initialterm von g_i bzgl. $\succ_{rlex}$ ist dann xy, y^3 bzw. x^3. Also ergibt sich

$\delta_1^{(1)} := (1,1) - (0,2) = (1,-1)$, $\delta_2^{(1)} := (0,3) - (0,2) = (0,1)$, $\delta_2^{(2)} := (0,3) - (0,1) = (0,2)$,
$\delta_3^{(1)} := (3,0) - (0,2) = (3,-2)$ und $\delta_3^{(2)} := (3,0) - (0,1) = (3,-1)$.

[4] Sei $\alpha = (x,y) \in D_1^*$. Dann erhält man

$$\omega \cdot \delta_1^{(1)} = x - y \geq 0 \quad \Longrightarrow x \geq y$$
$$\omega \cdot \delta_2^{(1)} = y \geq 0, \quad \omega \cdot \delta_2^{(2)} = 2y \geq 0 \quad \Longrightarrow y \geq 0$$
$$\omega \cdot \delta_3^{(1)} = 3x - 2y \geq 0, \quad \omega \cdot \delta_3^{(2)} = 3x - y \geq 0 \quad \Longrightarrow x \geq \frac{2}{3}y \text{ und } x \geq \frac{1}{3}y.$$

Daraus folgt $x \geq y \geq 0$.

Da der Vektor[5] $\omega_3 := (1,10) \notin D_1^* \cup D_2^* \cup D_3^*$ *ist, wählen wir die* ω_3*-gewichtete lexikographische Ordnung* $\succ_{(\omega_3, lex)}$*. Dann ist die reduzierte Gröbnerbasis von* I *bzgl.* $\succ_{(\omega_3, lex)}$*:*

$$F_4 := \{x^5 + x^4 + x^3,\ y - x^4\}.$$

Dann ergibt sich

$$D_4 := \{(1,0),(2,0),(-4,1)\},$$
$$D_4^* := \{(x,y) \in \mathbb{R}^2 : 0 \leq x \leq \frac{1}{4}y\}.$$

Da

$$\begin{aligned}\mathbb{R}^2_{\geq 0} &= (D_1^* \cap \mathbb{R}^2_{\geq 0}) \cup (D_2^* \cap \mathbb{R}^2_{\geq 0}) \cup (D_3^* \cap \mathbb{R}^2_{\geq 0}) \cup (D_4^* \cap \mathbb{R}^2_{\geq 0}) \\ &= D_1^* \cup D_2^* \cup D_3^* \cup D_4^*\end{aligned}$$

ist, ist

$$GF(I) := \{D_1^*,\ D_2^*,\ D_3^*,\ D_4^*\}$$

der Gröbnerfan von I *wie in Abbildung 1.2.*

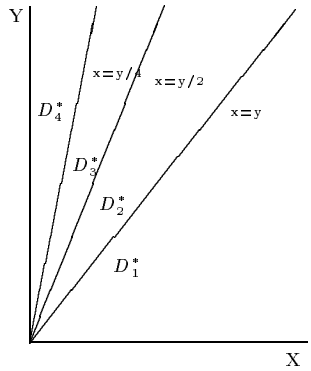

Abbildung 1.2: Gröbnerfan von I

.

Beispiel 1.5.10. *Wir bestimmen den Gröbnerfan vom Ideal*

$$I := \langle x^2 - y^3,\ xy + xz,\ x^2y + z^2x \rangle \subset \mathbb{Z}_{32003}[x,y,z].$$

Mit Hilfe des Computeralgebrasystems SINGULAR erhalten wir Gröbnerbasen G_i von I bzgl. verschiedener monomialer Ordnungen und die Elemente D_i^* des Gröbnerfans von I mit $1 \leq i \leq 9$ wie folgt:

1. Bzgl. der Ordnung $\succ_{lex}$:
   $G_1 = \{y^3z^3 + y^3z^2,\ y^4 + y^3z,\ xz^2 - y^3z,\ xy + xz,\ x^2 - y^3\}$,
   $D_1^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x - 3y + z \geq 0,\ x \geq \frac{3}{2}y,\ y \geq z\}$.

2. Bzgl. der Ordnung $\succ_{((2,1,0),\, lex)}$:
   $G_2 = \{xz^4 + xz^3,\ y^3z - xz^2,\ xy + xz,\ y^4 + xz^2,\ x^2 - y^3\}$,
   $D_2^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : y \geq z,\ x \geq \frac{3}{2}y,\ x - 3y + z \leq 0\}$.

[5] Für einen Vektor aus $\{(1,5),(1,6),(1,7),(1,8),(1,9)\}$ erhalten wir dieselbe reduzierte Gröbnerbasis wie für den Vektor $\omega_3 = (1,10)$.

3. Bzgl. der Ordnung $\succ_{((1,1,0),\ lex)}$:
   $G_3 = \{xz^4 + xz^3,\ xy + xz,\ x^2z - xz^2,\ y^3 - x^2,\ x^3 + xz^3\}$,
   $D_3^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : \frac{3}{2}z \leq x \leq \frac{3}{2}y,\ y \geq z\}$.

4. Bzgl. der Ordnung $\succ_{((1,1,1),\ lex)}$:
   $G_4 = \{xy + xz,\ y^3 - x^2,\ x^2z - xz^2,\ xz^3 + x^3,\ x^4 + x^3\}$,
   $D_4^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : z \leq x \leq \frac{3}{2}z,\ x \leq \frac{3}{2}y,\ y \geq z\}$.

5. Bzgl. der Ordnung $\succ_{((0,1,1),\ lex)}$:
   $G_5 = \{x^4 + x^3,\ xy + xz,\ x^3z + x^3,\ xz^2 - x^2z,\ y^3 - x^2\}$,
   $D_5^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x \leq z \leq y\}$.

6. Bzgl. der Ordnung $\succ_{((0,1,3),\ lex)}$:
   $G_6 = \{x^4 + x^3,\ xz + xy,\ xy^2 + x^2y,\ x^3y - x^3,\ y^3 - x^2\}$,
   $D_6^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x \leq y \leq z\}$.

7. Bzgl. der Ordnung $\succ_{((1,1,2),\ lex)}$:
   $G_7 = \{y^3 - x^2,\ xz + xy,\ x^2y + xy^2,\ x^4 + x^3\}$,
   $D_7^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : y \leq x \leq \frac{3}{2}y,\ y \leq z\}$.

8. Bzgl. der Ordnung $\succ_{((3,2,6),\ lex)}$:
   $G_8 = \{x^2 - y^3,\ y^4 + xy^2,\ xz + xy,\ y^3z - xy^2\}$,
   $D_8^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : \frac{3}{2}y \leq x \leq 2y,\ y \leq z\}$.

9. Bzgl. der Ordnung $\succ_{((1,0,1),\ lex)}$:
   $G_9 = \{y^6 - y^5,\ y^3z + y^4,\ xy^2 + y^4,\ xz + xy,\ x^2 - y^3\}$,
   $D_9^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x \geq 2y,\ y \leq z\}$.

Da $\cup_{i=1}^9 D_i^* = \mathbb{R}^3_{\geq 0}$ ist, ist der Gröbnerfan von I

$$GF(I) = \{D_1^*, D_2^*, D_3^*, D_4^*, D_5^*, D_6^*, D_7^*, D_8^*, D_9^*\}.$$

Betrachten wir die die reduzierte Gröbnerbasis von I bzgl. $\succ_{((1,1,1),\ lex)}$,

$$G_4 = \{\underline{xy} + xz,\ \underline{y^3} - x^2,\ \underline{x^2z} - xz^2,\ \underline{xz^3} + x^3,\ \underline{x^4} + x^3\},$$

so sehen wir dass gilt $in_{\succ_{((1,1,1),\ lex)}}(g) = in_{\succ_{rlex}}(g)$ für alle $g \in G_4$. Also existiert ein Gewichtsvektor $\omega \in D_4^*$, so dass $in_\omega(I) = in_{\succ_{rlex}}(I)$ ist (siehe Lemma 3.2.1 und Lemma 3.2.3).

Kapitel 2

Algorithmen

In diesem Kapitel stellen wir zuerst einige Varianten des Buchbergeralgorithmus vor. Diese beruhen auf Kriterien, die es erlauben, die Anzahl der tatsächlich zu reduzierenden S-Polynome zu verringern (siehe Buchberger (1979), Gebauer & Möller (1988) und Giovini et al. (1991)).

Um eine Gröbnerbasis mit Hilfe des Buchbergeralgorithmus oder einer seiner Varianten zu berechnen, muss man zunächst eine monomiale Ordnung wählen. Die Rechnung und die daraus resultierende Gröbnerbasis hängen ganz wesentlich von der Wahl dieser Ordnung ab. Für die meisten Anwendungen genügt es, irgendeine Gröbnerbasis zu berechnen. Es erweist sich dann als sinnvoll, die monomiale Ordnung so zu wählen, dass die Berechnung der Gröbnerbasis möglichst effizient ist. Typischerweise wählt man die reverslexikographische Ordnung (siehe Bayer & Stillmann, 1987).

Für manche Anwendungen benötigt man spezielle Gröbnerbasen und deswegen auch spezielle monomiale Ordnungen (etwa die lexikographische Ordnung zum eliminieren von Variablen, siehe Cox et al. 1997). In diesen Fällen ist es oft effizienter, die gewünschte Gröbnerbasis nicht direkt sondern über den Umweg einer reverslexikographischen Gröbnerbasis zu berechnen. Die Algorithmen von Faugère et al. (1993), Traverso (1996) und Collart et al. (1997) beruhen auf dieser Idee.

Faugère et al. verwenden lineare Algebra zur Konstruktion der gesuchten aus der reverslexikographischen Gröbnerbasis. Ihr Algorithmus gilt für nulldimensionale Ideale. Der Algorithmus von Traverso gilt für homogene Ideale, wobei hier die Hilbert-Funktion des Ideals benutzt wird. Collart et al. stellten eine andere Methode für beliebige Ideale vor, welche Eigenschaften des Gröbnerfans eines Ideals benutzt.

Für die Details verweisen wir auf die Arbeiten von Buchberger (1979, 1985), Gebauer & Möller (1988), Giovini et al. (1991), Faugère et al. (1993) und Traverso (1996). Der von Collart et al. (1997) entwickelte Algorithmus wird im nächsten Kapitel präsentiert.

2.1 Varianten des Buchbergeralgorithmus

Zur Berechnung einer Gröbnerbasis des Ideals $I := \langle G \rangle$ durch den Buchbergeralgorithmus muss man in einem einzelnen Schritt das S-Polynom von zwei Polynomen aus G berechnen und durch die Elemente von G dividieren. Ist der entstehende Rest

ungleich Null, so muss er zu G hinzugefügt werden. Ansonsten bleibt G in diesem Schritt unverändert. In der Praxis werden dabei viele Berechnungen durchgeführt, die zum Rest Null führen, also keine neuen Gröbnerbasiselemente liefern. Aus diesem Grund sucht man nach Kriterien, um Paare von Elementen in G zu erkennen, deren S-Polynome sich zu Null reduzieren lassen.

2.1.1 Zweite Variante des Buchbergeralgorithmus

Seien $F = \{f_1, f_2, \ldots, f_s\} \subset K[x_1, \ldots, x_n] \setminus \{0\}$ und $\succ$ eine monomiale Ordnung auf $K[x_1, \ldots, x_n]$.

Satz 2.1.1. ***(Erstes Buchbergerkriterium)*** *Seien $f_i, f_j \in F$. Dann gilt*

$$\gcd(in_\succ(f_i), in_\succ(f_j)) = 1 \quad \Longrightarrow \quad \mathrm{Spoly}\,(f_i, f_j) \xrightarrow{\{f_i, f_j\}}_+ 0.$$

Beweis. Siehe Cox et al. (1997). □

Definition 2.1.2. *Eine **Syzygie** der Leitterme $in_\succ(f_1), \ldots, in_\succ(f_s)$ ist ein s-Tupel von Polynomen $(h_1, \ldots, h_s) \in K[x_1, \ldots, x_n]^s$ mit*

$$h_1 \cdot in_\succ(f_1) + \cdots + h_s \cdot in_\succ(f_s) = 0.$$

Diese Syzygien bilden einen Untermodul $\mathrm{Syz}(in_\succ(F))$ *des freien $K[x_1, \ldots, x_n]$-Moduls $K[x_1, \ldots, x_n]^s$.*

Für je zwei $1 \leq i < j \leq s$ seien $m_{ij} = \frac{in_\succ(f_i)}{\gcd(in_\succ(f_i), in_\succ(f_j))}$ und

$$\begin{aligned} S_{ij} &:= m_{ji}\ \mathbf{e}_i - m_{ij}\ \mathbf{e}_j \\ &= (0, \ldots, 0,\ m_{ji}\mathbf{e}_i, 0, \ldots, 0, -m_{ij}\ \mathbf{e}_j, 0, \ldots, 0), \end{aligned}$$

wobei $\mathbf{e}_k$ der k-Einheitsvektor von $K[x_1, \ldots, x_n]^s$ ist . Wir schreiben

$$\mathcal{S} := \{S_{ij} \ : \ 1 \leq i < j \leq s\}.$$

Satz 2.1.3. *Sind F und $\mathcal{S}$ wie oben, so wird* $\mathrm{Syz}(in_\succ(F))$ *von den Elementen in $\mathcal{S}$ erzeugt.*

Beweis. Der Beweis ist in Cox et al. (1997) zu finden. □

Mit Hilfe dieser Syzygien erhalten wir die folgende Variante des Buchberger-Kriteriums:

Satz 2.1.4. *Seien F und $\mathcal{S}$ wie oben und sei $\mathcal{B} \subset \mathcal{S}$ ein Erzeugendensystem von* $\mathrm{Syz}(in_\succ(F))$. *Dann ist F eine Gröbnerbasis des Ideals $\langle F \rangle$ genau dann, wenn für jedes $(h_1, \ldots, h_s) \in \mathcal{B}$ gilt*

$$h_1 f_1 + \cdots + h_s f_s \xrightarrow{F}_+ 0.$$

Beweis. Der Beweis ist in Cox et al. (1997) zu finden. □

Satz 2.1.5. ***(Zweites Buchbergerkriterium)*** *Seien F und $\mathcal{S}$ wie oben und sei $\mathcal{B} \subseteq \mathcal{S}$ ein Erzeugendensystem von $\mathrm{Syz}(in_\succ(F))$. Sind i, j, k drei verschiedene Indizes mit $S_{ij}, S_{ik}, S_{jk} \in \mathcal{B}$, so wird $\mathrm{Syz}(in_\succ(F))$ auch durch $\mathcal{B} \setminus \{S_{ij}\}$ erzeugt, falls gilt:*

$$\mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) \text{ wird durch } in_\succ(f_k) \text{ geteilt.}$$

Beweis. Der Beweis ist in Cox et al. (1997) zu finden. □

Diese Kriterien sind die Grundlage für den sogenannten zweiten Buchbergeralgorithmus, der in Algorithmus 2.1.6 dargestellt ist. Zur Wahl des Paares $(i, j) \in B$ in

Algorithmus 2.1.6. ***2BuchAlg***$(f_1, \ldots, f_s)$;

Eingabe: *Eine endliche Menge von Polynomen*
$F := \{f_1, \ldots, f_s\}$ *in* $K[x_1, \ldots, x_n]$.

Ausgabe: *eine Gröbnerbasis G des Ideals $\langle F \rangle$ mit $F \subset G$.*

Initialisierung:

$G := F; \quad m := s + 1;$

$B := \{(i, j) : 1 \leq i < j \leq s\};$

while $(B \neq \emptyset)$ ***do***

1. *Wähle* $(i, j) \in B$;
2. ***if*** $(\gcd(in_\succ(f_i), in_\succ(f_j)) \neq 1$ *und für* (i, j) *existiert nicht ein* $k < m$ *mit* $i \neq k \neq j$, *so dass die Bedingungen des zweiten Buchbergerkriteriums erfüllt)* ***then***
   (a) $f_m := \overline{\mathrm{Spoly}\,(f_i, f_j)}^{G}$;
   (b) ***if*** $(f_m \neq 0)$ ***then***
      i. $B := B \cup \{(k, m) : 1 \leq k < m\}$;
      ii. $G := G \cup \{f_m\}$;
      iii. $m := m + 1$;
3. $B := B \setminus \{(i, j)\}$;

return(G).

Algorithmus 2.1.6: *Zweite Variante des Buchbergeralgorithmus.*

Algorithmus 2.1.6 gibt es verschiedene Strategien. Buchberger schlägt vor das Paar $(i, j) \in B$ in Algorithmus 2.1.6 so auszuwählen, dass gilt

$$\mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) = \min\{\mathrm{lcm}(in_\succ(f_u), in_\succ(f_v)) : (u, v) \in B\}. \tag{2.1}$$

Czapor (1991) schlägt vor das Paar $(i, j) \in B$ so auszuwählen, dass gilt

$$\deg(\mathrm{Spoly}\,(f_i, f_j)) = \min\{\deg(\mathrm{Spoly}\,(f_u, f_v)) : (u, v) \in B\}. \tag{2.2}$$

Beispiel 2.1.7. *(Beispiel 5.1 in Gebauer & Möller (1988)) Seien* $I := \langle f_1, f_2, f_3\rangle \subset \mathbb{Q}[x,y,z]$ *das Ideal mit*

$$f_1 := zy^2 + 2x + \frac{1}{2},\ f_2 := zx^2 - y^2 - \frac{1}{2}x,\ f_3 := -z + y^2x + 4x^2 + \frac{1}{4}$$

und $\succ_{lex}$ *die lexikographische Ordnung auf* $\mathbb{Q}[x,y,z]$ *mit* $z > y > x$.

Sei $\tau_{i,j} := \gcd(in_{\succ_{lex}}(f_i), in_{\succ_{lex}}(f_j))$. *Zur Berechnung einer Gröbnerbasis von* I *bzgl.* $\succ_{lex}$ *geht die zweite Variante des Buchbergeralgorithmus wie folgt vor:*

Sei $G := \{f_1, f_2, f_3\}$ *und* $B := \{(1,2),(1,3),(2,3)\}$.

1. *Wähle* $(2,3)$. *Dann hat man* $G = \{f_1, f_2, f_3, f_4\} =: G_1$ *mit*

$$f_4 := \overline{\mathrm{Spoly}\,(f_2, f_3)}^{G} = y^2x^3 - y^2 + 4x^4 + 1/4x^2 - 1/2x$$

und $B = \{(1,2),(1,3),(1,4),(2,4),(3,4)\}$.
Wähle $(3,4)$. *Da* $\tau_{3,4} = 1$ *ist, erhält man* $G = G_1$ *und*

$$B = \{(1,2),(1,3),(1,4),(2,4)\}.$$

2. *Wähle* $(1,3)$. *Dann hat man* $G = \{f_1, f_2, f_3, f_4, f_5\} =: G_2$ *mit*

$$f_5 := \overline{\mathrm{Spoly}\,(f_1, f_3)}^{G} = y^4x + 4y^2x^2 + 1/4y^2 + 2x + 1/2$$

und $B = \{(1,2),(1,4),(2,4),(1,5),\ldots,(4,5)\}$.

Da $\tau_{3,5} = 1$ *und das erste Buchbergerkriterium für* $(1,4)$, $(2,4)$, $(1,5)$ *bzw.* $(2,5)$ *erfüllt ist, erhält man am Ende der nächsten fünf Schleifen* $G = G_2$ *und* $B = \{(1,2),(4,5)\}$.

3. *Wähle* $(4,5)$. *Dann hat man* $G = \{f_1, f_2, \ldots, f_6\} =: G_3$ *mit*

$$f_6 := \overline{\mathrm{Spoly}\,(f_4, f_5)}^{G} = y^4 + 1/2y^2x + 2x^3 + 1/2x^2.$$

Da $\tau_{2,6} = \tau_{3,6} = 1$ *und das erste Buchbergerkriterium für* $(1,6)$ *erfüllt ist, erhält man am Ende der nächsten drei Schleifen* $G = G_3$ *und* $B = \{(1,2),(4,6),(5,6)\}$.

4. *Wähle* $(5,6)$. *Dann hat man* $G = \{f_1, f_2, \ldots, f_7\} =: G_4$ *mit*

$$f_7 := \overline{\mathrm{Spoly}\,(f_5, f_6)}^{G} = 7/2y^2x^2 + 1/4y^2 - 2x^4 - 1/2x^3 + 2x + 1/2.$$

Da $\tau_{3,7} = 1$ *und das erste Buchbergerkriterium für* $(1,2),(1,7)$ *bzw.* $(2,7)$ *erfüllt ist, erhält man am Ende der nächsten vier Schleifen* $G = G_4$ *und* $B = \{(4,6),(4,7),(5,7),(6,7)\}$.

5. *Wähle* $(4,7)$. *Dann hat man* $G = \{f_1, f_2, \ldots, f_8\} =: G_5$ *mit*

$$f_8 := \overline{\mathrm{Spoly}\,(f_4, f_7)}^{G} = -1/14y^2x - y^2 + 4/7x^5 + 29/7x^4 - 9/28x^2 - 9/14x.$$

Da $\tau_{3,8} = 1$ *und das erste Buchbergerkriterium für* $(1,8)$ *bzw.* $(2,8)$ *erfüllt ist, erhält man am Ende der nächsten drei Schleifen* $G = G_5$ *und*

$$B = \{(4,6),(5,7),(6,7),(4,8),(5,8),(6,8),(7,8)\}.$$

6. *Wähle* $(7,8)$. *Dann hat man* $G = \{f_1, f_2, \ldots, f_9\} =: G_6$ *mit*

$$f_9 := \overline{\text{Spoly}\,(f_7, f_8)}^{G} = 2745/14y^2 + 8x^6 - 54x^5 - 5688/7x^4 - 65/14x^3 + 54x^2 + 886/7x + 1/7.$$

Da $\tau_{2,9} = \tau_{3,9} = 1$ *und das erste Buchbergerkriterium für* $(1,9)$ *erfüllt ist, erhält man am Ende der nächsten drei Schleifen* $G = G_6$ *und*

$$B = \{(4,6),(5,7),(6,7),(4,8),(5,8),(6,8),(4,9),(5,9),(6,9),(7,9),(8,9)\}.$$

7. *Wähle* $(8,9)$. *Dann hat man* $G = \{f_1, f_2, \ldots, f_{10}\} =: G_7$ *mit*

$$f_{10} := \overline{\text{Spoly}\,(f_8, f_9)}^{G} = -112/2745x^7 - 812/2745x^6 + 119/2745x^4 + 154/2745x^3 - 7/5490x^2 - 7/183x - 28/2745.$$

Da $\tau_{1,10} = \tau_{3,10} = \tau_{6,10} = \tau_{9,10} = 1$ *und das erste Buchbergerkriterium für* $(2,10),(5,10),(8,10),(7,10),(7,9),(4,9),(4,10),(4,6)$ *bzw.* $(4,8)$ *erfüllt ist, erhält man am Ende der nächsten dreizehn Schleifen* $G = G_7$ *und*

$$B = \{(5,7),(6,7),(5,8),(6,8),(5,9),(6,9)\}.$$

8. *Wähle* $(6,9)$. *Da* $f_{11} := \overline{\text{Spoly}\,(f_6, f_9)}^{G} = 0$ *und das erste Buchbergerkriterium für* $(5,9),(6,8),(5,8),(6,7)$ *bzw.* $(5,7)$ *erfüllt ist, hat man* $G = G_7$ *und* $B = \emptyset$. *Deswegen terminiert der Algorithmus und liefert*

$$G = \{f_1, f_2, \ldots, f_{10}\}$$

als eine Gröbnerbasis von I *bzgl.* $\succ_{lex}$ *zurück.* □

In diesem Beispiel erhält man 45 S-Polynome, die im „ersten" Buchbergeralgorithmus (Algorithmus 1.4.16) berechnet und reduziert werden müssen. In der zweiten Variante des Buchbergeralgorithmus muss man nur 8 S-Polynome berechnen und reduzieren.

2.1.2 Algorithmus mit den Gebauer-Möller-Kriterien

Gebauer und Möller (1988) entwickelten die auf dem folgenden Satz beruhenden Kriterien.

Satz 2.1.8. ***(Gebauer-Möller-Kriterien)*** *Sei* $F = \{f_1, \ldots, f_s\} \subset K[x_1, \ldots, x_n] \setminus \{0\}$ *ein Erzeugendensystem eines Ideals* I, *auf das der Buchbergeralgorithmus angewandt werden soll. Das S-Polynom* $\text{Spoly}\,(f_i, f_j)$ *mit* $i < j$ *braucht nicht betrachtet zu werden, wenn eines der folgenden Kriterien für* i, j *erfüllt ist:*

1. ***Kriterium M:*** $\exists\ k < j$ *mit*

$$\text{lcm}(in_\succ(f_k), in_\succ(f_j)) \mid \text{lcm}(in_\succ(f_i), in_\succ(f_j)) \quad \textit{und}$$
$$\text{lcm}(in_\succ(f_k), in_\succ(f_j)) \neq \text{lcm}(in_\succ(f_i), in_\succ(f_j)).$$

2. ***Kriterium F:*** $\exists\ k < i$ *mit*

$$\mathrm{lcm}(in_\succ(f_k), in_\succ(f_j)) = \mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)).$$

3. ***Kriterium B:*** $\exists\ k > j$ *mit*

$$in_\succ(f_k) \mid \mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) \quad \textit{und}$$
$$\mathrm{lcm}(in_\succ(f_i), in_\succ(f_k)) \neq \mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) \neq \mathrm{lcm}(in_\succ(f_j), in_\succ(f_k)).$$

Beweis. Siehe Gebauer & Möller (1988). □

Seien $F = \{f_1, \ldots, f_s\} \subset K[x_1, \ldots, x_n] \setminus \{0\}$, $\emptyset \neq B \subset \{(i,j) : 1 \leq i < j \leq s\}$ und $t = s + 1$. Der auf den Kriterien basierende Unteralgorithmus **updatePairs**(B, t) funktioniert folgendermaßen:

1. Entferne die Paare (i,j) aus B, die die folgenden Bedingungen erfüllen:
   (a) $\mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) = \mathrm{lcm}(in_\succ(f_i), in_\succ(f_j), in_\succ(f_t))$,
   (b) $\mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) \neq \mathrm{lcm}(in_\succ(f_i), in_\succ(f_t))$,
   (c) $\mathrm{lcm}(in_\succ(f_i), in_\succ(f_j)) \neq \mathrm{lcm}(in_\succ(f_j), in_\succ(f_t))$.

   Wir bezeichnen mit B' die Menge der restlichen Paare.

2. Sei $B_1 := \{(i,t) : 1 \leq i < t\}$. Entferne das Paar (i,t) aus B_1, falls $(j,t) \in B_1$ existiert, so dass $\mathrm{lcm}(in_\succ(f_i), in_\succ(f_t))$ ein echtes Vielfaches von $\mathrm{lcm}(in_\succ(f_j), in_\succ(f_t))$ ist. Die Menge der restlichen Paare bezeichnen wir mit B_1'.

3. In jeder nicht leeren Teilmenge

$$T_\tau := \{(j,t) \in B_1' : \mathrm{lcm}(in_\succ(f_j), in_\succ(f_t)) = \tau\}$$

   für ein Monom τ wird das Paar $(i,t) \in T_\tau$ so ausgewählt, dass $\gcd(in_\succ(f_i), in_\succ(f_t)) = 1$ ist, falls ein solches Paar existiert. Ansonsten kann man ein beliebiges Paar $(i,t) \in T_\tau$ auswählen.

   Entferne dann die von (i,t) verschiedenen Paare aus B_1', die sich in T_τ befinden.

4. Entferne die Paare (i,t) mit $\gcd(in_\succ(f_i), in_\succ(f_t)) = 1$ aus B_1'.

5. Schließlich liefert der Algorithmus die Menge $B_1' \cup B'$ zurück.

Mit dem auf den Gebauer-Möller-Kriterien basierenden Algorithmus 2.1.9 kann man eine Gröbnerbasis eines beliebigen Ideals im Polynomring $K[x_1, \ldots, x_n]$ über dem Körper K ausrechnen.

Beispiel 2.1.10. *Wir betrachten wieder Beispiel 2.1.7. Zur Berechnung einer Gröbnerbasis von I bzgl. $\succ_{lex}$ geht der auf den Gebauer-Möller-Kriterien basierende Algorithmus wie folgt vor:*

1. *Vor der ersten Schleife erhält man $B = \{(1,3), (2,3)\}$.*
2. *Am Ende der ersten Schleife erhält man $B = \{(1,3)\}$ und $G = \{f_1, f_2, f_3, f_4\}$.*
3. *Am Ende der zweiten Schleife erhält man $B = \{(4,5)\}$ und $G = \{f_1, f_2, \ldots, f_5\}$.*

Algorithmus 2.1.9. ***GMAlg***$(f_1, \ldots, f_s)$;

Eingabe: *Eine endliche Menge von Polynomen* $F := \{f_1, \ldots, f_s\}$ *in* $K[x_1, \ldots, x_n] \setminus \{0\}$.

Ausgabe: *eine Gröbnerbasis* G *des Ideals* $\langle F \rangle$ *mit* $F \subset G$.

Initialisierung:

$G := F; \quad B := \emptyset; \quad m := s + 1;$

for $t = 2$ ***to*** s

$B :=$ ***updatePairs***(B, t);

while $(B \neq \emptyset)$ ***do***

Wähle $(i, j) \in B$, *das die Gleichung 2.1 erfüllt*;

$B := B \setminus \{(i, j)\}$;

$f_m := \overline{\text{Spoly}\,(f_i, f_j)}^{G}$;

if $(f_m \neq 0)$ ***then***

$B :=$ ***updatePairs***(B, m);
$G := G \cup \{f_m\}$;
$m := m + 1$;

return(G).

Algorithmus 2.1.9: *Algorithmus mit den Gebauer-Möller-Kriterien.*

4. *Am Ende der dritten Schleife erhält man* $B = \{(1,6),(5,6)\}$ *und* $G = \{f_1, f_2, \ldots, f_6\}$.

5. *Am Ende der vierten Schleife erhält man* $B = \{(1,6),(4,7),(6,7)\}$ *und* $G = \{f_1, f_2, \ldots, f_7\}$.

6. *Am Ende der fünften Schleife erhält man* $B = \{(1,6),(6,8),(7,8)\}$ *und* $G = \{f_1, f_2, \ldots, f_8\}$.

7. *Am Ende der sechsten Schleife erhält man* $B = \{(6,9),(8,9)\}$ *und* $G = \{f_1, f_2, \ldots, f_9\}$.

8. *Am Ende der siebten Schleife erhält man* $B = \{(6,9)\}$ *und* $G = \{f_1, f_2, \ldots, f_{10}\}$.

9. *In der achten Schleife ist die Normalform von* $\text{Spoly}\,(f_6, f_9)$ *bzgl.* G *gleich Null und* $B = \emptyset$. *Deswegen terminiert der Algorithmus und liefert* $G = \{f_1, f_2, \ldots, f_{10}\}$ *als eine Gröbnerbasis von* I *bzgl.* $\succ_{lex}$, *wobei* $f_1, f_2, \ldots, f_{10}$ *wie in Beispiel 2.1.7 sind.* □

In diesem Beispiel verhalten sich die beiden Varianten des Buchbergeralgorithmus ähnlich.

2.2 Algorithmus mit Sugar-Strategie

Wie bereits angedeutet spielt die Strategie zur Auswahl des "nächsten" S-Polynoms im Buchbergeralgorithmus eine wichtige Rolle. Für homogene Ideale, mit einer gradverträglichen monomialen Ordnung und mit Buchbergers in 2.1.1 vorgestellter Auswahlstrategie, funktioniert der Buchbergeralgorithmus üblicherweise gut. Ist ein inhomogenes Ideal gegeben, so kann man es homogenisieren, die Gröbnerbasis des homogenisierten Ideals berechnen und diese wieder dehomogenisieren. Diese Idee führt aber zu viel zu grossen Gröbnerbasen des inhomogenen Ideals. Eine Alternative zum homogenisieren ist die Sugar-Strategie (siehe Giovini et al. 1991). Sie beruht darauf, dass man im Verlauf des Buchbergeralgorithmus jedem neu berechneten Gröbnerbasiselement den Grad zuordnet, den es hätte, wenn man es im homogenisierten Fall berechnet hätte. Dieser Phantomgrad (der Sugar) wird nur für die Auswahlstrategie benutzt. In diesem Abschnitt stellen wir eine Variante dieses Verfahrens vor.

Definition 2.2.1. *Sei $\{f_1, \ldots, f_s\}$ die Eingabe des Buchbergeralgorithmus. Der* ***Sugar*** *eines im Buchbergeralgorithmus aufgetretenen Polynoms f wird mit S_f bezeichnet und Schritt für Schritt wie folgt erklärt:*

(i) $S_{f_i} = \deg(f_i)$ *für alle* $i = 1, \ldots, s$.

(ii) Ist m ein Monom, so ist $S_{mf} = \deg(m) + S_f$.

(iii) Ist $f = g + h$ für zwei Polynome g und h, so ist $S_f = \max(S_g, S_h)$.

Für zwei Polynome f_i, f_j und eine monomiale Ordnung $\succ$ setzen wir

$$\tau_{ij} := \operatorname{lcm}(in_\succ(f_i), in_\succ(f_j))$$
$$s_{ij} := \max\{S_{f_i} - \deg(in_\succ(f_i)),\ S_{f_j} - \deg(in_\succ(f_j))\} + \deg(\tau_{ij}).$$

Dann bezeichnet man auch mit

$$P_{ij} := ((i,j), s_{ij}, \tau_{ij})$$

das „kritische" Paar[1)] von zwei Polynomen f_i, f_j und mit τ_i den Initialterm von f_i.

Zum Vergleich von zwei Paaren verwenden wir die folgende Definition.

Definition 2.2.2. *Sei $P := \{P_{ij} \mid 1 \leq i < j \leq s\}$ die Menge der kritischen Paare von zwei verschiedenen Polynomen aus $\{f_1, f_2, \ldots, f_s\}$. Für je zwei kritische Paare $P_{ij}, P_{ab} \in P$ definiert man dann*

$$P_{ij} < P_{ab} \iff \begin{cases} s_{ij} < s_{ab} \text{ oder} \\ (s_{ij} = s_{ab} \text{ und } \tau_{ij} \prec \tau_{ab}) \text{ oder} \\ (s_{ij} = s_{ab},\ \tau_{ij} = \tau_{ab} \text{ und } j < b). \end{cases} \tag{2.3}$$

Dies wird in der Variante (siehe Algorithmus 2.2.3) des Buchbergeralgorithmus zur Ordnung der endlichen Menge kritischer Paare verwendet. In Algorithmus 2.2.3 wird die Sugar-Strategie mit der so genannten Fussy-Variante vorgestellt. Durch diese Strategie werden im inhomogenen Fall typischerweise weniger kritische Paare berechnet und reduziert als mit der normalen Auswahlstrategie (siehe Tabellen in Giovini et al. (1991)).

[1)] Das wahre kritische Paar ist (i, j).

Algorithmus 2.2.3. ***Sugar***$(f_1, \ldots, f_s)$;

Eingabe: *Eine endliche Menge von Polynomen* $F := \{f_1, \ldots, f_s\}$ *in* $K[x_1, \ldots, x_n]$.

Ausgabe: *Eine Gröbnerbasis* G *vom Ideal* $\langle F \rangle$ *mit* $F \subset G$.

Initialisierung:

- $k := s + 1$;
- $G := F; \quad P_1 := \emptyset$;
- $P := \{((i,j), s_{ij}, \tau_{ij}) \ : \ 1 \leq i < j \leq s\}$;

while $(P \neq \emptyset)$ ***do***

1. (i,j) := *die erste Komponente des ersten Elements* P_{ij} *der durch Gleichung (2.3) geordneten Menge von* P;
2. $P := P \setminus \{P_{ij}\}$;
3. $f_k := \overline{\mathrm{Spoly}\,(f_i, f_j)}^{G}$;
4. ***if*** $(f_k \neq 0)$ ***then***
   (a) $G := G \cup \{f_k\}$;
   (b) $P := P \setminus \{((i,j), s_{ij}, \tau_{ij}) : \tau_{ij}$ *ist ein striktes Vielfaches von* τ_{ik} *und* τ_{jk}, $s_{ik} \leq s_{ij}$ *und* $s_{jk} \leq s_{ij}\}$;
   (c) $P_1 := \{((i,k), s_{ik}, \tau_{ik}) \mid 1 \leq i < k\}$;
   (d) ***while*** $(1 \leq i < k)$ ***do***
   if $(\gcd(\tau_i, \tau_k) == 1)$ ***then***
   $P_1 := P_1 \backslash \{((j,k), s_{jk}, \tau_{jk}) \ : \ \tau_{jk} = \tau_{ik}\}$;
   (e) $P_1 := \{P_{i_1 k}, \ldots, P_{i_s k}\}$;
   (*die durch Gleichung (2.3) geordnete Menge von* P_1)
   (f) $P_1 := P_1 \setminus \{P_{ik} \ : \ \exists P_{jk} < P_{ik}, \tau_{jk}$ *teilt* $\tau_{ik}\}$;
   (g) $P_1 := P_1 \setminus \{P_{ik} \ : \ \gcd(\tau_i, \tau_k) = 1\}$;
   (h) $P := P \cup P_1$;
   (i) $k := k + 1$;

return(G).

Algorithmus 2.2.3: *Fussy-Variante der Sugar-Strategie.*

Beispiel 2.2.4. *Wir betrachten wieder Beispiel 2.1.7. Zur Berechnung einer Gröbnerbasis von I bzgl. $\succ_{lex}$ geht der Algorithmus wie folgt vor:*

Aus den drei Polynomen f_1, f_2 und f_3 ergibt sich

1. $S_{f_1} = \deg(f_1) = 3$, $S_{f_2} = 3$ *und* $S_{f_3} = 3$,
2. $s_{12} = \max\{3-3, 3-3\} + 5 = 5$, $s_{13} = 5$ *und* $s_{23} = 5$,
3. $\tau_{12} = zy^2x^2$, $\tau_{13} = zy^2$ *und* $\tau_{23} = zx^2$.

Deswegen erhält man vor der ersten Schleife $P := \{P_{2,3}, P_{1,3}, P_{1,2}\}$.

1. *Wähle $(2,3)$. Am Ende der ersten Schleife erhält man $G := \{f_1, f_2, f_3, f_4\}$ und $P = \{P_{1,3}, P_{1,2}\}$*[2] *mit*

$$f_4 := y^2x^3 - y^2 + 4x^4 + 1/4x^2 - 1/2x$$

und $f_4 = f_2 - x^2 f_3$. Also ist

$$S_{f_4} = \max\{S_{f_2}, S_{x^2 f_3}\} = \max\{3, 2+3\} = 5.$$

2. *Wähle $(1,3)$. Am Ende der zweiten Schleife erhält man $G := \{f_1, f_2, \ldots, f_5\}$ und $P = \{P_{1,2}, P_{4,5}\}$ mit*

$$f_5 := y^4x + 4y^2x^2 + 1/4y^2 + 2x + 1/2.$$

3. *Wähle $(1,2)$. Am Ende der dritten Schleife erhält man $G := \{f_1, f_2, \ldots, f_6\}$ und $P = \{P_{5,6}, P_{4,5}\}$ mit*

$$f_6 := y^4 + 1/2y^2x + 2x^3 + 1/2x^2.$$

4. *Wähle $(5,6)$. Am Ende der vierten Schleife erhält man $G := \{f_1, f_2, \ldots, f_7\}$ und $P = \{P_{4,7}, P_{4,5}, P_{5,7}\}$ mit*

$$f_7 := 7/2y^2x^2 + 1/4y^2 - 2x^4 - 1/2x^3 + 2x + 1/2.$$

5. *Wähle $(4,7)$. Am Ende der fünften Schleife erhält man $G := \{f_1, f_2, \ldots, f_8\}$ und $P = \{P_{4,5}, P_{7,8}, P_{5,7}, P_{5,8}\}$ mit*

$$f_8 := -1/14y^2x - y^2 + 4/7x^5 + 29/7x^4 - 9/28x^2 - 9/14x.$$

6. *Wähle $(4,5)$. Am Ende der sechsten Schleife erhält man $P = \{P_{7,8}, P_{5,7}, P_{5,8}\}$ und G bleibt unverändert, weil das S-Polynom $S(f_4, f_5)$ sich zu Null reduziert.*

---

[2] Aus den vier Polynomen f_1, f_2, f_3 und f_4 ergibt sich

(a) $\sigma_{14} = \gcd(in_\succ(f_1), in_\succ(f_4)) = y^2$, $\tau_{14} = zy^2x^3$ und $s_{14} = 6$,

(b) $\sigma_{24} = x^2$, $\tau_{24} = zy^2x^3$ und $s_{24} = 6$,

(c) $\sigma_{34} = 1$, $\tau_{34} = zy^2x^3$ und $s_{34} = 7$.

Da $\sigma_{34} = 1$ und $\tau_{14} = \tau_{24} = \tau_{34}$ sind, erhält man $P_1 = \emptyset$. Also ist $P = \{P_{1,3}, P_{1,2}\}$.

7. *Wähle* $(7,8)$. *Am Ende der siebsten Schleife erhält man* $G := \{f_1, f_2, \ldots, f_9\}$ *und* $P = \{P_{5,7}, P_{8,9}, P_{5,8}, P_{6,9}\}$ *mit*

$$f_9 := 2745/14y^2 + 8x^6 - 54x^5 - 5688/7x^4 - 65/14x^3 + 54x^2 + 886/7x + 1/7.$$

8. *Wähle* $(5,7)$. *Am Ende der achten Schleife erhält man* $P = \{P_{8,9}, P_{5,8}, P_{6,9}\}$ *und* G *bleibt unverändert, weil das S-Polynom* $S(f_5, f_7)$ *sich zu Null reduziert.*

9. *Wähle* $(8,9)$. *Am Ende der neunten Schleife erhält man* $G := \{f_1, f_2, \ldots, f_{10}\}$ *und* $P = \{P_{5,8}, P_{6,9}\}$ *mit*

$$f_{10} := -112/2745x^7 - 812/2745x^6 + 119/2745x^4 + 154/2745x^3 - 7/5490x^2 - 7/183x - 28/2745.$$

10. *Wähle* $(5,8)$. *Am Ende der zehnten Schleife erhält man* $P = \{P_{6,9}\}$ *und* G *bleibt unverändert, weil das S-Polynom* $S(f_5, f_8)$ *sich zu Null reduziert.*

11. *Wähle* $(6,9)$. *Im elften Schleife reduziert S-Polynom* $S(f_6, f_9)$ *sich zu Null. Da* $P = \emptyset$ *ist, terminiert die Berechnung und liefert* $G := \{f_1, f_2, \ldots, f_{10}\}$ *als eine Gröbnerbasis von* I *bzgl.* $\succ_{lex}$. □

2.3 Algorithmen mit zwei monomialen Ordnungen

2.3.1 Hilbert-driven Algorithmus

Sei I ein homogenes Ideal in $K[\mathbf{x}] := K[x_1, \ldots, x_n]$. Für eine positive ganze Zahl s bezeichnen wir mit $K[x_1, \ldots, x_n]_s$ den K-Vektorraum der homogenen Polynome vom Grad s (einschliesslich des Nullpolynoms). Dieser hat die Dimension $\binom{n-1+s}{s}$.

Setzen wir

$$I_s := I \cap K[x_1, \ldots, x_n]_s,$$

so ist I_s ein Untervektorraum von $K[x_1, \ldots, x_n]_s$. Also ist $K[x_1, \ldots, x_n]_s/I_s$ ein endlichdimensionaler Vektorraum über K.

Ist I ein monomiales Ideal, so ist die Dimension von $K[x_1, \ldots, x_n]_s/I_s$ die Anzahl der Monome vom Grad s, die nicht in I liegen.

Definition 2.3.1. *Die* ***Hilbert-Funktion*** *von* $K[\mathbf{x}]/I$ *ist definiert als*

$$\begin{aligned} H_{K[\mathbf{x}]/I} &: \mathbb{N} \longrightarrow \mathbb{N} \\ &: s \longmapsto dim(K[x_1, \ldots, x_n]_s/I_s). \end{aligned}$$

Die Potenzreihe

$$P_{K[\mathbf{x}]/I}(z) := \sum_{s=0}^{\infty} H_{K[\mathbf{x}]/I}(s) z^s$$

heißt ***Hilbert-Poincaré-Reihe*** *von* $K[\mathbf{x}]/I$.

Satz 2.3.2. *Die Hilbert-Poincaré-Reihe* $P_{K[\mathbf{x}]/I}$ *von* $K[\mathbf{x}]/I$ *ist eine rationale Funktion der Form* $\frac{f_I(z)}{(1-z)^n}$ *mit* $f_I(z) \in \mathbb{Z}[z]$.

Beweis. Siehe Greuel & Pfister (2002). □

Satz 2.3.3. *Ist $I \subset K[\mathbf{x}]$ ein homogenes Ideal und $\succ$ eine monomiale Ordnung auf $K[\mathbf{x}]$, so gilt*

$$H_{K[\mathbf{x}]/in_{\succ}(I)} = H_{K[\mathbf{x}]/I}.$$

Das heißt, die Hilbert-Funktion von $K[\mathbf{x}]/I$ und $K[\mathbf{x}]/in_{\succ}(I)$ stimmen überein.

Beweis. Siehe Cox et al. (1997). □

Dann gilt:

Korollar 2.3.4. *Seien $\succ_1$ und $\succ_2$ zwei monomiale Ordnungen und sei I ein homogenes Ideal. Dann gilt*

$$H_{K[\mathbf{x}]/in_{\succ_1}(I)} = H_{K[\mathbf{x}]/I} = H_{K[\mathbf{x}]/in_{\succ_2}(I)}.$$

Korollar 2.3.5. *Ist G eine Gröbnerbasis von I bzgl. $\succ$, so ist*

$$H_{K[\mathbf{x}]/I} = H_{K[\mathbf{x}]/\langle in_{\succ}(G)\rangle}.$$

Satz 2.3.6. *Sei $\succ$ eine monomiale Ordnung und G eine Teilmenge des homogenen Ideals I. Dann gilt:*

(i) $H_{K[\mathbf{x}]/in_{\succ}(I)}(s) \leq H_{K[\mathbf{x}]/\langle in_{\succ}(G)\rangle}(s)$, $\forall s \in \mathbb{N}$.

(ii) Ist G endlich und

$$H_{K[\mathbf{x}]/in_{\succ}(I)} = H_{K[\mathbf{x}]/\langle in_{\succ}(G)\rangle},$$

so ist G eine Gröbnerbasis von I bzgl. $\succ$.

Beweis. Siehe Traverso (1996). □

Von diesem Satz erhalten wir die folgende Bemerkung.

Bemerkung 2.3.7. *Sei G eine endliche Teilmenge von $I \subset K[\mathbf{x}]$, die keine Gröbnerbasis von I bzgl. einer monomialen Ordnung $\succ$ ist. Dann existieren zwei ganze Zahlen m und k, so dass gilt:*

(i) $H_{K[\mathbf{x}]/\langle in_{\succ}(I)\rangle}(s) = H_{K[\mathbf{x}]/\langle in_{\succ}(G)\rangle}(s)$, $\forall 0 \leq s < m$ *und*

(ii) $k = H_{K[\mathbf{x}]/\langle in_{\succ}(G)\rangle}(m) - H_{K[\mathbf{x}]/\langle in_{\succ}(I)\rangle}(m) > 0$.

Da $H_{K[\mathbf{x}]/\langle in_{\succ}(I)\rangle}(m)$ bzw. $H_{K[\mathbf{x}]/\langle in_{\succ}(G)\rangle}(m)$ die Anzahl der nicht in $in_{\succ}(I)$ bzw. $in_{\succ}(G)$ liegenden Monome vom Grad m ist, müssen zu diesem Zeitpunkt k Polynome im Grad m zu G hinzugefügt werden. Diese Polynome ergeben sich aus dem Rest von S-Polynomen, deren Grad m ist, bei Division durch G.

Satz 2.3.8. *Seien I und J zwei homogene Ideale in $K[\mathbf{x}]$ mit $J \subset I$. Sei $\frac{f_I(z)}{(1-z)^n}$ (bzw. $\frac{f_J(z)}{(1-z)^n}$) die Hilbert-Poincaré-Reihe von $K[\mathbf{x}]/I$ (bzw. $K[\mathbf{x}]/J$) mit*

$$\begin{aligned} f_I(z) &= 1 + a_1 z + a_2 z^2 + \cdots + a_u z^u \text{ und} \\ f_J(z) &= 1 + b_1 z + b_2 z^2 + \cdots + b_v z^v. \end{aligned}$$

Dann sind die folgenden Bedingungen äquivalent:

(i) $H_{K[\mathbf{x}]/I}(s) = H_{K[\mathbf{x}]/J}(s)$ *für* $s = 1, \ldots, m-1$ *und* $H_{K[\mathbf{x}]/I}(m) < H_{K[\mathbf{x}]/J}(m)$.

(ii) $a_1 = b_1, \ldots, a_{m-1} = b_{m-1}$ *und* $a_m < b_m$.

Sind diese Bedingungen erfüllt, so gilt $\dim I_m - \dim J_m = b_m - a_m$.

Beweis. Siehe Traverso (1996). □

Mit Hilfe der Hilbert-Poincaré-Reihe entwickelte Traverso (1996) einen Algorithmus zur Berechnung einer Gröbnerbasis eines homogenen Ideals, der **Hilbert-driven Algorithmus** genannt wird (siehe Algorithmus 2.3.9).

Algorithmus 2.3.9. ***Hilbert_Driven***$(G, \succ)$;

Eingabe: *Eine endliche Menge* G *homogener Polynome in* $K[\mathbf{x}]$.

Ausgabe: *Eine Gröbnerbasis des Ideals* $\langle G \rangle$ *bzgl.* $\succ$.

Initialisierung:

- $F :=$ *eine Gröbnerbasis von* $\langle G \rangle$ *bzgl. einer von* $\succ$ *verschiedenen monomialen Ordnung* $\gg$;
- $H :=$ *die Hilbert-Funktion von* $K[\mathbf{x}]/\langle in_{\gg}(F) \rangle$;

while (1) ***do***

1. $l := \#(G)$; *(die Anzahl der Elemente von* G*)*
2. $H_1 :=$ *die Hilbert-Funktion von* $K[\mathbf{x}]/\langle in_{\succ}(G) \rangle$;
3. ***if*** $(H == H_1)$ ***then return***(G);
4. $m := \min\{s \mid H(s) \neq H_1(s)\}$;
5. $k := H_1(m) - H(m)$;
6. $B := \{(f_i, f_j) \mid f_i, f_j \in G,\ 1 \leq i < j \leq l\}$;
7. ***while*** $(B \neq \emptyset)$ ***do***
   - *(a)* *wähle* $(f_i, f_j) \in B$;
   - *(b)* $B := B \setminus \{(f_i, f_j)\}$;
   - *(c)* $h := \mathrm{Spoly}(f_i, f_j)$;
   - *(d)* ***if*** $(\deg(h) == m)$ ***then***
     - *i.* $r :=$ *ein Rest von* h *bei Division durch* G *bzgl.* $\succ$;
     - *ii.* ***if*** $(r \neq 0)$ ***then*** $G := G \cup \{r\}$;
     - *iii.* ***if*** $(\#(G) - l == k)$ ***then go to*** *step 1.*

Algorithmus 2.3.9: *Hilbert-driven-Algorithmus.*

Bemerkung 2.3.10. *Analog ergibt sich ein Hilbert-driven Algorithmus zur Berechnung einer Gröbnerbasis eines ω-homogenen Ideals mit Hilfe der ω-gewichteten Hilbertfunktion.*

Beispiel 2.3.11. *Seien* $I := \langle f_1, f_2\rangle \subset \mathbb{Q}[a,b,c,d] =: R$ *das homogene Ideal mit*

$$f_1 := a^2b - c^3, \quad f_2 := ab^3 - d^4$$

und $\succ_{lex}$ *die lexikographische Ordnung auf* $\mathbb{Q}[a,b,c,d]$ *mit* $a > b > c > d$.

Zur Berechnung einer Gröbnerbasis von I *bzgl.* $\succ_{lex}$ *geht der Hilbert-driven Algorithmus wie folgt vor:*

Zuerst wird eine Gröbnerbasis von I *bzgl. einer von* $\succ_{lex}$ *verschiedenen Ordnung auf* $\mathbb{Q}[a,b,c,d]$, *etwa* $\succ_{rlex}$ *mit* $a > b > c > d$, *ausgerechnet. Diese ist*

$$F := \{a^2b - c^3,\ ab^3 - d^4,\ b^2c^3 - ad^4,\ bc^6 - a^3d^4,\ c^9 - a^5d^4\}.$$

Der Zähler der Hilbert-Poincaré-Reihe von R/I_1 *mit* $I_1 := \langle in_{\succ_{rlex}}(F)\rangle$ *ist*

$$f_{I_1}(z) = 1 - z^3 - z^4 + z^7.$$

Der Zähler der Hilbert-Poincaré-Reihe von R/I_2 *mit* $I_2 := \langle in_{\succ_{lex}}(f_1), in_{\succ_{lex}}(f_2)\rangle = \langle a^2b, ab^3\rangle$ *ist*

$$f_{I_2}(z) = 1 - z^3 - z^4 + z^5.$$

Also ist $H_{R/I_1} \neq H_{R/I_2}$, *weil* $H_{R/I_1}(s) = H_{R/I_2}(s)$ *nur für* $s = 0,1,2,3,4$ *und* $H_{R/I_2}(5) - H_{R/I_1}(5) = 1$ *ist.*

Das S-Polynom von f_1 *und* f_2 *(bzgl.* $\succ_{lex}$*) ist*

$$\mathrm{Spoly}\,(f_1, f_2) = b^2 \cdot f_1 - a \cdot f_2 = ad^4 - b^2c^3.$$

Da $\deg(\mathrm{Spoly}\,(f_1, f_2)) = 5$ *ist und* H_{R/I_2} *sich von* H_{R/I_1} *im Grad 5 erstmals unterscheidet, berechnen wir die Normalform des S-Polynoms bzgl.* G *mit der Ordnung* $\succ_{lex}$ *und erhalten* $f_3 := ad^4 - b^2c^3$. *Dieses Polynom muss zu* f_1 *und* f_2 *hinzugefügt werden, d.h.* $G = \{f_1, f_2, f_3\}$ *und* $in_{\succ_{lex}}(G) = \{a^2b, ab^3, ad^4\}$.

Der Zähler der Hilbert-Poincaré-Reihe von R/I_3 *mit* $I_3 := \langle in_{\succ_{lex}}(G)\rangle$ *ist*

$$f_{I_3} = 1 - z^3 - z^4 + z^7 + z^8 - z^9.$$

H_{R/I_1} *unterscheidet sich von* H_{R/I_3} *im Grad 8, weil* $H_{R/I_1}(s) = H_{R/I_3}(s)$ *nur für* $s = 0, 1, \ldots, 7$ *und* $H_{R/I_3}(8) - H_{R/I_1}(8) = 1$ *ist.*

Wir berechnen das S-Polynom jedes Paares (f_i, f_j) *mit* $1 \leq i < j \leq 3$. *Dann ergibt sich*

$$\mathrm{Spoly}\,(f_1, f_3) = d^4 \cdot f_1 - ab \cdot f_3 = ab^3c^3 - c^3d^4,$$
$$\mathrm{Spoly}\,(f_2, f_3) = d^4 \cdot f_2 - b^3 \cdot f_3 = b^5c^3 - d^8.$$

Da H_{R/I_3} *sich von* H_{R/I_1} *erstmals im Grad 8 unterscheidet und das Polynom* $\mathrm{Spoly}\,(f_1, f_3)$ *Grad 7 hat, kann man* $\mathrm{Spoly}\,(f_1, f_3)$ *weglassen und berechnet nur die Normalform von* $\mathrm{Spoly}\,(f_2, f_3)$ *bzgl.* G *mit der Ordnung* $\succ_{lex}$. *Diese ist* $f_4 := b^5c^3 - d^8$.

Das Polynom f_4 *muss zu* G *hinzugefügt werden, d.h.* $G = \{f_1, f_2, f_3, f_4\}$ *und* $in_{\succ_{lex}}(G) = \{a^2b, ab^3, ad^4, b^5c^3\}$.

Die Zähler der Hilbert-Poincaré-Reihe von R/I_4 *mit* $I_4 := \langle in_{\succ_{lex}}(G)\rangle$ *ist*

$$f_{I_4} = 1 - z^3 - z^4 + z^7.$$

Wegen $H_{R/I_4} = H_{R/I_1}$ *ist* $\{f_1, f_2, f_3, f_4\}$ *eine Gröbnerbasis von* I *bzgl.* $\succ_{lex}$. □

2.3.2 FGLM-Algorithmus

Faugère, **G**ianni, **L**azard und **M**ora (1993) entwickelten einen Basiskonvertierungsalgorithmus für nulldimensionale Ideale, der **FGLM-Algorithmus** genannt wird. In diesem Unterkapitel geben wir einen Überblick über diesen Algorithmus.

Definition 2.3.12. *Ein Ideal $I \subset K[x_1,\ldots,x_n]$ heißt **nulldimensionales Ideal**, falls die Dimension des K-Vektorraums $K[x_1,\ldots,x_n]/I$ endlich ist.*

Satz 2.3.13. *Sei G die reduzierte Gröbnerbasis eines nulldimensionalen Ideals $I \subset K[x_1,\ldots,x_n]$ bzgl. einer monomialen Ordnung $\succ$. Für jedes $i \in \{1,\ldots,n\}$ existiert dann ein $0 < \nu \in \mathbb{Z}$, so dass $x_i^\nu = in_\succ(g)$ für ein $g \in G$ gilt.*

Beweis. Siehe Cox et al. (1997). □

Sei $I \subset K[x_1,\ldots,x_n]$ ein nulldimensionales Ideal und G die reduzierte Gröbnerbasis von I bzgl. einer monomialen Ordnung $\succ$.

Ist

$$B(G) := \{m \in K[x_1,\ldots,x_n] \text{ Monom } : \ m \notin in_\succ(I)\},$$

so ist $\{m + I : m \in B(G)\}$ eine endliche Basis des K-Vektorraums

$$K[x_1,\ldots,x_n]/I := \{\overline{f}^G + I : f \in K[x_1,\ldots,x_n]\}$$

mit üblicher Addition und Skalarmultiplikation. Jedes Monom von $B(G)$ bezeichnen wir als *Standardmonom* bzgl. G. Falls nicht anders angegeben, setzen wir $B(G) := \{b_1,\ldots,b_d\}$ mit $b_d \succ b_{d-1} \succ \cdots \succ b_1$.

Setzen wir

$$M(G) := \{x_i \cdot b \mid b \in B(G),\ 1 \le i \le n,\ x_i \cdot b \notin B(G)\},$$

so gilt für jedes $m \in M(G)$ genau eine der folgenden Bedingungen:

1. $\exists g \in G : m = in_\succ(g)$ (In diesem Fall gilt: wird m durch x_i geteilt, so ist $\frac{m}{x_i} \in B(G)$).
2. $\exists x_j : \frac{m}{x_j} \in M(G)$.

Beispiel 2.3.14. *Gegeben sei $G = \{xy+y^2, y^3-x^2, x^3+x^2\}$ die reduzierte Gröbnerbasis eines nulldimensionalen Ideals I bzgl. $\succ_{rlex}$ mit $x \succ_{rlex} y$.*

Dann erhalten wir

$$\begin{aligned} in_{\succ_{rlex}}(G) &= \{xy, y^3, x^3\}, \\ B(G) &= \{1, y, x, y^2, x^2\} \text{ und} \\ M(G) &= \{xy, y^3, xy^2, x^2y, x^3\}. \end{aligned}$$

Betrachten wir die Exponenten der Monome von $in_{\succ_{rlex}}(G), B(G)$ bzw. $M(G)$, so erhalten wir drei Mengen

$$\begin{aligned} exp_{in_{\succ_{rlex}}(G)} &:= \{(1,1),(0,3),(3,0)\}, \\ exp_{B(G)} &:= \{(0,0),(0,1),(1,0),(0,2),(2,0)\} \text{ bzw.} \\ exp_{M(G)} &:= \{(1,1),(0,3),(1,2),(2,1),(3,0)\} \end{aligned}$$

wie die Abbildung 2.1 zeigt.

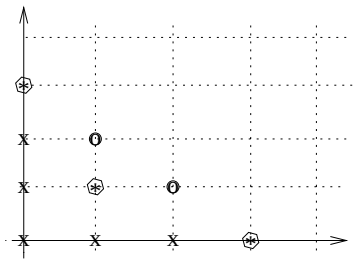

Abbildung 2.1: Exponenten von $B(G)$, $M(G)$ und $in_{\succ_{rlex}}(G)$.

Notation 2.3.15. *Seien $B(G)$ wie oben und $f \in K[x_1, \ldots, x_n]$ mit*

$$f = \alpha_1 b_1 + \cdots + \alpha_d b_d, \quad \alpha_i \in K \text{ für } 1 \leq i \leq d.$$

Dann bezeichnen wir mit $[f]_{B(G)}$ den Koordinatenvektor $(\alpha_1, \ldots, \alpha_d)^T \in K^d$ und mit $\mathbf{e}_j$ den j-ten Einheitsvektor von K^d.

Sei G die reduzierte Gröbnerbasis eines nulldimensionalen Ideals bzgl. $\succ$ und m ein Monom. Wir beschäftigen uns nun mit einer Vorgehensweise zur Berechnung des Koordinatenvektors $[\overline{m}^G]_{B(G)} \in K^d$ des Polynoms $\overline{m}^G$ bzgl. $B(G)$.

Aus den Definitionen von $in_\succ(G), B(G)$ und $M(G)$ folgt, dass für ein beliebiges Monom $m \in B(G) \cup M(G)$ nur eine der drei folgenden Behauptungen gilt:

$$m \in B(G), \ \ m \in in_\succ(G) \text{ oder } m \in M(G) \backslash in_\succ(G).$$

Im folgenden setzen wir voraus, dass der Koordinatenvektor $[\overline{m'}^G]_{B(G)}$ für alle Monome $m' \prec m$ bereits berechnet wurde. Dann erhält man den Koordinatenvektor $[\overline{m}^G]_{B(G)}$ wie folgt:

1. Ist $m = b_i \in B(G)$, so ist $[\overline{m}^G]_{B(G)} = \mathbf{e}_i \in K^d$.

2. Ist $g = m + \sum_{i=1}^d \alpha_i b_i \in G$ mit $in_\succ(g) = m$, so ist
$$[\overline{m}^G]_{B(G)} = (-\alpha_1, \ldots, -\alpha_d)^T \in K^d.$$

3. Ist $m = x_i \cdot m'$ für ein $m' \notin B(G)$, so ist $m' \prec m$ und $[\overline{m'}^G]_{B(G)}$ bereitgestellt. Sei $m' = f + \sum_{j=1}^d \lambda_j b_j$ für ein $f \in \langle G \rangle$, $\lambda_j \in K$ und $b_j \in B(G)$ für alle $j = 1, \ldots, d$. Dann ergibt sich
$$m = x_i f + \sum_{j=1}^d \lambda_j x_i b_j \text{ und } \overline{m}^G = \sum_{j=1}^d \lambda_j \overline{x_i b_j}^G.$$
In diesem Fall sind $x_i b_j \prec m$ für alle $\lambda_j \neq 0$[3] und $[\overline{x_i b_j}^G]_{B(G)}$ wurde ausgerechnet.
Daraus folgt
$$[\overline{m}^G]_{B(G)} = \sum_{j=1}^d \lambda_j [\overline{x_i b_j}^G]_{B(G)}.$$

[3] Ist $x_i b_j = m$ für ein $\lambda_j \neq 0$, so ist $m' = b_j \in B(G)$. Dies ist ein Widerspruch zur Tatsache, dass $m' \notin B(G)$ ist.

Auf dieser Idee basiert der Algorithmus 2.3.16 zur Berechnung der entsprechenden Matrizen M_i des Automorphismus

$$\begin{array}{rccc} \phi_i & : & K[x_1,\ldots,x_n]/\langle G\rangle & \longrightarrow & K[x_1,\ldots,x_n]/\langle G\rangle \\ & & m+\langle G\rangle & \longmapsto & \overline{x_i\cdot m}^G+\langle G\rangle \end{array}$$

für $i=1,\ldots,n$.

Wie wir später sehen werden, spielt Algorithmus 2.3.16 zur Beschleunigung des FGLM-Algorithmus (siehe Algorithmus 2.3.18) eine große Rolle.

Beispiel 2.3.17. *Seien G, I und $\succ_{rlex}$ wie Beispiel 2.3.14. Mit Hilfe des Algorithmus 2.3.16 berechnen wir die Matrizen M_i des Automorphismus*

$$\begin{array}{rccc} \phi_i & : & K[x_1,x_2]/\langle G\rangle & \longrightarrow & K[x_1,x_2]/\langle G\rangle \\ & & m+\langle G\rangle & \longmapsto & \overline{x_i\cdot m}^G+\langle G\rangle \end{array}$$

für alle $i\in\{1,2\}$ mit $x_1:=x$ und $x_2:=y$.

Seien $g_1:=xy+y^2$, $g_2:=y^3-x^2$ und $g_3:=x^3+x^2$. Aus G ergibt sich

$$in_{\succ_{rlex}}(G)=\{xy,y^3,x^3\} \text{ und } B(G)=\{1,y,x,y^2,x^2\}.$$

Seien $b_1:=1$, $b_2:=y$, $b_3:=x$, $b_4:=y^2$ und $b_5:=x^2$.
Seien M_1,M_2 zwei 5×5 Nullmatrizen, $L:=\emptyset$ und $m:=1$. Wir betrachten den Algorithmus am Ende des k-ten Durchlaufs ($1\leq k\leq 10$):

k=1: *Man hat $m=y$ und $L=\{x\}$.*

k=2: *Da $b_2=y=x_2\cdot b_1$ ist, bleibt M_1 unverändert und man hat*

$$M_2=[\mathbf{e}_2\ 0\ 0\ 0\ 0],\ m=x \text{ und } L=\{xy,y^2\}.$$

k=3: *Da $b_3=x=x_1\cdot b_1$ ist, bleibt M_2 unverändert und man hat*

$$M_1=[\mathbf{e}_3\ 0\ 0\ 0\ 0],\ m=y^2 \text{ und } L=\{xy,x^2\}.$$

k=4: *Da $b_4=y^2=x_2\cdot b_2$ ist, bleibt M_1 unverändert und man hat*

$$M_2=[\mathbf{e}_2\ \mathbf{e}_4\ 0\ 0\ 0],\ m=xy \text{ und } L=\{x^2,y^3,xy^2\}.$$

k=5: *Da $xy=in_{\succ_{rlex}}(g_1)$ gilt, ist*

$$\overline{xy}^G=-y^2=-b_4.$$

Da $xy=x_1\cdot b_2=x_2\cdot b_3$ gilt, hat man

$$M_1=[\mathbf{e}_3\ (-\mathbf{e}_4)\ 0\ 0\ 0],\ M_2=[\mathbf{e}_2\ \mathbf{e}_4\ (-\mathbf{e}_4)\ 0\ 0],$$
$$m=x^2 \text{ und } L=\{y^3,xy^2\}.$$

k=6: *Da $b_5=x^2=x_1\cdot b_3$ ist, bleibt M_2 unverändert und man hat*

$$M_1=[\mathbf{e}_3\ (-\mathbf{e}_4)\ \mathbf{e}_5\ 0\ 0],\ m=y^3 \text{ und } L=\{xy^2,x^2y,x^3\}.$$

Algorithmus 2.3.16. ***Matphi***(G) :

Eingabe: *Die reduzierte Gröbnerbasis G des nulldimensionalen Ideals $\langle G\rangle \subset K[x_1,\ldots,x_n]$ bzgl. $\succ$.*

Ausgabe: *Für $1 \le i \le n$ die Matrix M_i des Automorphismus ϕ_i von $K[x_1,\ldots,x_n]/\langle G\rangle$ mit $\phi_i(m+\langle G\rangle) = \overline{x_i \cdot m}^G + \langle G\rangle$.*

Initialisierung:

$B(G) := \{b_1,\ldots,b_d\}$ *mit* $b_d \succ b_{d-1} \succ \cdots \succ b_1$;
(die geordnete Menge der Standardmonome bzgl. G)

$M_i := \mathbf{0}_{d\times d},\ \forall 1 \le i \le n$;

$v_j := \mathbf{0}_{d\times 1},\ \forall 0 \le j \le d$;

$L := \emptyset; \quad m := 1;$

while $(m \neq 0)$ ***do***

1. ***if*** $(\exists g = m + \sum_{l=1}^{d} \beta_l b_l \in G$ *mit* $in_\succ(g) = m)$ ***then***

   $v_0 := (-\beta_1,\ldots,-\beta_d)^T$;

2. ***else***

   (a) ***if*** $(\exists m' \in in_\succ(G)$ *mit* $m = x_i \cdot m'$ *für ein* $1 \le i \le n)$ ***then***

   $v_0 := \sum_{j=1}^{d} \lambda_j [\overline{x_i \cdot b_j}^G]_{B(G)}$, *wobei* $\sum_{j=1}^{d} \lambda_j b_j = \overline{m'}^G$ *ist*;

   (b) ***else***

   i. ***if*** $(m = b_t$ *für ein* $1 \le t \le d)$ ***then*** $v_0 := \mathbf{e}_t$;

   ii. $L := L \cup \{x_i \cdot m : 1 \le i \le n\}$;

3. ***while*** $(1 \le i \le n$ *und* $1 \le k \le d)$ ***do***

   if $(m = x_i \cdot b_k)$ ***then***

   i. $v_k := v_0$;

   ii. $M_i := [v_1,\ldots,v_k,\ldots,v_d]$;

4. ***if*** $(L \neq \emptyset)$ ***then***

   (a) $m := min_\succ\{p : p \in L\}$;

   (b) $L := L \setminus \{m\}$;

5. ***else*** $m := 0$;

return$(M_1, M_2, \ldots, M_n)$.

Algorithmus 2.3.16: *Matphi-Algorithmus.*

k=7: *Da* $y^3 = in_{\succ_{rlex}}(g_2)$ *und* $y^3 = x_2 \cdot b_4$ *gilt, ist* $\overline{y^3}^G = x^2 = b_5$. *Dann bleibt* M_1 *unverändert und man hat*

$$M_2 = [\mathbf{e}_2\ \mathbf{e}_4\ (-\mathbf{e}_4)\ \mathbf{e}_5\ 0],\ m = xy^2 \text{ und } L = \{x^2y, x^3\}.$$

k=8: *Da* $xy^2 = y \cdot xy = x_2 \cdot in_{\succ_{rlex}}(g_1)$ *und* $\overline{xy}^G = -y^2$ *ist, erhält man*

$$\overline{xy^2}^G = -\overline{y^3}^G = -x^2.$$

Andererseits gilt $xy^2 = x \cdot y^2 = x_1 \cdot b_4$. *Dann bleibt* M_2 *unverändert und man hat*

$$M_1 = [\mathbf{e}_3\ (-\mathbf{e}_4)\ \mathbf{e}_5\ (-\mathbf{e}_5)\ 0],\ m = x^2y \text{ und } L = \{x^3\}.$$

k=9: *Da* $x^2y = x \cdot xy = x_1 \cdot in_{\succ_{rlex}}(g_1)$ *und* $\overline{xy}^G = -y^2$ *ist, erhält man*

$$\overline{x^2y}^G = -\overline{xy^2}^G = x^2.$$

Andererseits gilt $x^2y = y \cdot x^2 = x_2 \cdot b_5$. *Dann bleibt* M_1 *unverändert und man hat*

$$M_2 = [\mathbf{e}_2\ \mathbf{e}_4\ (-\mathbf{e}_4)\ \mathbf{e}_5\ \mathbf{e}_5],\ m = x^3 \text{ und } L = \emptyset.$$

k=10: *Da* $x^3 = in_{\succ_{rlex}}(g_3)$ *und* $x^3 = x_1 \cdot b_5$ *gilt, ist* $\overline{x^3}^G = -x^2 = -b_5$. *Dann bleibt* M_2 *unverändert und man hat*

$$M_1 = [\mathbf{e}_3\ (-\mathbf{e}_4)\ \mathbf{e}_5\ (-\mathbf{e}_5)\ (-\mathbf{e}_5)] \text{ und } m = 0.$$

Deshalb terminiert der Algorithmus und liefert die folgenden Matrizen zurück:

$$M_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \end{pmatrix} \text{ und } M_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{2.4}$$

Nun beschäftigen wir uns mit dem FGLM-Algorithmus. Gegeben seien die reduzierte Gröbnerbasis G_1 eines nulldimensionalen Ideals $I \subset K[x_1, \ldots, x_n]$ bzgl. einer monomialen Ordnung $\succ_1$ und $B(G_1)$ die endliche Menge der Standardmonome bzgl. G_1. Mit Hilfe des FGLM-Algorithmus kann man eine Gröbnerbasis G_2 von I bzgl. einer anderen monomialen Ordnung $\succ_2$ berechnen.

Um ein Gröbnerbasiselement von G_2 zu erreichen, benötigt man grob gesprochen die in einer Gleichung der endlich vielen Monomen enthaltenen Informationen, etwa

$$\overline{m}^{G_1} + \alpha_1\ \overline{b'_1}^{G_1} + \cdots + \alpha_s\ \overline{b'_s}^{G_1} = 0. \tag{2.5}$$

Hierin bezeichnen $b'_1, \ldots, b'_s$ die berechneten Standardmonome von I bzgl. $\succ_2$ mit $m \succ_2 b'_s \cdots \succ_2 b'_2 \succ_2 b'_1$. Um die Lösbarkeit dieser Gleichung zu testen, kann man die folgende Vektorgleichung betrachten:

$$[\overline{m}^{G_1}]_{B(G_1)} + \alpha_1\ [\overline{b'_1}^{G_1}]_{B(G_1)} + \cdots + \alpha_s\ [\overline{b'_s}^{G_1}]_{B(G_1)} = 0. \tag{2.6}$$

Falls diese Gleichung eine Lösung hat, ist das Polynom

$$g := m + \alpha_1\, b'_1 + \cdots + \alpha_s\, b'_s$$

ein neues Gröbnerbasiselement von G_2.

Der FGLM-Algorithmus benötigt endlich viele Durchläufe einer while-Schleife, in denen drei Mengen betrachtet werden müssen, nämlich G_2, $B(G_2)$ und $M(G_2)$. Zunächst setzen wir $G_2 = B(G_2) = M(G_2) := \emptyset$. Da I ein nulldimensionales Ideal ist, erhalten wir $B(G_2) = \{1\}$ am Ende des ersten Durchlaufs der Schleife.

Im zweiten Durchlauf definieren wir das Monom

$$m := \min_{\succ_2}\{x_j : j = 1, \ldots, n\}$$

und betrachten die Vektorgleichung

$$[\overline{m}^{G_1}]_{B(G_1)} + \alpha\, [\overline{1}^{G_1}]_{B(G_1)} = \mathbf{0} \tag{2.7}$$

in der Unbestimmten $\alpha \in K$. Es sind zwei Fälle zu unterscheiden:

(i) Falls die Gleichung (2.7) eine Lösung hat, ist das Polynom $g_1 := m + \alpha \in I$. Also wird g_1 zu G_2 und m zu $M(G_2)$ hinzugefügt. Die Menge $B(G_2)$ bleibt unverändert.

(ii) Falls die Gleichung (2.7) keine Lösung hat, ist $m \notin in_{\succ_2}(I)$ (siehe Fußnote 6). Das heißt, m ist ein Standardmonom von I bzgl. $\succ_2$. Deswegen muss m in $B(G_2)$ hinzugefügt werden und G_2 sowie $M(G_2)$ bleiben unverändert.

Wir betrachten nun die Situation im i-ten Durchlauf der Schleife, in dem die drei folgenden Mengen zur Verfügung stehen:

1. $G_2 := \{g_1, g_2, \ldots, g_{t_i}\}$ mit $in_{\succ_2}(g_{t_i}) \succ_2 in_{\succ_2}(g_{t_i-1}) \succ_2 \cdots \succ_2 in_{\succ_2}(g_1)$ und für jedes $1 \le s, t \le t_i$, $s \ne t$ wird $in_{\succ_2}(g_s)$ nicht durch $in_{\succ_2}(g_t)$ geteilt,
2. $B(G_2) := \{b'_1, \ldots, b'_{d_i}\}$[4] mit $b'_{d_i} \succ_2 b'_{d_i-1} \succ_2 \cdots \succ_2 b'_1$ und
3. $M(G_2) := \{m_1, \ldots, m_{s_i}\}$[5] mit der Bedingung: Für jedes $m \in M(G_2)$ existiert $g \in G_2$, so dass m durch $in_{\succ_2}(g)$ geteilt wird.

In diesem Durchlauf beginnt der Algorithmus mit dem Monom

$$m := \min_{\succ_2}\{x_i \cdot b' \;:\; b' \in B(G_2),\; x_i \cdot b' \notin B(G_2) \cup M(G_2),\; 1 \le i \le n\}. \tag{2.8}$$

Falls m durch $in_{\succ_2}(g)$ für ein $g \in G_2$ geteilt wird, setzen wir

$$M(G_2) := \{m_1, \ldots, m_{s_i}, m_{s_i+1}\} \text{ mit } m_{s_i+1} := m.$$

Die Mengen $B(G_2)$ sowie G_2 bleiben unverändert. Dann verlässt der Algorithmus den i-ten Durchlauf der Schleife und geht in den $(i+1)$-ten Durchlauf.

[4] Aus der Konstruktion von $B(G_2)$ folgt: Ist m ein Monom mit $b'_{d_i} \succ_2 m$ und m wird nicht durch einen der Initialterme $in_{\succ_2}(g_1), \ldots, in_{\succ_2}(g_{t_i})$ geteilt, so ist $m \in B(G_2)$.

[5] Im Algorithmus werden die berechneten echten Vielfachen des Initialterms eines Polynoms in G_2 nicht in $M(G_2)$ hinzugefügt. Deshalb erhält man nur $M(G_2) = in_{\succ_2}(G)$.

Ansonsten betrachten wir die Vektorgleichung

$$[\overline{m}^{G_1}]_{B(G_1)} + \sum_{k=1}^{d_i} \alpha_k \, [\overline{b'_k}^{G_1}]_{B(G_1)} = \mathbf{0} \tag{2.9}$$

in den Unbestimmten $\alpha_k \in K$ für $k = 1, \ldots, d_i$.

Fall 1. Besitzt die Gleichung (2.9) eine Lösung, so auch die Gleichung

$$\overline{m}^{G} + \sum_{k=1}^{d_i} \alpha_k \, \overline{b'_k}^{G} = 0.$$

Daraus folgt

$$g_{t_i+1} := m + \sum_{k=1}^{d_i} \alpha_k \, b'_k \in I \quad \text{mit } in_{\succ_2}(g_{t_i+1}) = m.$$

Dieses Polynom bzw. m muss in G_2 bzw. $M(G_2)$ hinzugefügt werden. Also erhalten wir am Ende des i-ten Durchlaufs der Schleife

$$\begin{aligned} G_2 &:= \{g_1, \ldots, g_{t_i}, g_{t_i+1}\}, \\ M(G_2) &:= \{m_1, \ldots, m_{s_i}, m_{s_i+1}\} \text{ mit } m_{s_i+1} := m. \end{aligned}$$

$B(G_2)$ bleibt unverändert. Dann verlässt der Algorithmus den i-ten Durchlauf der Schleife und geht in den $(i+1)$-ten Durchlauf.

Fall 2. Besitzt die Gleichung (2.9) keine Lösung, so ist m ein Standardmonom[6] von I bzgl. $\succ_2$. Deswegen muss m in $B(G_2)$ hinzugefügt werden. Also erhalten wir am Ende des i-ten Durchlaufs der Schleife

$$B(G_2) := \{b'_1, \ldots, b'_{d_i}, b'_{d_i+1}\} \text{ mit } b'_{d_i+1} := m.$$

Die Mengen G_2 sowie $M(G_2)$ bleiben unverändert. Dann verlässt der Algorithmus den i-ten Durchlauf der Schleife und geht in den $(i+1)$-ten Durchlauf.

Der Algorithmus terminiert, wenn kein durch Gleichung (2.8) mit den aktuellen Mengen $B(G_2)$ und $M(G_2)$ definiertes Monom m mehr zur Verfügung steht. Dies kommt vor, weil die Menge $B(G_2)$ der Standardmonome des nulldimensionalen Ideals I bzgl. $\succ_2$ endlich ist. Der Beweis der Korrektheit des Algorithmus ist in Faugère et al. (1993) zu finden.

---

[6] **Annahme**: $m \in in_{\succ_2}(I)$. Es gibt $h \in I$ mit $in_{\succ_2}(h) | m$. Daraus folgt $m = q + r$ für ein $q \in I$ und $r \in K[x_1, \ldots, x_n]$, so dass $in_{\succ_2}(m) \succ_2 r$ und r eine Linearkombination der zu $B(G_2)$ gehörenden Standardmonome von I bzgl. $\succ_2$ ist. Sei $r = \sum_{k=1}^{d_i} \lambda_k b'_k$ mit $\lambda_k \in K$ für $1 \leq k \leq d_i$. Also gilt $\overline{m - r}^{G_1} = \overline{m - \sum_{k=1}^{d_i} \lambda_k b'_k}^{G_1} = 0$. Daraus folgt

$$[\overline{m}^{G_1}]_{B(G_1)} - \sum_{k=1}^{d_i} \lambda_k [\overline{b'_k}^{G_1}]_{B(G_1)} = \mathbf{0}.$$

Dies ist ein Widerspruch zur Unlösbarkeit von Gleichung (2.9).

Algorithmus 2.3.18. ***FGLM***$(I, \succ_2)$;

Eingabe: *Ein nulldimensionales Ideal* $I \subset R := K[x_1, \ldots, x_n]$ *und eine monomiale Ordnung* $\succ_2$ *auf* R.

Ausgabe: *Die reduzierte Gröbnerbasis* G_2 *von* I *bzgl.* $\succ_2$.

Initialisierung:

$G_1 :=$ *die reduzierte Gröbnerbasis von* I *bzgl.* $\succ_1$;

$B(G_1) :=$ *die geordnete Menge der Standardmonome von* I *bzgl.* $\succ_1$;

$G_2 = B(G_2) = M(G_2) = L := \emptyset$;

$m := 1$;

while $(m \neq 0)$ ***do***

1. ***if*** (m *ist kein echtes Vielfaches eines Polynoms in* $M(G_2)$) ***then***

   (a) ***if*** (*Für jedes* $b' \in B(G_2)$ *existiert* $\alpha_{b'} \in K$, *so dass gilt:*

   $$[\overline{m}^{G_1}]_{B(G_1)} + \sum_{b' \in B(G_2)} \alpha_{b'} [\overline{b'}^{G_1}]_{B(G_1)} = \mathbf{0})$$

   then

   i. $g := m + \sum_{b' \in B(G_2)} \alpha_{b'} b'$;
   ii. $G_2 := G_2 \cup \{g\}$;
   iii. $M(G_2) := M(G_2) \cup \{m\}$;

   (b) ***else***

   i. $B(G_2) := B(G_2) \cup \{m\}$;
   ii. $L := L \cup \{x_i \cdot m : 1 \leq i \leq n\}$;

2. ***if*** $(L \neq \emptyset)$ ***then***

   (a) $m := \min_{\succ_2}\{h : h \in L\}$;
   (b) $L := L \setminus \{m\}$;

3. ***else*** $m := 0$;

return(G_2).

Algorithmus 2.3.18: *FGLM Algorithmus*

Beispiel 2.3.19. *Sei $I := \langle G \rangle$ wie Beispiel 2.3.14. Die Berechnung der reduzierten Gröbnerbasis von I bzgl. der lexikographischen Ordnung $\succ_{lex}$ mit $x > y$ erfolgt mit dem FGLM-Algorithmus wie folgt:*

Sei $G_1 := G$. Dann ist $B(G_1) = \{1, y, x, y^2, x^2\}$. Im Beispiel 2.3.14 erhält man die folgenden Koordinatenvektoren:

$$\begin{array}{lll} [\overline{1}^{G_1}]_{B(G_1)} = \boldsymbol{e}_1, & [\overline{y}^{G_1}]_{B(G_1)} = \boldsymbol{e}_2, & [\overline{x}^{G_1}]_{B(G_1)} = \boldsymbol{e}_3, \\ [\overline{y^2}^{G_1}]_{B(G_1)} = \boldsymbol{e}_4, & [\overline{xy}^{G_1}]_{B(G_1)} = -\boldsymbol{e}_4, & [\overline{x^2}^{G_1}]_{B(G_1)} = \boldsymbol{e}_5, \\ [\overline{y^3}^{G_1}]_{B(G_1)} = \boldsymbol{e}_5, & [\overline{xy^2}^{G_1}]_{B(G_1)} = -\boldsymbol{e}_5, & [\overline{x^2y}^{G_1}]_{B(G_1)} = \boldsymbol{e}_5, \\ [\overline{x^3}^{G_1}]_{B(G_1)} = -\boldsymbol{e}_5. & & \end{array}$$

Seien $G_2 = B(G_2) = M(G_2) = L := \emptyset$ und $m := 1$. Wir betrachten die Situation des Algorithmus am Ende des k-ten Durchlaufs $(1 \leq k \leq 8)$:

k=1: *Man erhält $B(G_2) = \{1\}$, $m = y$ und $L = \{x\}$.*

k=2: *Da $\mathbf{e}_2 + \alpha_1 \mathbf{e}_1 = 0$ keine Lösung besitzt, hat man $B(G_2) = \{1, y\}$, $m = y^2$ und $L = \{x, xy\}$.*

k=3: *Man hat*

$$B(G_2) = \{1, y, y^2\},\ m = y^3 \text{ und } L = \{x, xy, xy^2\},$$

da $\mathbf{e}_4 + \alpha_1 \mathbf{e}_1 + \alpha_y \mathbf{e}_2 = 0$ keine Lösung besitzt.

k=4: *Man hat*

$$B(G_2) = \{1, y, y^2, y^3\},\ m = y^4 \text{ und } L = \{x, xy, xy^2, xy^3\},$$

da $\mathbf{e}_5 + \alpha_1 \mathbf{e}_1 + \alpha_y \mathbf{e}_2 + \alpha_{(y^2)} \mathbf{e}_4 = 0$ keine Lösung besitzt.

k=5: *Da*

$$\mathbf{e}_5 + \alpha_1 \mathbf{e}_1 + \alpha_y \mathbf{e}_2 + \alpha_{(y^2)} \mathbf{e}_4 + \alpha_{(y^3)} \mathbf{e}_5 = 0$$

eine Lösung besitzt, nämlich $\alpha_{(y^3)} = -1$ und 0 für die anderen Koeffizienten, bleibt $B(G_2)$ unverändert und man hat

$$G_2 = \{y^4 - y^3\},\ M(G_2) = \{y^4\},\ m = x \text{ und } L = \emptyset.$$

k=6: *Die Mengen G_2 und $M(G_2)$ bleiben unverändert und ergibt sich*

$$B(G_2) = \{1, y, y^2, y^3, x\},\ m = xy \text{ und } L = \{x^2\},$$

da $\mathbf{e}_3 + \alpha_1 \mathbf{e}_1 + \alpha_y \mathbf{e}_2 + \alpha_{(y^2)} \mathbf{e}_4 + \alpha_{(y^3)} \mathbf{e}_5 = 0$ keine Lösung besitzt.

k=7: *Die Menge $B(G_2)$ bleibt unverändert und man erhält*

$$G_2 = \{y^4 - y^3,\ xy + y^2\},\ M(G_2) = \{y^4, xy\},\ m = x^2 \text{ und } L = \emptyset,$$

da $-\mathbf{e}_4 + \alpha_1 \mathbf{e}_1 + \alpha_y \mathbf{e}_2 + \alpha_{(y^2)} \mathbf{e}_4 + \alpha_{(y^3)} \mathbf{e}_5 + \alpha_x \mathbf{e}_3 = 0$ eine Lösung besitzt.

k=8: *Die Menge $B(G_2)$ bleibt unverändert und man erhält*

$$G_2 = \{y^4 - y^3,\ xy + y^2,\ x^2 - y^3\},\ M(G_2) = \{y^4, xy, x^2\} \text{ und } m = 0,$$

da $\mathbf{e}_5 + \alpha_1 \mathbf{e}_1 + \alpha_y \mathbf{e}_2 + \alpha_{(y^2)} \mathbf{e}_4 + \alpha_{(y^3)} \mathbf{e}_5 + \alpha_x \mathbf{e}_3 = 0$ eine Lösung besitzt.

Wegen $m = 0$ terminiert der Algorithmus und liefert die Menge

$$G_2 = \{y^4 - y^3,\ xy + y^2,\ x^2 - y^3\}$$

als die reduzierte Gröbnerbasis von I bzgl. $\succ_{lex}$ zurück.

Kapitel 3

Gröbnerwalkalgorithmus

In diesem Kapitel wollen wir einen Basiskonvertierungsalgorithmus präsentieren, der auf dem Konzept des Gröbnerfans basiert und als *Gröbnerwalkalgorithmus* bezeichnet wird. Dieser Algorithmus besteht aus endlich vielen ähnlichen Schritten. Im ersten Abschnitt dieses Kapitels erläutern wir einen solchen Schritt. Der Gröbnerwalkalgorithmus selbst wird erst im dritten Abschnitt vorgestellt. Der Algorithmus terminiert, weil die Anzahl der Gröbnerkegel im Gröbnerfan eines Ideals endlich ist. Die wichtigsten Eigenschaften eines Gröbnerkegels werden im zweiten Abschnitt vorgestellt.

Falls nicht anders angegeben, betrachten wir immer Ideale im Polynomring $K[x_1, \ldots, x_n]$ über einem beliebigen Körper K und Gewichtsvektoren in $\mathbb{R}^n_{\geq 0}$. Ab Abschnitt 3.2 betrachten wir nur Gewichtsvektoren in $\mathbb{Q}^n_{\geq 0}$.

3.1 Direkte Basiskonvertierung

Seien $\succ$ und $\gg$ zwei verschiedene monomiale Ordnungen, die einen Gewichtsvektor ω im Sinne der folgenden Definitionen verfeinern. In diesem Abschnitt stellen wir eine Vorgehensweise vor, um die reduzierte Gröbnerbasis eines Ideals I bzgl. $\gg$ zu erreichen, falls die reduzierte Gröbnerbasis von I bzgl. $\succ$ angegeben wird.

Bemerkung und Definition 3.1.1. *Sei $\omega \in \mathbb{R}^n_{\geq 0}$ ein Gewichtsvektor. Wir sagen, eine monomiale Ordnung $\succ$ auf $K[x_1, \ldots, x_n]$* ***verfeinert*** *ω, falls für je zwei Monome s, t aus $K[x_1, \ldots, x_n]$ gilt*

$$\deg_\omega(s) > \deg_\omega(t) \Longrightarrow s \succ t.$$

Ist eine monomiale Ordnung $\succ$ auf $K[x_1, \ldots, x_n]$ gegeben, so definiert man die monomiale Ordnung $\succ_\omega$ auf $K[x_1, \ldots, x_n]$ als

$$s \succ_\omega t :\Longleftrightarrow \deg_\omega(s) > \deg_\omega(t) \text{ oder } (\deg_\omega(s) = \deg_\omega(t) \text{ und } s \succ t)$$

für je zwei Monome s, t aus $K[x_1, \ldots, x_n]$. Dann verfeinert $\succ_\omega$ den Gewichtsvektor ω.

Notation 3.1.2. *Ist $M \subset K[x_1, \ldots, x_n] \backslash \{0\}$ eine nichtleere Menge, so setzen wir*

$$M_\succ := \{in_\succ(f) : f \in M\} \text{ und}$$
$$M_\omega := \{in_\omega(f) : f \in M\}.$$

Ist M ein von Null verschiedenes Ideal, so bezeichnen wir auch mit $\langle M_\succ \rangle$ das Initialideal von M bzgl. $\succ$ und mit $\langle M_\omega \rangle$ das Initialformenideal von M bzgl. ω.

Bemerkung 3.1.3. *Sei $f \in K[x_1, \ldots, x_n] \backslash \{0\}$ und $\langle 0 \rangle \neq I \subset K[x_1, \ldots, x_n]$ ein Ideal. Verfeinert $\succ$ den Gewichtsvektor ω, so gilt:*

(i) $in_\succ(in_\omega(f)) = in_\succ(f)$.

(ii) $\langle \langle I_\omega \rangle_\succ \rangle = \langle I_\succ \rangle$.

Beweis. (i) Das Polynom f läßt sich als

$$f = in_\omega(f) + h$$

schreiben, wobei $\deg_\omega(h) < \deg_\omega(f) = \deg_\omega(in_\omega(f))$ ist. Ist $h = 0$, so ist die Aussage richtig. Sei m ein beliebiges Monom von $in_\omega(f)$ und m_h ein beliebiges Monom von h. Dann ist $\deg_\omega(m) > \deg_\omega(m_h)$. Da $\succ$ den Gewichtsvektor ω verfeinert, ist $m \succ m_h$. Daraus ergibt sich $in_\succ(f) = in_\succ(in_\omega(f))$.

(ii) folgt aus (i). □

Aus Bemerkung 3.1.3 erhält man das folgende Korollar.

Korollar 3.1.4. *Sei G eine Gröbnerbasis von I bzgl. der Ordnung $\succ$, die den Gewichtsvektor ω verfeinert. Dann ist $G_\omega := \{in_\omega(g) : g \in G\}$ eine Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. $\succ$.*

Beweis. Aus Bemerkung 3.1.3 ergibt sich

$$\langle \langle I_\omega \rangle_\succ \rangle = \langle I_\succ \rangle = \langle G_\succ \rangle = \langle (G_\omega)_\succ \rangle.$$

□

Sei $I \subset K[x_1, \ldots, x_n]$ ein beliebiges Ideal und ω ein Gewichtsvektor. Falls das Initialformenideal $\langle I_\omega \rangle$ ein monomiales Ideal ist, existiert eine monomiale Ordnung $\succ$ auf $K[x_1, \ldots, x_n]$, so dass $\langle I_\succ \rangle = \langle I_\omega \rangle$ gilt. In diesem Fall sagen wir, dass der Gewichtsvektor ω die Ordnung $\succ$ für I **repräsentiert**.

Lemma 3.1.5. *Sei G die reduzierte Gröbnerbasis eines Ideals I bzgl. einer monomialen Ordnung $\succ$. Der Gewichtsvektor ω repräsentiert $\succ$ für I genau dann, wenn $in_\omega(g) = in_\succ(g)$ für alle $g \in G$ ist, d.h.*

$$\langle I_\succ \rangle = \langle I_\omega \rangle \Longleftrightarrow in_\omega(g) = in_\succ(g),\ \forall g \in G.$$

Beweis. „ $\Longrightarrow$ “ Zu zeigen $in_\omega(g) = in_\succ(g)$ für alle $g \in G$.

Annahme: Es gibt ein $h \in G$ mit $in_\omega(h) \neq in_\succ(h)$. Da ω die Ordnung $\succ$ repräsentiert und G eine Gröbnerbasis von I bzgl. $\succ$ ist, gilt

$$in_\omega(h) \in \langle I_\omega \rangle = \langle I_\succ \rangle = \langle G_\succ \rangle.$$

Dann existiert ein Polynom $g \in G$, $g \neq h$, dessen Leitterm einen Term von $in_\omega(h)$ teilt. Dies ist ein Widerspruch zur Reduziertheit der Gröbnerbasis G von I.

„ $\Longleftarrow$ “ Sei $in_\omega(g) = in_\succ(g)$ für alle $g \in G$. Wir wollen zeigen, dass $\langle I_\succ \rangle = \langle I_\omega \rangle$ gilt. Wir zeigen zunächst $\langle I_\succ \rangle \subseteq \langle I_\omega \rangle$. Da G die reduzierte Gröbnerbasis von I bzgl. $\succ$ und $in_\omega(g) = in_\succ(g)$ für alle $g \in G$ ist, gilt

$$\langle I_\succ \rangle = \langle G_\succ \rangle = \langle G_\omega \rangle \subseteq \langle I_\omega \rangle.$$

Nun zeigen wir $\langle I_\succ \rangle \supseteq \langle I_\omega \rangle$. Seien $f \in I$ beliebig und m ein Term von $in_\omega(f)$. Wir zeigen, dass $m \in \langle G_\succ \rangle = \langle I_\succ \rangle$. Ist $m = in_\succ(f)$, so sind wir fertig. Andernfalls nutzen wir aus, dass G eine Gröbnerbasis von I bzgl. $\succ$ ist: Es existieren ein Term v_1 und ein Polynom $g_{j_1} \in G$, so dass gilt

$$in_\succ(f) = v_1 \cdot in_\succ(g_{j_1}) = v_1 \cdot in_\omega(g_{j_1}).$$

Wir setzen

$$f_1 := f - v_1 \cdot g_{j_1}.$$

Dann ist $f_1 \in I$ und $in_\succ(f) \succ in_\succ(f_1)$. Da

$$\deg_\omega(f) \geq \deg_\omega(in_\succ(f)) = \deg_\omega(v_1 \cdot in_\omega(g_{j_1})) > \deg_\omega(v_1 \cdot m_g)$$

für jedes von $in_\omega(g_{j_1})$ verschiedene Monom m_g von g_{j_1} ist, ist m auch ein Term von $in_\omega(f_1)$. Fahren wir also so fort, so erhalten wir nach endlich vielen Schritten ein Polynom in I, dessen Initialterm mit m übereinstimmt ($\succ$ ist eine Wohlordnung). Dies zeigt die Behauptung. □

Lemma 3.1.6. *Sei $\{0\} \neq I \subset K[x_1, \ldots, x_n]$ ein Ideal und $\omega \in \mathbb{R}^n_{\geq 0} \backslash \{0\}$ ein Gewichtsvektor. Dann ist $\langle I_\omega \rangle$ ein ω-homogenes Ideal.*

Beweis. Da $\langle I_\omega \rangle$ ein Ideal im noetherschen Ring $K[x_1, \cdots, x_n]$ ist, existieren endlich viele Polynome $h_1, \ldots, h_r \in I$ mit

$$\langle I_\omega \rangle = \langle in_\omega(h_1), \cdots, in_\omega(h_r) \rangle.$$

Jedes $f \in \langle I_\omega \rangle$ läßt sich auf zwei Weisen schreiben: als Linearkombination

$$f = f_1 \cdot in_\omega(h_1) + \cdots + f_r \cdot in_\omega(h_r),$$

mit geeigneten $f_i \in K[x_1, \ldots, x_n]$, und als

$$f = g_1 + \cdots + g_t \quad \text{mit} \quad \deg_\omega(g_i) > \deg_\omega(g_j) \quad \text{für} \quad i < j,$$

mit ω-homogenen $g_j \in K[x_1, \ldots, x_n]$.

Also ist jede ω-homogene Komponente g_j von f eine Linearkombination der $in_\omega(h_i)$, d.h. $g_j \in \langle I_\omega \rangle$ gilt für alle $j = 1, \ldots, t$. □

Lemma 3.1.7. *Seien ω ein Gewichtsvektor und $\succ$ eine monomiale Ordnung. Jede reduzierte Gröbnerbasis G eines ω-homogenen Ideals I bzgl. $\succ$ besteht aus ω-homogenen Polynomen.*

Beweis. **Annahme:** $g \in G$ ist kein ω-homogenes Polynom. Also ist $0 \neq h := g - in_\omega(g) \in I$. Dann gibt es ein $p \in G$, so dass $in_\succ(h)$ teilbar ist durch $in_\succ(p)$. Dies ist ein Widerspruch zur Reduziertheit der Gröbnerbasis G von I bzgl. $\succ$. □

Lemma 3.1.8. *Seien $\succ, \gg$ zwei verschiedene monomiale Ordnungen, die den Gewichtsvektor ω verfeinern. Sei $G = \{g_1, \ldots, g_r\}$ eine Gröbnerbasis von I bzgl. $\succ$ und $M = \{m_1, \ldots, m_s\}$ die reduzierte Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. $\gg$. Für jedes m_i existieren ω-homogene Polynome $h_{i1}, \ldots, h_{ir} \in K[x_1, \ldots, x_n]$ mit*

$$m_i = \sum_{j=1}^{r} h_{ij} \cdot in_\omega(g_j) \quad \textit{und} \quad \deg_\omega(m_i) = \deg_\omega(h_{ij} \cdot in_\omega(g_j))$$

für alle $j \in \{1, \ldots, r\}$ mit $h_{ij} \neq 0$.

Beweis. Nach Korollar 3.1.4 ist

$$G_\omega = \{in_\omega(g_1), \ldots, in_\omega(g_r)\}$$

eine Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. $\succ$. Nach Lemma 3.1.7 ist jedes Polynom m_i ein ω-homogenes Polynom.

Da $M = \{m_1, \ldots, m_s\} \subset \langle I_\omega \rangle$ und G_ω eine Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. $\succ$ ist, existieren endlich viele Polynome $h_{ij} \in K[x_1, \ldots, x_n]$ mit

$$m_i = \sum_{j=1}^{r} h_{ij} \cdot in_\omega(g_j)$$

für alle $i \in \{1, \ldots, s\}$. Lassen wir gegebenenfalls überflüssige Terme weg, so erhalten wir h_{ij} mit den gewünschten Eigenschaften. □

Man beachte, dass man die h_{ij} mit Hilfe von Division mit Rest berechnen kann.

Satz 3.1.9. *Seien $I, \succ, \gg, \omega, G, M, h_{ij}$ wie im Lemma 3.1.8. Ist $F := \{f_1, \ldots, f_s\}$ mit*

$$f_i = \sum_{j=1}^{r} h_{ij}.g_j$$

für alle $i \in \{1, \ldots, s\}$, so ist F eine Gröbnerbasis von I bzgl. $\gg$.

Beweis. Da $G \subset I$ und

$$f_i = \sum_{j=1}^{r} h_{ij}.g_j$$

für alle $f_i \in F$ ist, so gilt $F \subset I$. Es ist ausreichend zu zeigen, dass $\langle F_\gg \rangle = \langle I_\gg \rangle$ gilt.

Da jedes h_{ij} ein ω-homogenes Polynom ist, so ist $in_\omega(h_{ij}) = h_{ij}$ und

$$h_{ij} \cdot in_\omega(g_j) = in_\omega(h_{ij} \cdot g_j).$$

Dann ergibt sich

$$m_i = \sum_{j=1}^{r} h_{ij} \cdot in_\omega(g_j) = \sum_{j=1}^{r} in_\omega(h_{ij} g_j) = in_\omega\left(\sum_{j=1}^{r} h_{ij} g_j\right) = in_\omega(f_i)$$

für alle $i \in \{1, \ldots, s\}$.

Da $\gg$ den Gewichtsvektor ω verfeinert, ergibt sich aus Bemerkung 3.1.3 (i)

$$in_{\gg}(f_i) = in_{\gg}(in_\omega(f_i)) = in_{\gg}(m_i)$$

für alle $i \in \{1, \ldots, s\}$. Daraus folgt, dass $\langle F_{\gg} \rangle = \langle M_{\gg} \rangle$ gilt.

Da M die reduzierte Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. der Ordnung $\gg$ ist, die ω verfeinert, ergibt sich aus Bemerkung 3.1.3 (ii)

$$\langle F_{\gg} \rangle = \langle M_{\gg} \rangle = \langle \langle I_\omega \rangle_{\gg} \rangle = \langle I_{\gg} \rangle.$$

□

Durch die Anwendung von Korollar 3.1.4, Lemma 3.1.8 und Satz 3.1.9 erhalten wir eine Gröbnerbasis $GB(I, \gg)$ eines beliebigen Ideals I bzgl. einer monomialen Ordnung $\gg$, falls

1. die reduzierte Gröbnerbasis $rGB(I, \succ)$ von I bzgl. einer monomialen Ordnung $\succ$ und

2. ein Gewichtsvektor ω, der durch die beiden monomialen Ordnungen $\succ$ und $\gg$ verfeinert wird,

gegeben sind (siehe Abbildung 3.1). Diese Vorgehensweise wird **direkte Basiskonvertierung** des Ideals I von $\succ$ zu $\gg$ genannt und im Gröbnerwalkalgorithmus gebraucht.

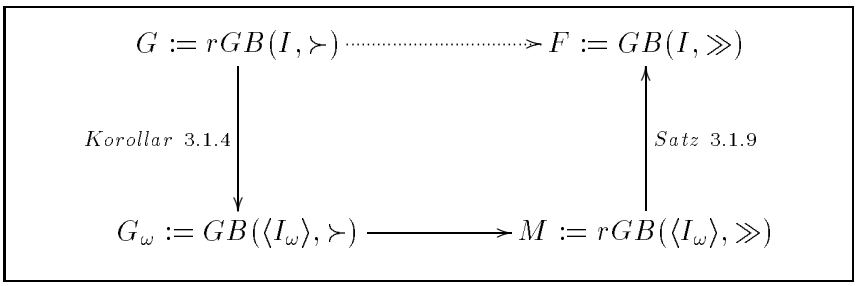

Abbildung 3.1: Direkte Basiskonvertierung des Ideals I von $\succ$ zu $\gg$.

Da $\langle I_\omega \rangle$ ein ω-homogenes Ideal ist, können wir eine Variante des Hilbert-driven Buchbergeralgorithmus benutzen, um M auszurechnen. Dabei benutzen wir die Gröbnerbasis G_ω von $\langle I_\omega \rangle$, um zunächst die "ω-homogene" Hilbert-Funktion von $R/\langle I_\omega \rangle$ zu berechnen.

3.2 Gröbnerkegel

Sei $\succ$ eine monomiale Ordnung auf dem Polynomring $R := K[x_1, \ldots, x_n]$ und I ein Ideal in diesem Ring. Wir bezeichnen die Menge

$$cone(\succ) := cl\{\omega \in \mathbb{R}^n_{\geq 0} : \langle I_\succ \rangle = \langle I_\omega \rangle\},$$

wobei cl der übliche topologische Abschluß in $\mathbb{R}^n$ ist, als **Gröbnerkegel** von I bzgl. $\succ$.

In diesem Abschnitt werden wir Eigenschaften eines solchen Gröbnerkegels untersuchen.

Sei $G = \{g_1, \ldots, g_s\}$ die reduzierte Gröbnerbasis von I bzgl. $\succ$ mit

$$g_i = \mathbf{x}^{\alpha_{i0}} + a_{\alpha_{i1}} \mathbf{x}^{\alpha_{i1}} + \cdots + a_{\alpha_{im_i}} \mathbf{x}^{\alpha_{im_i}}$$

und $in_\succ(g_i) = \mathbf{x}^{\alpha_{i0}}$ für alle $1 \leq i \leq s$.

Nach Lemma 3.1.5 gilt $\langle I_\succ \rangle = \langle I_\omega \rangle$ genau dann wenn $in_\omega(g_i) = in_\succ(g_i)\ \forall i$, d.h. wenn

$$\omega \cdot \alpha_{i0} > \omega \cdot \alpha_{ij},\ \forall i,\ \forall j = 1, \ldots, m_i.$$

Setzen wir

$$D := \{\alpha_{i0} - \alpha_{ij}\ :\ i = 1, \ldots, s,\ j = 1, \ldots, m_i\},$$

so gilt also

$$cone(\succ) = \{\omega \in \mathbb{R}^n_{\geq 0}\ :\ \omega \cdot \alpha_{i0} \geq \omega \cdot \alpha_{ij}, \forall i, \forall j\} = \{\omega \in \mathbb{R}^n_{\geq 0}\ :\ \omega \cdot \delta \geq 0, \delta \in D\}.$$

Aus Lemma 1.5.3 folgt, dass $cone(\succ)$ ein abgeschlossener konvexer Polyederkegel ist, der sich im positiven Orthant $\mathbb{R}^n_{\geq 0}$ befindet. Nach Satz 1.5.7 (ii) ist das Innere von $cone(\succ)$,

$$(cone(\succ))^o := \{\omega \in \mathbb{R}^n_{\geq 0}\ :\ \omega \cdot \alpha_{i0} > \omega \cdot \alpha_{ij}, \forall i, \forall j\},$$

nicht leer.

Dann ist

$$GF(I) = \{cone(\succ)\ : \succ \text{ ist eine monomiale Ordnung auf } R\}$$

der Gröbnerfan von I.

Nach Bemerkung 1.5.6 ist der Gröbnerfan $GF(I)$ von I endlich. Schließlich ergibt sich aus Satz 1.5.7 (i)

$$\bigcup_{cone(\succ)\ \in\ GF(I)} cone(\succ) = \mathbb{R}^n_{\geq 0}.$$

Wegen Lemma 3.1.5 erhalten wir das folgende Lemma.

Lemma 3.2.1. *Seien $\succ, \gg$ zwei verschiedene monomiale Ordnungen und sei G die reduzierte Gröbnerbasis von I bzgl. $\succ$. Es gilt $cone(\succ) = cone(\gg)$ genau dann, wenn $in_\succ(g) = in_\gg(g)$ für alle $g \in G$ ist.*

Beweis. Sei $\omega \in cone(\succ) = cone(\gg)$. Aus Lemma 3.1.5 folgt

$$\langle I_\succ \rangle = \langle I_\omega \rangle = \langle I_\gg \rangle \Longleftrightarrow in_\succ(g) = in_\omega(g) = in_\gg(g),\ \forall g \in G.$$

□

Korollar 3.2.2. *Seien $\succ$ und $\gg$ zwei verschiedene monomiale Ordnungen. Sei G bzw. H die reduzierte Gröbnerbasis von I bzgl. $\succ$ bzw. $\gg$. Es gilt $cone(\succ) = cone(\gg)$ genau dann, wenn $G = H$ ist.*

Lemma 3.2.3. *Seien $\succ$ eine monomiale Ordnung, G die reduzierte Gröbnerbasis von I bzgl. $\succ$ und τ ein Gewichtsvektor. Dann sind die folgenden Bedingungen äquivalent:*

(i) $\tau \in cone(\succ)$.

(ii) *Für jedes $g \in G$ gilt $in_\succ(g) = in_\succ(in_\tau(g))$, d.h. $\succ$ verfeinert τ.*

(iii) *Es gibt eine monomiale Ordnung $\gg$, die τ verfeinert, so dass gilt*

$$cone(\succ) = cone(\gg).$$

Wir nennen $cone(\succ)$ auch Gröbnerkegel von τ, falls $\tau \in cone(\succ)$ gilt.

Beweis. $(i) \Longrightarrow (ii)$ **Annahme:** Es gibt ein Polynom $g \in G$ mit $in_\succ(g) \neq in_\succ(in_\tau(g))$. Dann gilt

$$\deg_\tau(g) > \deg_\tau(in_\succ(g)).$$

Deshalb gibt es eine offene Umgebung U von τ mit

$$\deg_\sigma(g) > \deg_\sigma(in_\succ(g)) \tag{3.1}$$

für alle $\sigma \in U$. Also gilt $in_\succ(g) \neq in_\sigma(g)$ für alle $\sigma \in U$. Nach Lemma 3.1.5 gibt es dann keinen Gewichtsvektor $\sigma \in U$, der die Ordnung $\succ$ für I repräsentiert. Dies ist ein Widerspruch zu $\tau \in cone(\succ)$.

$(ii) \Longrightarrow (iii)$ Da $\succ_\tau$ den Gewichtsvektor τ verfeinert, können wir $\succ_\tau$ als $\gg$ wählen. Dann gilt

$$in_{\succ_\tau}(g) = in_{\succ_\tau}(in_\tau(g)) = in_\succ(in_\tau(g)) = in_\succ(g)$$

für alle $g \in G$, d.h. $cone(\succ_\tau) = cone(\succ)$.

$(iii) \Longrightarrow (i)$ Da $cone(\succ) = cone(\gg)$ und G die reduzierte Gröbnerbasis von I bzgl. $\succ$ ist, folgt aus Korollar 3.2.2, dass G auch die reduzierte Gröbnerbasis von I bzgl. $\gg$ ist. Sei σ ein Gewichtsvektor, der die Ordnung $\gg$ für I repräsentiert. Aus Lemma 3.1.5 ergibt sich

$$in_\sigma(g) = in_\gg(g),\ \forall g \in G. \tag{3.2}$$

Da $\gg$ den Gewichtsvektor τ verfeinert, ergibt sich

$$in_\gg(g) = in_\gg(in_\tau(g)),\ \forall g \in G. \tag{3.3}$$

Seien $\omega := (1-r)\sigma + r\tau \in \overline{\sigma\tau} := \{(1-s)\sigma + s\tau : 1 \leq s \leq 1\}$ ein beliebiger von τ verschiedener Gewichtsvektor auf der Strecke $\overline{\sigma\tau}$ und

$$g = \mathbf{x}^{\alpha} + \sum a_{\beta_i} \mathbf{x}^{\beta_i} \in G$$

beliebig mit $in_{\gg}(g) = \mathbf{x}^{\alpha}$.

Aus (3.2) bzw. (3.3) ergibt sich $\sigma \cdot \alpha > \sigma \cdot \beta_i$ bzw. $\tau \cdot \alpha \geq \tau \cdot \beta_i$ für alle i mit $a_{\beta_i} \neq 0$. Daraus folgt, dass für alle i mit $a_{\beta_i} \neq 0$ gilt

$$\begin{aligned} \omega \cdot \alpha &= ((1-r)\sigma + r\tau) \cdot \alpha = (1-r)(\sigma \cdot \alpha) + r(\tau \cdot \alpha) \\ &> (1-r)(\sigma \cdot \beta_i) + r(\tau \cdot \beta_i) \\ &= ((1-r)\sigma + r\tau) \cdot \beta_i \\ &= \omega \cdot \beta_i. \end{aligned}$$

Also ist $in_{\omega}(g) = \mathbf{x}^{\alpha} = in_{\gg}(g)$ für alle $g \in G$.

Aus Lemma 3.1.5 folgt, dass ω die Ordnung $\gg$ für I repräsentiert, d.h. $\langle I_{\omega} \rangle = \langle I_{\gg} \rangle$. Da diese Aussage für einen beliebigen Gewichtsvektor $\omega \neq \tau$ in $\overline{\sigma\tau}$ richtig ist, gilt $\tau \in cone(\gg) = cone(\succ)$. □

Satz 3.2.4. *Seien σ, τ zwei verschiedene Gewichtsvektoren und $\succ$ eine monomiale Ordnung, die τ verfeinert. Dann existiert $\omega \in \overline{\sigma\tau}$ mit $\omega \neq \sigma$, so dass $\overline{\sigma\omega} \subseteq cone(\succ_{\sigma})$ gilt. Wir nennen ω einen* ***dazwischenliegenden Gewichtsvektor*** *der Strecke $\overline{\sigma\tau}$.*

Beweis. Da $\succ_{\sigma}$ den Gewichtsvektor σ verfeinert, folgt aus Lemma 3.2.3, dass $\sigma \in cone(\succ_{\sigma})$ gilt.

Sei $G := \{g_1, \ldots, g_r\}$ die reduzierte Gröbnerbasis von I bzgl. $\succ_{\sigma}$. Für jedes $g_i \in G$ existiert $h_i \in K[x_1, \ldots, x_n]$ mit $g_i = in_{\sigma}(g_i) + h_i$.

Da

$$\deg_{\sigma}(in_{\sigma}(g_i)) > \deg_{\sigma}(h_i)$$

ist, existiert eine offene Umgebung U von σ mit

$$\deg_{\psi}(u) > \deg_{\psi}(v)$$

für alle $\psi \in U$, für jedes Monom u von $in_{\sigma}(g_i)$ und v von h_i.

Deshalb können wir $\sigma \neq \omega_i \in U$ mit $\omega_i \in \overline{\sigma\tau}$ so wählen, dass für alle $\psi \in \overline{\sigma\omega_i}$

$$\deg_{\psi}(u) > \deg_{\psi}(v)$$

für jedes Monom u von $in_{\sigma}(g_i)$ und v von h_i gilt.

Aus Korollar 3.1.4 folgt, dass $G_{\sigma} = \{in_{\sigma}(g_1), \ldots, in_{\sigma}(g_r)\}$ eine Gröbnerbasis von $\langle I_{\sigma} \rangle$ bzgl. $\succ_{\sigma}$ ist. Da $in_{\succ}(in_{\sigma}(g_i)) = in_{\succ_{\sigma}}(in_{\sigma}(g_i))$ für alle $i = 1, \ldots, r$ gilt, ist G_{σ} eine Gröbnerbasis von $\langle I_{\sigma} \rangle$ bzgl. $\succ$. Da $\succ$ den Gewichtsvektor τ verfeinert, ergibt sich

$$in_{\succ}(in_{\tau}(in_{\sigma}(g_i))) = in_{\succ}(in_{\sigma}(g_i)) = in_{\succ_{\sigma}}(g_i).$$

Also ist $\deg_{\tau}(in_{\sigma}(g_i)) = \deg_{\tau}(in_{\succ_{\sigma}}(g_i))$ für alle $i = 1, \ldots, r$. Mit anderen Worten: Es gilt

$$\deg_{\tau}(u) = \deg_{\tau}(in_{\succ_{\sigma}}(g_i))$$

für jedes Monom u von $in_{\sigma}(g_i)$ und für alle $i = 1, \ldots, r$.

Aus $\psi \in \overline{\sigma\omega_i} \subseteq \overline{\sigma\tau}$ ergibt sich dann

$$\deg_\psi(t) \leq \deg_\psi(in_{\succ_\sigma}(g_i))$$

für jedes Monom t von g_i. Damit gilt $in_{\succ_\sigma}(in_\psi(g_i)) = in_{\succ_\sigma}(g_i)$.

Sei $\omega \in \{\omega_1, \ldots, \omega_r\}$ so gewählt, dass $\overline{\sigma\omega}$ die kürzeste Strecke unter allen $\overline{\sigma\omega_i}$ ist. Dann ist $\sigma \neq \omega \in \overline{\sigma\tau}$ mit

$$in_{\succ_\sigma}(in_\omega(g_i)) = in_{\succ_\sigma}(g_i), \forall g_i \in G.$$

Aus Lemma 3.2.3 folgt $\omega \in cone(\succ_\sigma)$. Da σ und ω im konvexen Gröbnerkegel $cone(\succ_\sigma)$ liegen, gilt $\overline{\sigma\omega} \subseteq cone(\succ_\sigma)$. □

Bemerkung 3.2.5. *(i) In der Praxis arbeitet man nur mit rationalen Gewichtsvektoren. Wir setzen deswegen voraus, dass alle im folgenden betrachteten Gewichtsvektoren rational sind. Entsprechend stellen wir uns vor, dass im folgenden jeder Gröbnerkegel $cone(\succ)$ durch $cone(\succ) \cap \mathbb{Q}^n_{\geq 0}$ ersetzt wird.*

(ii) Seien ω, τ zwei verschiedene Gewichtsvektoren, $\gg$ eine monomiale Ordnung, die τ verfeinert, und G die reduzierte Gröbnerbasis eines Ideals I bzgl. $\gg_\omega$. Für jedes Polynom $g \in G$ sei $M(g)$ die Menge der von $in_{\gg_\omega}(g)$ verschiedenen Terme von g.

Wir setzen

$$\omega(s) := \omega + s(\tau - \omega)$$

sowie

$$t := \min\{s \in \mathbb{Q} \cap (0,1] : \deg_{\omega(s)}(in_{\gg_\omega}(g)) = \deg_{\omega(s)}(m) \text{ für ein } g \in G \\ \text{und ein } m \in M(g) \text{ mit } \deg_\omega(in_{\gg_\omega}(g)) \neq \deg_\omega(m)\}.$$

Existiert dieses Minimum, so setzen wir

$$\omega' := \omega(t) = \omega + t(\tau - \omega).$$

Dann ist $\overline{\omega\omega'}$ das maximale Geradensegment von $\overline{\omega\tau}$ in $cone(\gg_\omega)$. Existiert das Minimum nicht, so sagen wir ω' ist undefiniert. In diesem Fall ist G auch die reduzierte Gröbnerbasis von I bezüglich $\gg$.

(iii) Den auf obigen Überlegungen basierenden Algorithmus haben wir im Computeralgebrasystem SINGULAR *implementiert.* SINGULAR *verarbeitet nur Gewichtsvektoren, deren Komponenten aus nichtnegativen ganzen Zahlen bestehen. Aus diesem Grund ersetzen wir jeden wie in (ii) berechneten Vektor ω' durch ein geeignetes skalares Vielfaches ω'', so dass die Komponenten von ω'' nichtnegative ganze Zahlen sind. Der Gewichtsvektor ω'' liegt dann im gleichen Gröbnerkegel wie ω, aber nicht notwendig auf der Strecke $\overline{\omega\tau}$ (siehe etwa Beispiel 3.3.3). Im nächsten Schritt des Algorithmus geht man auf der Strecke $\overline{\omega''\tau}$ weiter. Insbesondere bei den in Kapitel 4 besprochenen Varianten des Algorithmus haben wir es oft mit Gewichtsvektoren zu tun, deren Komponenten sehr gross sind. Dies ist in unserer Implementierung vor allem deswegen ein Problem, weil in* SINGULAR *jede Komponente eines Gewichtsvektors $< 2^{31}$ sein muss. Auf dieses Problem kommen wir in Kapitel 4 zurück.*

3.3 Gröbnerwalkalgorithmus

Nun wollen wir den **Gröbnerwalkalgorithmus**, mit dem man das folgende Problem lösen kann, präsentieren.

> *Sei G die reduzierte Gröbnerbasis eines beliebigen Ideals I in $K[x_1,\ldots,x_n]$ bzgl. einer monomialen Ordnung $\succ$. Wie rechnet man die reduzierte Gröbnerbasis von I bzgl. einer von $\succ$ verschiedenen monomialen Ordnung $\gg$ ohne eine direkte Berechnung einer solchen Gröbnerbasis aus?*

Für zwei verschiedene monomiale Ordnungen $\succ, \gg$ auf $K[x_1,\ldots,x_n]$ kann man zwei $(n\times n)$ Matrizen A und B mit rationalen Koeffizienten so wählen, dass A bzw. B die monomiale Ordnung $\succ$ bzw. $\gg$ definiert. Ist $\sigma := (\sigma_1,\ldots,\sigma_n)$ bzw. $\tau := (\tau_1,\ldots,\tau_n)$ die erste Zeile von A bzw. B, so verfeinert $\succ$ bzw. $\gg$ den Gewichtsvektor σ bzw. τ. Also gilt $\sigma \in cone(\succ)$ und $\tau \in cone(\gg)$.

Der Gröbnerwalkalgorithmus basiert auf dem folgenden Verfahren: Wir gehen auf der Strecke $\overline{\sigma\tau}$ von σ nach τ. Diese Strecke kann in der Vereinigung von mehreren Gröbnerkegeln liegen.

Sei $\omega_0 := \sigma$. Da $\gg$ den Gewichtsvektor τ verfeinert, können wir nach Bemerkung 3.2.5 einen Gewichtsvektor $\omega_1 \neq \omega_0$ finden, so dass $\overline{\omega_0\omega_1}$ in $cone(\gg_{\omega_0})$ enthalten ist und das maximale Geradensegment von $\overline{\omega_0\tau}$ in $cone(\gg_{\omega_0})$ ist. Gehen wir auf der Strecke $\overline{\omega_1\tau}$ weiter, so kommen wir nach Satz 3.2.4 zunächst in den Gröbnerkegel $cone(\gg_{\omega_1})$.

Fahren wir so fort, so ergibt sich

$$\sigma := \omega_0, \omega_1, \omega_2, \ldots$$
$$cone(\succ) := cone(\gg_{\omega_0}), cone(\gg_{\omega_1}), cone(\gg_{\omega_2}), \ldots$$

mit $\omega_k \in cone(\gg_{\omega_{k-1}}) \cap cone(\gg_{\omega_k})$ für $k = 1, \ldots$.

Da der Gröbnerfan von I endlich ist (siehe Bemerkung 1.5.6), bricht dieses Verfahren nach endlich vielen Schritten ab, wobei der letzte berechnetete Gewichtsvektor entweder gleich τ ist oder undefiniert ist.

Im k-ten Schritt starten wir mit der reduzierten Gröbnerbasis G_{k-1} von I bzgl. $\gg_{\omega_{k-1}}$ und berechnen eine Gröbnerbasis F von I bzgl. $\gg_{\omega_k}$ mit einer direkten Basiskonvertierung wie in Abschnitt 3.1. Dieser Schritt wird in Abbildung 3.2 skizziert.

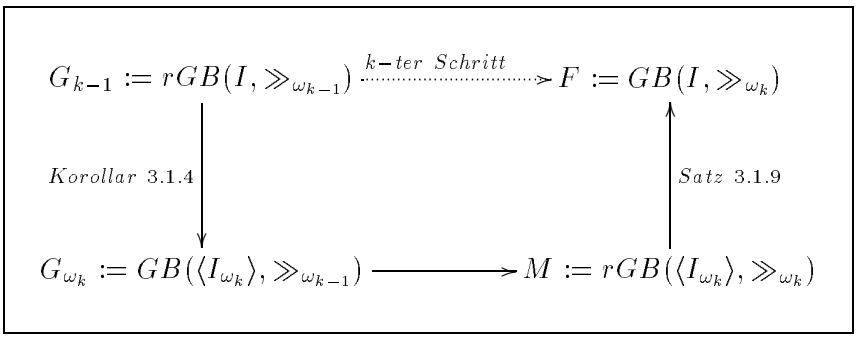

Abbildung 3.2: Der k-te Schritt des Gröbnerwalkalgorithmus.

In unserer Implementierung des Gröbnerwalkalgorithmus werden die folgenden Prozeduren benutzt (zur Beschreibung der Prozeduren verwenden wir die Bezeichnungen aus Lemma 3.1.8, dem Beweis von Lemma 3.1.8 und Satz 3.2.4).

- Die Prozedur **InitialForm**(G, ω) liefert die Menge G_ω, die eine Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. der aktuellen monomialen Ordnung $\succ$ ist.
- Die Prozedur **ReduziertGB**$(G_\omega, \succ)$ liefert die reduzierte Gröbnerbasis des von G_ω erzeugten ω-homogenen Ideals $\langle G_\omega \rangle$ bzgl. der monomialen Ordnung $\succ$. Eine solche Basis wird durch den Hilbert-driven Algorithmus ausgerechnet.
- Die Prozedur **LiftGB**$(G, G_\omega, M, \gg)$ liefert eine Gröbnerbasis des von G erzeugten Ideals $\langle G \rangle$ bzgl. der monomialen Ordnung $\gg$. Diese Gröbnerbasis wird mit Hilfe des Satzes 3.1.9 ausgerechnet.
- Die Prozedur **Reduzierung**$(F, \succ)$ liefert die reduzierte Gröbnerbasis des von der Gröbnerbasis F erzeugten Ideals $\langle F \rangle$ bzgl. der monomialen Ordenung $\succ$.
- Die Prozedur **NächsterVektor**(G, ω, τ) liefert den Gewichtsvektor ω'', der in Bemerkung 3.2.5 beschrieben wird.

Insgesamt erhalten wir den in Algorithmus 3.3.1 beschriebenen Algorithmus.

Beispiel 3.3.2. *Durch den Gröbnerwalkalgorithmus werden wir die reduzierte Gröbnerbasis vom Ideal*

$$I := \langle x^2 - y^3,\, xy + xz,\, x^2y + z^2x \rangle \subset \mathbb{R}[x, y, z]$$

bzgl. der lexikographischen Ordnung $\succ_{lex}$ *auf* $\mathbb{R}[x, y, z]$ *mit* $x > y > z$ *ausrechnen.*

Wir geben die reduzierte Gröbnerbasis von I *bzgl. einer anderen monomialen Ordnung, z.B. der Ordnung* $\succ_{rlex}$ *auf* $\mathbb{R}[x, y, z]$ *mit* $x > y > z$ *an:*

$$G := \{xy + xz,\, x^2z - xz^2,\, y^3 - x^2,\, xz^3 + x^3,\, x^4 + x^3\}.$$

Wähle $\sigma := (1, 1, 1), \tau := (1, 0, 0)$ *und* $\omega := (1, 1, 1)$.

Am Ende der ersten Schleife haben wir

$$G_\omega := \{xy + xz, x^2z - xz^2, y^3, xz^3, x^4\}$$

$$M := \{xy + xz, y^3, x^2z - xz^2, xz^3, x^4\}$$

$$F := \{xy + xz, y^3 - x^2, x^2z - xz^2, xz^3 + x^3, x^4 + x^3\}$$

$$G := \{xy + xz, y^3 - x^2, x^2z - xz^2, xz^3 + x^3, x^4 + x^3\}$$

$$\sigma := (1, 1, 1)$$

$$\omega := (3, 2, 2).$$

Am Ende der zweiten Schleife haben wir

$$G_\omega := \{xy + xz, y^3 - x^2, x^2z, xz^3 + x^3, x^4\}$$

$$M := \{xy + xz, x^2 - y^3, y^3z, y^4, xz^4\}$$

Algorithmus 3.3.1. ***GröbnerWalk***$(G, \succ, \gg)$;

Eingabe: *Zwei verschiedene monomiale Ordnungen* $\succ, \gg$; *die reduzierte Gröbnerbasis* G *von* I *bzgl.* $\succ$.

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl.* $\gg$.

Initialisierung: • *Wähle zwei Matrizen* A *und* B, *deren erste Zeilen* σ *bzw.* τ *durch* $\succ$ *bzw.* $\gg$ *verfeinert werden;*
• $\omega := \sigma$;

while *(1)* ***do***

1. $G_\omega := InitialForm(G, \omega)$;
2. $M := ReduziertGB(G_\omega, \gg_\omega)$;
3. $F := LiftGB(G, G_\omega, M, \gg_\omega)$;
4. $G := Reduzierung(F, \gg_\omega)$;
5. ***if*** $(\omega == \tau)$ ***then***
   return(G);
6. $\sigma := \omega$;
7. $\omega := NächsterVektor(G, \sigma, \tau)$;
8. ***if*** $(\omega$ *ist undefiniert)* ***then***
   return(G);

end

Algorithmus 3.3.1: *Gröbnerwalkalgorithmus.*

$$F := \{xy + xz, -y^3 + x^2, y^3z - xz^2, y^4 + xz^2, xz^4 + x^2z^2\}$$
$$G := \{xy + xz, x^2 - y^3, y^3z - xz^2, y^4 + xz^2, xz^4 + xz^3\}$$
$$\sigma := (3, 2, 2)$$
$$\omega := (2, 1, 1).$$

Am Ende der dritten Schleife haben wir

$$G_\omega := \{xy + xz, x^2, y^3z - xz^2, y^4 + xz^2, xz^4\}$$
$$M := \{xy + xz, y^4 + y^3z, xz^2 - y^3z, x^2, y^3z^3\}$$
$$F := \{xy + xz, y^4 + y^3z, -y^3z + xz^2, x^2 - y^3, y^3z^3 + xz^3\}$$
$$G := \{xy + xz, y^4 + y^3z, xz^2 - y^3z, x^2 - y^3, y^3z^3 + y^3z^2\}$$
$$\sigma := (2, 1, 1)$$

$$\omega := (1,0,0).$$

Am Ende der vierten Schleife haben wir

$$G_\omega := \{xy + xz, y^4 + y^3z, xz^2, x^2, y^3z^3 + y^3z^2\}$$
$$M := \{y^3z^3 + y^3z^2, y^4 + y^3z, xz^2, xy + xz, x^2\}$$
$$F := \{y^3z^3 + y^3z^2, y^4 + y^3z, xz^2 - y^3z, xy + xz, x^2 - y^3\}$$
$$G := \{y^3z^3 + y^3z^2, y^4 + y^3z, xz^2 - y^3z, xy + xz, x^2 - y^3\}.$$

Wegen $\omega = (1,0,0) = \tau$ *verlassen wir die Schleife und erhalten*

$$G := \{y^3z^3 + y^3z^2, y^4 + y^3z, xz^2 - y^3z, xy + xz, x^2 - y^3\},$$

die reduzierte Gröbnerbasis von I *bzgl* $\succ_{lex}$.

Zusammenfassung: *Der Weg des Gröbnerwalkalgorithmus ist wie folgt (siehe Tabelle 3.1): Zuerst geht er auf der Strecke von* $(1,1,1)$ *nach* $(1,0,0)$ *und verlässt den ersten Gröbnerkegel im Punkt* $(3,2,2)$*, um den zweiten Gröbnerkegel zu betreten, d.h. die Strecke von* $(1,1,1)$ *nach* $(3,2,2)$ *liegt im ersten Gröbnerkegel. Dann geht er auf der Strecke von* $(3,2,2)$ *nach* $(1,0,0)$ *und verlässt den zweiten Gröbnerkegel im Punkt* $(2,1,1)$*, um den nächsten Gröbnerkegel zu betreten. Wenn er auf der Strecke von* $(2,1,1)$ *nach* $(1,0,0)$ *geht, wird der Zielgewichtsvektor* $(1,0,0)$ *erreicht. Deswegen terminiert der Gröbnerwalkalgorithmus im nächsten Schritt.*

Kegel	Strecke	"Schnittpunkt"
1.	$(1,1,1) \Longrightarrow (1,0,0)$	$(3,2,2)$
2.	$(3,2,2) \Longrightarrow (1,0,0)$	$(2,1,1)$
3.	$(2,1,1) \Longrightarrow (1,0,0)$	$(1,0,0)$

Tabelle 3.1: Strecke und "Schnittpunkte" des Gröbnerwalkalgorithmus.

Wir wollen nun untersuchen, in welchen Gröbnerkegeln jeder der obigen Schnittpunkte liegt. Der Gröbnerfan von I besteht aus folgenden Gröbnerkegeln (siehe Beispiel 1.5.10 und Abbildung 3.3):

$$D_1^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x - 3y + z \geq 0,\ x \geq \frac{3}{2}y,\ y \geq z\},$$
$$D_2^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : y \geq z,\ x \geq \frac{3}{2}y,\ x - 3y + z \leq 0\},$$
$$D_3^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : \frac{3}{2}z \leq x \leq \frac{3}{2}y,\ y \geq z\},$$
$$D_4^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : z \leq x \leq \frac{3}{2}z,\ x \leq \frac{3}{2}y,\ y \geq z\},$$
$$D_5^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x \leq z \leq y\}.$$
$$D_6^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : x \leq y \leq z\},$$
$$D_7^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : y \leq x \leq \frac{3}{2}y,\ y \leq z\},$$
$$D_8^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : \frac{3}{2}y \leq x \leq y,\ y \leq z\},$$
$$D_9^* = \{(x,y,z) \in \mathbb{R}^3_{\geq 0} : 2y \leq x,\ y \leq z\}.$$

Dann ergibt sich:

$$\begin{aligned}
\omega_0 &:= \sigma = (1,1,1) \in D_4^* \cap D_5^* \cap D_6^* \cap D_7^*, \\
\omega_1 &:= (3,2,2) \in D_2^* \cap D_3^* \cap D_4^* \cap D_7^* \cap D_8^*, \\
\omega_2 &:= (2,1,1) \in D_1^* \cap D_2^* \cap D_8^* \cap D_9^*, \\
\omega_3 &:= \tau = (1,0,0) \in D_1^* \cap D_9^*.
\end{aligned}$$

Die Projektion des Gröbnerfans von I und der Strecke $\overline{\sigma\tau}$, auf der der Gröbnerwalkalgorithmus entlanggeht, auf die Ebene $x+y+z=1$ kann man in der Abbildung 3.3 sehen. □

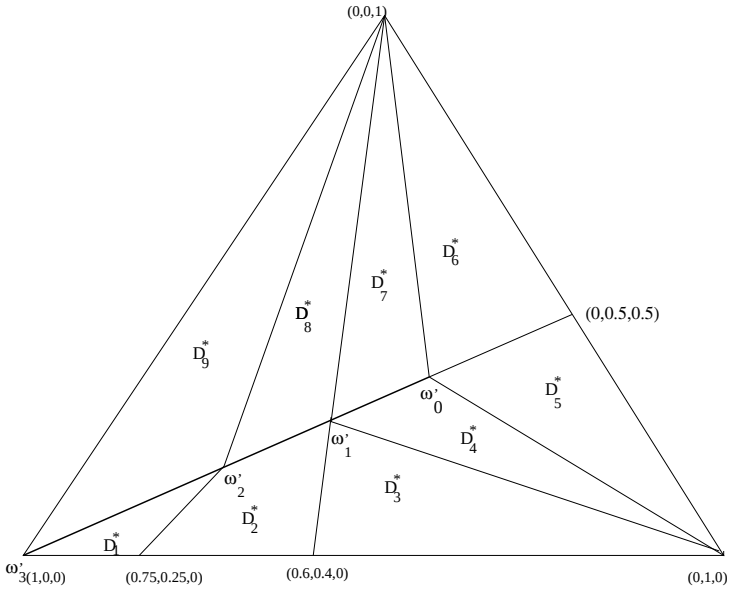

Abbildung 3.3: Die Projektion des Gröbnerfans von I auf der Ebene $x+y+z=1$.

Um die reduzierte Gröbnerbasis eines Ideals I bzgl. einer monomialen Ordnung $\gg$, die einen Gewichtsvektor τ verfeinert, mit Hilfe des Gröbnerwalkalgorithmus zu berechnen, kann man einen beliebigen Startgewichtsvektor σ auswählen. Hierbei wird die reduzierte Gröbnerbasis von I bzgl. der σ-gewichteten monomialen Ordnung $\gg_\sigma$ ausgerechnet und dann als die Eingabe des Gröbnerwalkalgorithmus verwendet.

Beispiel 3.3.3. *Nun betrachten wir wieder das Beispiel 1.5.9. Zur Berechnung der reduzierten Gröbnerbasis des Ideals*

$$I := \langle x^3 - xy^2, y^2 - xy, xy^2 + xy + y \rangle \subset \mathbb{R}[x,y]$$

bzgl. der lexikographischen Ordnung $\succ_{lex}$ auf $\mathbb{R}[x,y]$ mit $x > y$ wenden wir verschiedene Startgewichtsvektoren, z.B. $\sigma_1 = (1,1)$, $\sigma_2 = (1,2)$, $\sigma_3 = (1,3)$, $\sigma_4 = (1,5)$ und $\sigma_5 = (0,5)$ und den Zielgewichtsvektor $\tau = (1,0)$ an. Das heißt, der Gröbnerwalkalgorithmus geht auf verschiedenen Strecken und Gröbnerkegeln entlang.

Fall 1. *Die Strecke $\overline{\sigma_1\tau}$ liegt nur im Gröbnerkegel D_1^*.*

Fall 2. *Die Strecke $\overline{\sigma_2\tau}$ liegt in zwei Gröbnerkegeln D_1^* und D_2^*. Im Punkt $(1,1)$ verlässt diese Strecke den Gröbnerkegel D_2^* und betritt den Gröbnerkegel D_1^*.*

Fall 3. *Die Strecke $\overline{\sigma_3\tau}$ liegt in drei Gröbnerkegeln D_1^*, D_2^* und D_3^*. Im Punkt $(1,2)$ verlässt diese Strecke den Gröbnerkegel D_3^* und betritt den Gröbnerkegel D_2^*. Dann verlässt sie den Gröbnerkegel D_2^* im Punkt $(1,1)$, um den Gröbnerkegel D_i^* zu betreten.*

Fall 4. *Die Strecke $\overline{\sigma_4\tau}$ liegt in allen vier Gröbnerkegeln D_1^*, D_2^*, D_3^* und D_4^*. Die Schnittpunkte sind $(1,4), (1,2)$ und $(1,1)$. Diese Punkte liegen auf der Strecke $\overline{\sigma_4\tau}$.*

Fall 5. *Die Strecke $\overline{\sigma_5\tau}$ liegt in allen vier Gröbnerkegeln D_1^*, D_2^*, D_3^* und D_4^*. Der Gröbnerwalkalgorithmus geht nicht auf der Strecke $\overline{\sigma_5\tau}$, sondern auf den zwei folgenden Strecken:*

(i) die Strecke von $\sigma_5 = (0,5)$ nach $(1,4)$, die in D_4^* liegt, und

(ii) die Strecke von $(1,4)$ nach $\tau = (1,1)$, die in drei Gröbnerkegeln D_1^*, D_2^* und D_3^* liegt (siehe Abbildung 3.4).

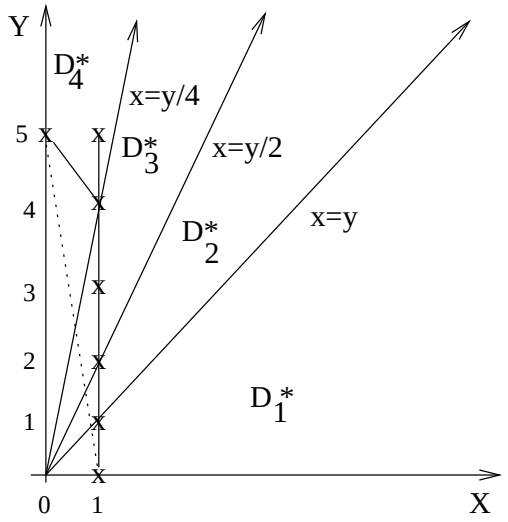

Abbildung 3.4: Strecken $\overline{\sigma_i\tau}$ im Gröbnerfan von I.

Tabelle 3.2 zeigt den Zeitverbrauch des Gröbnerwalkalgorithmus, der in SINGULAR und SACLIB implementiert ist, zur Berechnung der reduzierten lexikographischen Gröbnerbasis eines Ideals im Polynomring $\mathbb{Q}[x_1, \ldots, x_n]$. Dazu haben wir einen Laptop PC Pentium II 266 MHz Prozessor mit 32 Megabyte RAM benutzt. Die Laufzeiten von SACLIB sind nach Amrhein et al. (1996a) zitiert. Sie haben die Beispiele auf einen SPARCstation 10/40 mit 32 Megabyte RAM getestet.

In den meisten Fällen, in denen die reduzierte Gröbnerbasis eines Ideals bzgl. einer lexikographischen monomialen Ordnung ausgerechnet wird, wird mehr als 90% der Laufzeit des Gröbnerwalkalgorithmus im letzten Schritt des Algorithmus gebraucht. Davon entfällt die meiste Zeit auf die Berechnung der reduzierten Gröbnerbasis des τ-homogenen Ideals G_τ mit $\tau = (1, 0, \ldots, 0)$ (siehe z.B. Tabelle 3.3). Dies liegt daran dass der Weg des Gröbnerwalks im letzten Schritt typischerweise durch einen Punkt geht der im Schnitt mehrerer Gröbnerkegel liegt, und deswegen jedes Element von G_ω aus vielen Termen besteht. Aus diesem Grund geht man alternativ oft so vor, dass man einen geeigneten von τ verschiedenen Zielgewichtsvektor τ' wählt. Entsprechende Algorithmen werden im nächsten Kapitel vorgestellt.

Beispiele	n	SINGULAR	SACLIB
bsp1	3	0.02	3.33
bsp2	3	0.02	14.33
bsp3	3	0.25	2.32
bsp4	3	0.06	4.39
trinks1	6	65.45†	54.5
canny	5	0.96	133
canny2	5	4.06	269
s6	6	1.10	3395
bsp5	3	0.08	1544
bsp6	3	1.71	6452
bsp7	3	12.76	1141

Tabelle 3.2: Laufzeiten in Sekunden des Gröbnerwalkalgorithmus (†Im zweiten Schritt benötigt die Berechnung der reduzieten lexikographischen Gröbnerbasis von $\langle G_\tau \rangle$ viel Zeit. G_τ besteht aus einem Monom und 13 Polynomen. Sechs Polynome davon haben zehn Monome, fünf Polynome davon haben elf Monome und die anderen bestehen aus vier bzw. acht Monomen).

Tabelle 3.3[1)] zeigt den maximalen Speicherverbrauch in Kilobyte, die Anzahl der Schritte und den Zeitverbrauch jeder Prozedur im letzten Schritt des Gröbnerwalkalgorithmus.

Beispiele	Speicher	Schritte	Laufzeit (%) zur				Summe
			InFo	sG_ω	Lift	InRd	
arnborg7	898	7	0.00	9.65	85.53	0.44	95.62
trinks1	4630	2	0.00	92.27	7.53	0.03	99.83
canny2	1339	5	0.00	93.41	0.55	0.27	94.23
bsp6	1507	4	0.00	71.01	23.67	1.18	95.86
bsp7	3276	8	0.00	30.67	37.18	30.90	98.75

Tabelle 3.3: Laufzeitanalyse jeder Prozedur im letzten Schritt des Gröberwalkalgorithmus.

[1)] InFo, sG_ω, Lift, bzw. InRd ist die Abkürzung der Prozedur **InitialForm**, **ReduziertGB**, **LiftGB** bzw. **Reduzierung**.

Kapitel 4

Entwicklungen des Gröbnerwalkalgorithmus

In diesem Kapitel präsentieren wir die auf dem Gröbnerwalkalgorithmus basierenden Algorithmen. In den ersten zwei Abschnitten des Kapitels stellen wir die von Amrhein et al. (1996, 1997 und 1998) entwickelten Algorithmen vor, nämlich den Perturbation- und Fractalwalkalgorithmus. Der von Tran (2000) entwickelte Algorithmus wird im dritten Abschnitt präsentiert. Schließlich stellen wir unsere eigenen Algorithmen im letzten Abschnitt vor.

4.1 Perturbationwalkalgorithmus

Gegeben sei die reduzierte Gröbnerbasis eines Ideals I im Polynomring $R := K[x_1, \ldots, x_n]$ über einem beliebigen Körper K bzgl. einer monomialen Ordnung $\succ$ auf R. Zur Berechnung der reduzierten Gröbnerbasis von I bzgl. einer von $\succ$ verschiedenen monomialen Ordnung $\gg$ auf R geht der Gröbnerwalkalgorithmus auf der Strecke $\overline{\sigma\tau}$ entlang, wobei ein Gewichtsvektor σ (bzw. τ) durch die Ordnung $\succ$ (bzw. $\gg$) verfeinert wird.

Falls die reduzierte Gröbnerbasis von I bzgl. einer lexikographischen monomialen Ordnung $\succ_{lex}$ auf R ausgerechnet und der Gewichtsvektor $\tau = (1, 0, \ldots, 0) \in \mathbb{N}^n$ als Zielgewichtsvektor gewählt wird, so muss im letzten Schritt des Gröbnerwalkalgorithmus die reduzierte Gröbnerbasis des τ-homogenen Ideals $\langle G_\tau \rangle$ bzgl. der τ-gewichteten monomialen Ordnung $\succ_{(\tau,lex)} = \succ_{lex}$ durch einen verbesserten Buchbergeralgorithmus, z.B. dem Hilbert-driven-Algorithmus, ausgerechnet werden.

In den meisten Fällen haben die Elemente des Erzeugendensystems G_τ viele Terme, weil τ in mehreren Gröbnerkegeln liegt. Deswegen ist diese Berechnung sehr aufwendig und benötigt viel Zeit und Speicher (siehe z.B. Tabelle 3.8 in Kapitel 3 und Tabelle 4.4 in diesem Kapitel).

Aus Korollar 3.2.2 folgt, dass die reduzierte Gröbnerbasis eines beliebigen Ideals bzgl. monomialer Ordnungen, die im gleichen Gröbnerkegel liegen, gleich ist. Aus diesem Grund kann man einen von τ verschiedenen Gewichtsvektor τ' so wählen, dass τ' und τ im gleichen Gröbnerkegel liegen. Dasselbe kann man auch mit σ machen, indem man einen Gewichtsvektor σ' wählt. So geht der Gröbnerwalkalgorithmus auf einer von $\overline{\sigma\tau}$ verschiedenen Strecke $\overline{\sigma'\tau'}$ entlang. Theoretisch kann man Satz 1.5.7 (ii) anwenden, um σ' und τ' zu bestimmen.

Sei d eine positive ganze Zahl mit $1 \leq d \leq n$ und G die reduzierte Gröbnerbasis des Ideals $\langle G \rangle \subset K[x_1, \ldots, x_n]$ bzgl. einer Ordnung $\succ$, die durch eine Matrix M mit Zeilen $\omega_1, \cdots, \omega_n$ definiert wird. Für alle $1 \leq i \leq n$ sei $\omega_i := (\omega_{i1}, \ldots, \omega_{in}) \in \mathbb{R}^n$ und $\max(\omega_i) := \max\{|\omega_{ij}| \ : \ 1 \leq j \leq n\}$. In der Praxis wählt man eine ganzzahlige Matrix M.

Ist $\epsilon \in \mathbb{R}$ eine kleine positive Zahl, so liegt der Gewichtsvektor

$$\omega := \omega_1 + \epsilon\omega_2 + \epsilon^2\omega_3 + \cdots + \epsilon^{d-1}\omega_d$$

im Gröbnerkegel von ω_1. Man nennt ω einen **gestörten Gewichtsvektor** des Gewichtsvektors ω_1 vom Grad d bzgl. M.

Amrhein et al. (1996) wählten ϵ, so dass $\frac{1}{\epsilon} \in \mathbb{Z}$ ist und

$$\tfrac{1}{\epsilon} > \deg(g)(\max(\omega_2) + \cdots + \max(\omega_d))$$

für alle $g \in G$ gilt. In unseren Implementierungen wählen wir $\omega_i \in \mathbb{Z}^n$ und

$$\frac{1}{\epsilon} := \deg(g)(\max(\omega_2) + \cdots + \max(\omega_d)) + 1. \tag{4.1}$$

Also liegen ω und sein in der Praxis benutztes ganzzahliges Vielfaches

$$\omega' = (\frac{1}{\epsilon})^{d-1} \cdot \omega = (\frac{1}{\epsilon})^{d-1}\omega_1 + (\frac{1}{\epsilon})^{d-2}\omega_2 + \cdots + \omega_d$$

im gleichen Gröbnerkegel. Wir bezeichnen auch ω' als einen gestörten Gewichtsvektor von ω_1 vom Grad d. Der Algorithmus 4.1.1 liefert einen solchen gestörten Gewichtsvektor zurück.

In Algorithmus 4.1.2 muss der Zielgewichtsvektor τ gestört werden. Um ein geignetes ϵ wie oben zu finden braucht man die reduzierte Gröbnerbasis des gegebenen Ideals bzgl. der Ordnung $\gg$, die τ verfeinert. Allerdings liegt eine solche reduzierte Gröbnerbasis am Anfang des Algorithmus nicht vor. Deswegen benutzt man die reduzierte Gröbnerbasis des gleichen Ideals bzgl. $\succ$. Diese Verfahrensweise garantiert jedoch nicht, dass der berechnete gestörte Gewichtsvektor τ' von τ im entsprechenden Gröbnerkegel liegt. Folglich muss am Ende des Algorithmus getestet werden, ob die beiden Gewichtsvektoren τ' und τ im gleichen Gröbnerkegel liegen. Dazu muss man zeigen, dass

$$\tau' \in cone(\gg) \Longleftrightarrow in_{\gg}(g) = in_{\gg}(in_{\tau'}(g)), \ \forall g \in F$$

gilt, wobei F die reduzierte Gröbnerbasis des Ideals $\langle F \rangle$ bzgl. $\gg_{\tau'}$ ist (siehe Lemma 3.2.3).

Liegen die beiden Gewichtsvektoren τ und τ' im gleichen Gröbnerkegel, so terminiert der Algorithmus und liefert die gewünschte Gröbnerbasis zurück. Andernfalls ist F *nicht* die gewünschte Gröbnerbasis. Damit man die gewünschte Gröbnerbasis erreicht, kann man den Buchbergeralgorithmus mit $\gg$ auf das Ideal $\langle F \rangle$ anwenden. Eine andere Möglichkeit ist die folgende: Ist τ' durch eine Störung vom Grad d entstanden, so ersetzen wir an dieser Stelle τ' durch einen neuen Zielgewichtsvektor τ'' mit einer kleineren Störung (etwa vom Grad $(d-1)$), der mit Hilfe der reduzierten Gröbnerbasis F berechnet wird. Dann wenden wir den Gröbnerwalkalgorithmus

Algorithmus 4.1.1. ***PertVektor***(G, M, d);

Eingabe: *Eine ganzzahlige Matrix M wie in Beispiel 1.2.7, die eine monomiale Ordnung $\succ$ definiert, die reduzierte Gröbnerbasis G eines Ideals I in $K[x_1, \ldots, x_n]$ bzgl. $\succ$, und der Perturbationgrad d mit $1 \leq d \leq n$.*

Ausgabe: *Der gestörte Gewichtsvektor ω von ω_1 vom Grad d, wobei ω_1 die erste Zeile von M ist.*

Initialisierung: *Für $i = 1, \ldots, n$ sei $\omega_i := (\omega_{i1}, \ldots, \omega_{in}) \in \mathbb{Z}^n$ die i-te Zeile von M.*

begin

1. $m_i := \max\{|\omega_{ij}| \ : \ 1 \leq j \leq n\}, \ \forall 1 \leq i \leq d$;
2. $m := \sum_{i=2}^{d} m_i$;
3. $e := \max\{\deg(g) : g \in G\}$;
4. $e := me + 1$;
5. $\omega := e^{d-1}\omega_1 + e^{d-2}\omega_2 + \cdots + e\omega_{d-1} + \omega_d$;
6. ***return***(ω);

end.

Algorithmus 4.1.1: *Störungvektoralgorithmus vom Grad d.*

auf das Ideal $\langle F \rangle$ an. Diese Vorgehensweise wird wiederholt, bis die gewünschte reduzierte Gröbnerbasis erreicht ist oder der Grad des gestörten Zielgewichtsvektors gleich 1 ist. Im letzteren Fall muss man dann wieder den Buchbergeralgorithmus heranziehen.

Die Prozedur **GröbnerWalk**$(G, \succ, \gg)$ liefert, wie im letzten Kapitel besprochen, die durch den Gröbnerwalkalgorithmus ausgerechnete reduzierte Gröbnerbasis des Ideals $\langle G \rangle$ bzgl. der Ordnung $\gg$, wobei G die reduzierte Gröbnerbasis von $\langle G \rangle$ bzgl. der Ordnung $\succ$ ist (siehe Algorithmus 3.3.1).

Die Prozedur **RedGB_BA**$(G, \gg)$ liefert die durch den Buchbergeralgorithmus ausgerechnete reduzierte Gröbnerbasis des Ideals $\langle G \rangle$ bzgl. der Ordnung $\gg$.

Den Algorithmus 4.1.2 bezeichnen wir als **Perturbationwalkalgorithmus** vom Perturbationgradpaar (d_1, d_2). Für $d_1 = d_2 = 1$ ist der Perturbationwalkalgorithmus gleich dem Gröbnerwalkalgorithmus.

Wir haben einen Computer mit Pentium IV 2.40 GHz Prozessor und 1 Gegabyte RAM benutzt, um die folgenden Beispiele zu berechnen.

Algorithmus 4.1.2. ***PertWalk***$(G, \succ, \gg, d_1, d_2)$;

Eingabe: *Zwei verschiedene monomiale Ordnungen $\succ$ bzw. $\gg$, die reduzierte Gröbnerbasis G eines Ideals I in $K[x_1, \ldots, x_n]$ bzgl. $\succ$, und zwei positive ganze Zahlen d_1, d_2 mit $1 \leq d_1, d_2 \leq n$.*

Ausgabe: *Die reduzierte Gröbnerbasis F von I bzgl. $\gg$.*

Initialisierung: • *Wähle zwei Matrizen A und B, deren erste Zeilen σ bzw. τ durch $\succ$ bzw. $\gg$ verfeinert werden;*
• $\sigma' := PertVektor(G, A, d_1)$;
• $d := d_2$;

while *(1)* ***do***

1. $\tau' := PertVektor(G, B, d)$;
2. $F :=$*GröbnerWalk*$(G, \gg_{\sigma'}, \gg_{\tau'})$;
3. ***if*** $(\tau' \in cone(F, \gg))$ ***then***
   return(F);
4. ***else***
   return$(RedGB\_BA(F, \gg))$;

end.

Alternativ ersetzen wir den 4. Schritt durch den folgenden Schritt:

4'. ***else***
   (a) ***if*** $(d = 1)$ ***then***
      return$(RedGB\_BA(F, \gg))$;
   (b) $d := d - 1$;
   (c) $G := F$; $\sigma' = \tau'$;

Algorithmus 4.1.2: *Perturbationwalkalgorithmus vom Perturbationgradpaar (d_1, d_2).*

Beispiel 4.1.3. *(**Trinks1** in Tran (1998)) Mit Hilfe des Perturbationwalkalgorithmus von verschiedenen Perturbationgradpaaren berechnen wir die reduzierte Gröbnerbasis vom Ideal*

$$\begin{aligned} I = \langle\, & 45y + 35u - 165v - 36,\ 36y + 25z + 40t - 27u, \\ & 25yu - 165v^2 + 15x - 18z + 30t,\ 15yz + 20tu - 9x, \\ & -11v^3 + xy + 2zt,\ -11uv + 3v^2 + 99x \,\rangle \end{aligned}$$

im Polynomring $\mathbb{Q}[x, y, z, t, u, v]$ bzgl. $\succ_{lex}$ auf $\mathbb{Q}[x, y, z, t, u, v]$ mit

$$x > y > z > t > u > v.$$

Fall 1. Für das Perturbationgradpaar $(1,1)$ geht der Algorithmus auf der Strecke von $(1,1,1,1,1,1)$ nach $(1,0,0,0,0,0)$ entlang. Er braucht zwei Schritte und 13.08 Sekunden, um die reduzierte lexikographische Gröbnerbasis zurückzuliefern.

Fall 2. Für das Perturbationgradpaar $(2,2)$ geht der Algorithmus auf der Strecke von $(4,4,4,4,4,3)$ nach $(4,1,0,0,0,0)$ entlang. Er braucht zwei Schritte und 1.68 Sekunden, um die reduzierte lexikographische Gröbnerbasis zurückzuliefern.

Fall 3. Für das Perturbationgradpaar $(3,3)$ geht der Algorithmus auf der Strecke von $(49,49,49,49,48,42)$ nach $(49,7,1,0,0,0)$ entlang. Er braucht vier Schritte und 6.82 Sekunden, um die reduzierte lexikographische Gröbnerbasis zurückzuliefern.

Fall 4. Für das Perturbationgradpaar $(4,4)$ geht der Algorithmus auf der Strecke von $(1000,1000,1000,999,990,900)$ nach $(1000,100,10,1,0,0)$ entlang. Er braucht acht Schritte und 0.33 Sekunden, um die reduzierte lexikographische Gröbnerbasis zurückzuliefern.

Tabelle 4.1[1)] zeigt die Zusammenfassung der obigen Ergebnisse.

PertWalk mit	Perturbationgradpaar				
	(1,1)	(2,2)	(3,3)	(4,4)	
Variante 4 / Variante 4'	13.08	1.68	6.82	0.33	Zeit
Variante 4 / Variante 4'	2	2	4	8	Schritte
Variante 4 / Variante 4'	4584	1541	3127	805	Speicher

Tabelle 4.1: Berechnungszeiten in Sekunden, die Anzahl der Schritte und der maximale Speicherverbrauch in Kilobyte des Perturbationwalkalgorithmus.

Beispiel 4.1.4. *(**Beispiel 3.3** in Tran (2000)) Durch die Anwendung des Perturbationwalkalgorithmus von verschiedenen Perturbationgradpaaren auf das Ideal*

$$\begin{aligned} I = &\langle\, 16 + 3x^3 + 16x^2z + 14x^2y^3, \\ &\quad 6 + y^3z + 17x^2z^2 + 7xy^2z^2 + 13x^3z^2 \,\rangle \\ &\subset \mathbb{Q}[x,y,z], \end{aligned} \tag{4.2}$$

um die reduzierte Gröbnerbasis von I bzgl. $\succ_{lex}$ auf $\mathbb{Q}[x,y,z]$ mit $x > y > z$ auszurechnen, erhalten wir Tabelle 4.2[2)].

[1)] Für die Perturbationgradpaare $(5,5)$ bzw. $(6,6)$ sind eine oder mehrere Komponenten des letzten berechneten dazwischenliegenden Gewichtsvektors größer als 2^{31}. In diesen Fällen geht der Algorithmus nur bis auf den zehnten (bzw. zweiten) Schritt. Für die anderen Paare liegen die gestörten Zielgewichtsvektoren im richtigen Gröbnerkegel. Deswegen stimmen die beiden Varianten überein.

[2)] $>$ 212M bedeutet, dass nach 212 Minuten das Programm abbricht, weil kein Speicher mehr zur Verfügung steht, um die reduzierte Gröbnerbasis eines Ideals $\langle F\rangle$ zu berechnen. Bz., Sc. bzw. Sp. ist die Abkürzung für Berechnungszeit, Anzahl der Schritte und maximalen Speicherverbrauch.

PertWalk mit	Perturbationgradpaar (1,1)	(2,2)	(3,3)	
Variante 4	4585.21	>212M	4885.55	Bz.
Variante 4'		35.16	10.96	
Variante 4	16	21	44	Sc.
Variante 4'		21+6	44+8	
Variante 4	231355	∞	165257	Sp.
Variante 4'		21189	7035	

Tabelle 4.2: Berechnungszeiten in Sekunden, die Anzahl der Schritte und der maximale Speicherverbrauch in Kilobyte des Perturbationwalkalgorithmus.

Im letzten Schritt des Perturbationwalkalgorithmus vom Perturbationgradpaar $(2,2)$ und $(3,3)$ liegt der gestörte Zielgewichtsvektor nicht im entsprechenden Gröbnerkegel. Deswegen muss die reduzierte lexikographische Gröbnerbasis des Ideals $\langle F\rangle$ ausgerechnet werden, wobei F die reduzierte Gröbnerbasis von I bzgl. der τ'-gewichteten monomialen Ordnung $\succ_{(\tau',lex)}$ und τ' der gestörte Zielgewichtsvektor ist.

Im Perturbationwalkalgorithmus vom Perturbationgradpaar $(2,2)$ besteht F aus 4 Polynomen mit 289, 278, 280 bzw. 290 Monomen und vom Perturbationgradpaar $(3,3)$ aus 5 Polynomen mit 44, 355, 370, 359 bzw. 359 Monomen.

Die reduzierte lexikographische Gröbnerbasis von I besteht aus 4 Polynomen mit 44, 466, 460 bzw. 459 Monomen und braucht ca. 590 Kilobyte an Speicher. □

Beispiel 4.1.5. *(**cyclic 6** in Faugère et al. (1993)) Sei*

$$\begin{aligned} I = \langle\, & x_1+x_2+x_3+x_4+x_5+x_6,\\ & x_1x_2+x_1x_6+x_2x_3+x_3x_4+x_4x_5+x_5x_6,\\ & x_1x_2x_3+x_1x_2x_6+x_1x_5x_6+x_2x_3x_4+x_3x_4x_5+x_4x_5x_6,\\ & x_1x_2x_3x_4+x_1x_2x_3x_6+x_1x_2x_5x_6+x_1x_4x_5x_6+x_2x_3x_4x_5+\\ & \quad x_3x_4x_5x_6,\\ & x_1x_2x_3x_4x_5+x_1x_2x_3x_4x_6+x_1x_2x_3x_5x_6+x_1x_2x_4x_5x_6+\\ & \quad x_1x_3x_4x_5x_6+x_2x_3x_4x_5x_6,\\ & x_1x_2x_3x_4x_5x_6-1\,\rangle \end{aligned}$$

ein Ideal in $\mathbb{Q}[x_1,x_2,x_3,x_4,x_5,x_6]$. Tabelle 4.3[3)] zeigt Berechnungszeiten in Sekunden, die Anzahl der Schritte und den maximalen Speicherverbrauch in Kilobyte des Perturbationwalkalgorithmus von verschiedenen Perturbationgradpaaren, um die reduzierte Gröbnerbasis von I bzgl. $\succ_{lex}$ auf $\mathbb{Q}[x_1,x_2,x_3,x_4,x_5,x_6]$ mit $x_1\succ x_2\succ x_3\succ x_4\succ x_5\succ x_6$ auszurechnen.

Im Perturbationwalkalgorithmus vom Perturbationgradpaar (d,d) muss die reduzierte Gröbnerbasis des τ^d-homogenen Ideals $\langle G_d\rangle$ ausgerechnet werden, wobei τ^d der gestörte Zielgewichtsvektor des ursprünglichen Zielgewichtsvektors $\tau := (1,0,0,0,0,0)$ vom Grad d ist. Diese Berechnung benötigt leider sehr viel Speicher.

[3)] >xM bedeutet, dass nach x Minuten das Programm abbricht. Für die Perturbationgradpaare $(5,5)$ bzw. $(6,6)$ sind eine oder mehrere Komponenten des berechneten dazwischenliegenden Gewichtsvektors größer als 2^{31}. In diesen beiden Fällen geht der Algorithmus nur bis auf den zweiten Schritt.

PertWalk mit	Perturbationgradpaar				
	(1,1)	(2,2)	(3,3)	(4,4)	
Variante 4	>51M	>24M	>35M	47.13	Bz.
Variante 4'					
Variante 4	2	9	69	124	Sc.
Variante 4'					
Variante 4	∞	∞	∞	34245	Sp.
Variante 4'					

Tabelle 4.3: Zeit-, Schritt- und Speicherverbrauch des Perturbationwalkalgorithmus.

(i) Für $d = 1$ besteht G_1 aus 70 Polynomen mit 1, 9, 24, 24, 24, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 25, 25, 25, 25, 25, 25, 25, 25, 25, 24, 24, 24, 24, 24, 24, 25, 25, 24, 24, 24, 24, 24, 24, 25, 25, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 26, 27, 27, 27, 27, 27, 27, 27, 26, 27, 27 bzw. 27 Monomen.

(ii) Für $d = 2$ besteht G_2 aus 47 Polynomen mit 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 2, 2, 2, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 2, 1 bzw. 1 Monomen.

(iii) Für $d = 3$ besteht G_3 aus 39 Polynomen mit 24, 24, 24, 24, 24, 24, 24, 24, 24, 2, 24, 2, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 2, 1, 2, 2, 2, 1, 1 bzw. 1 Monomen.

Die reduzierte lexikographische Gröbnerbasis selbst besteht aus 17 Polynomen mit 8, 18, 20, 22, 23, 23, 24, 10, 18, 25, 11, 23, 23, 24, 27, 25 bzw. 6 Monomen und ist in 9.395 Byte gespeichert. □

In Beispiel 4.1.3 ist der Zeitverbrauch beider Algorithmen, nämlich des Perturbationwalkalgorithmus mit der Variante 4 und des Perturbationwalkalgorithmus mit der Variante 4', ähnlich. Während in Beispiel 4.1.4 der Perturbationwalkalgorithmus mit der Variante 4' schneller ist als der Perturbationwalkalgorithmus mit der Variante 4.

Betrachten wir die Beispiele, so können wir folgendes feststellen:

1. Der Perturbationwalkalgorithmus vom Perturbationgradpaar (d, d) mit $d \neq 1$ ist schneller als der Gröbnerwalkalgorithmus.

2. Die Anzahl der Schritte und die Berechnungszeit des Perturbationwalkalgorithmus hängen vom gewählten Perturbationgradpaar ab. Sind die Komponenten der berechneten dazwischenliegenden Gewichtsvektoren kleiner als die obere Schranke der erlaubten ganzen Zahl in SINGULAR (2^{31}) und liegt der gestörte Zielgewichtsvektor im richtigen Gröbnerkegel, so gilt im Allgemeinen: je größer das Perturbationgradpaar ist, desto mehr Schritte und weniger Zeit braucht der Perturbationwalkalgorithmus.

3. Ist das Perturbationgradpaar groß genug, kann es sein, dass eine oder mehrere Komponenten des berechneten dazwischenliegenden Gewichtsvektors größer als 2^{31} ist oder der gestörte Zielgewichtsvektor liegt nicht im richtigen Gröbnerkegel.

4. Die Prozedur (*Reduzierung*) zur Reduzierung einer Gröbnerbasis benötigt viel Zeit.

Tabelle 4.4 zeigt den Zeitverbrauch des Perturbationwalkalgorithmus zur Berechnung der reduzierten lexikographischen Gröbnerbasis eines Ideals im Polynomring $\mathbb{Q}[x_1, \ldots, x_n]$. Tabelle 4.5 zeigt die Anzahl der Schritte des Algorithmus. Tabelle 4.6 zeigt die Laufzeitanalyse des Perturbationwalkalgorithmus vom besten Perturbationgradpaar.

Beispiele	n	Perturbationgradpaar					
		(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)
ex1	3	0.20	0.15†	**0.09**†	—	—	—
ex2	3	1698.66	**0.58**	0.61†	—	—	—
ex3	3	6999.66	**0.49**	0.56	—	—	—
kats5	5	665.83	0.84	0.59	**0.05**	‡	—
s6	6	0.22	0.18	**0.08**	0.09	‡	‡

Tabelle 4.4: Berechnungszeiten in Sekunden des Perturbationwalkalgorithmus mit Variante 4' (†Der gestörte Zielgewichtsvektor liegt nicht im richtigen Gröbnerkegel; ‡Eine oder mehrere Komponenten des berechneten dazwischenliegenden Gewichtsvektors sind größer als 2^{31}).

Beispiele	Perturbationgradpaar					
	(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)
ex1	13	35	31	—	—	—
ex2	11	46	42	—	—	—
ex3	13	50	56	—	—	—
kats5	2	4	11	23	4	—
s6	2	12	29	39	1	1

Tabelle 4.5: Anzahl der Schritte des Perturbationwalkalgorithmus.

Beispiele	Grad	Laufzeit (%) zur				
		InFo	sG_ω	Lift	InRd	NäVe‡
ex1†	(3,3)	0.00	7.69	7.69	48.96	0.00
ex2	(2,2)	5.17	7.90	3.45	84.48	0.00
ex3	(2,2)	4.17	2.08	14.58	75.00	0.00
kats5	(4,4)	20.00	0.00	0.00	80.00	0.00
s6	(3,3)	12.50	37.50	12.50	12.50	12.50

Tabelle 4.6: Die Laufzeitanalyse des Perturbationwalkalgorithmus (†38.48% der Berechnungszeit wird in den Schritten nach der Ausführung von Variante 4' benötigt; ‡*NäVe* ist die Abkürzung der Prosedur *NächsterVektor*).

4.2 Fractalwalkalgorithmus

Die Effizienz des in Unterkapitel 4.1 vorgestellten Perturbationwalkalgorithmus hängt sehr vom gewählten Perturbationgradpaar ab. Für niedrige Paare erhält man ein Erzeugendensystem G_ω eines Initialformenideals $\langle I_\omega \rangle \subset K[x_1, \ldots, x_n]$, dessen Elemente viele Terme haben. Dann muss eine reduzierte Gröbnerbasis von $\langle I_\omega \rangle$ ausgerechnet werden. Diese Berechnung ist im Allgemeinen sehr aufwendig. Für hohe Paare werden eine oder mehrere Komponenten eines berechneten dazwischenliegenden Gewichtsvektors sehr groß. Um diese Abhängigkeit zu vermeiden, haben Amrhein et al. (1998) eine andere Variante des Gröbnerwalkalgorithmus entwickelt. Dieses Unterkapitel gibt einen Überblick über diese Methode.

Um eine reduzierte Gröbnerbasis von $\langle I_\omega \rangle$ zu bestimmen, gehen Amrhein et al. wie folgt vor: Sie rufen rekursiv den Perturbationwalkalgorithmus für aufsteigende Perturbationgradpaare auf. Diese Vorgehensweise wird **Fractalwalkalgorithmus** genannt. Genauer muss man zur Berechnung der reduzierten Gröbnerbasis von $\langle I_\omega \rangle$ bezüglich der ω-gewichteten Ordnung $\gg_\omega$ zuerst den Gewichtsvektor ω bezüglich der aktuellen Ordnung[4)] und gleichzeitig auch bezüglich der Zielordnung $\gg$ vom Grad 2 stören. Wir bezeichnen mit $\omega'_{(2)}$ bzw. $\omega''_{(2)}$ den entsprechenden gestörten Gewichtsvektor von ω. Man führt dann den Gröbnerwalkalgorithmus auf der Strecke $\overline{\omega'_{(2)}\omega''_{(2)}}$ aus.

Dazu berechnet man den nächsten dazwischenliegenden Gewichtsvektor $\omega_{(2)} \in \overline{\omega'_{(2)}\omega''_{(2)}}$ und die reduzierte Gröbnerbasis von $\langle (G_\omega)_{\omega_{(2)}} \rangle$ bezüglich der Ordnung $\gg_{\omega_{(2)}}$. Wir hoffen, dass die Elemente von $(G_\omega)_{\omega_{(2)}}$ weniger Terme haben als die Elemente von G_ω. Diese Gröbnerbasis wird durch den Perturbationwalkalgorithmus mit dem höheren Perturbationgradpaar berechnet. Hierin muss zuerst der Gewichtsvektor $\omega_{(2)}$ bezüglich zwei verschiedener Ordnungen gestört werden. Seien $\omega'_{(3)}$ und $\omega''_{(3)}$ zwei gestörte Gewichtsvektoren von $\omega_{(2)}$ vom Grad 3. Dann führt man den Gröbnerwalkalgorithmus auf der Strecke $\overline{\omega'_{(3)}\omega''_{(3)}}$ zur Berechnung der reduzierten Gröbnerbasis von $\langle (G_\omega)_{\omega_{(2)}} \rangle$ bezüglich der Ordnung $\gg_{\omega_{(2)}}$ aus. Diese Rekursion wird solange wiederholt, bis das gewählte Perturbationgradpaar mit (n, n) übereinstimmt. Der Buchbergeralgorithmus wird erst in der Rekursionstiefe n durchgeführt. In dieser Variante werden die Elemente des Erzeugendensystems eines Initialformenideals nicht betrachtet. Man bezeichnet diese Vorgehensweise dann auch als **Fractalwalkalgorithmus mit blinder Rekursion**.

Werden dagegen die Elemente des Erzeugendensystems eines Initialformenideals betrachtet, spricht man vom **Fractalwalkalgorithmus mit alternativer Rekursion**.

Im Fall der alternativen Rekursion wird die nächste Rekursion nur dann durchgeführt, wenn zum Beispiel ein oder mehrere Elemente des Erzeugendensystems mehr als zwei Monome haben.[5)] Diese Strategie wird gewählt, da der Buchbergeralgorithmus sehr effizient für binomiale Ideale ist.

Zur Berechnung jedes Gewichtsvektors $\omega''_{(i)}$ [6)] für $2 \leq i \leq n$ verwendet man entweder die eingegebene reduzierte Gröbnerbasis oder eine sich während der Ausführung

[4)] Diese Ordnung ist ungleich der Ordnung $\gg_\omega$.

[5)] Wenn eine Rekursion niemals aufgerufen wird, stimmt diese Variante mit dem Gröbnerwalkalgorithmus überein.

[6)] Die Gewichtsvektoren $\omega'_{(2)}, \omega'_{(3)}, \ldots, \omega'_{(n)}$ sind die gestörten Gewichtsvektoren eines Gewichtsvektors mit verschiedenen Graden von 2 bis n, der durch die aktuelle Ordnung verfeinert wird. Die

des Algorithmus ergebende Gröbnerbasis. Deswegen muss am Ende jeder Rekursion getestet werden, ob der aktuelle gestörte Zielgewichtsvektor $\omega''_{(i)}$ für ein $i \in \{2, \ldots, n\}$ im richtigen Gröbnerkegel liegt.[7)]

Normalerweise verwendet man eine Basiskonvertierung, inklusive der Varianten des Gröbnerwalkalgorithmus, um eine graduierte reduzierte Gröbnerbasis in die lexikographische reduzierte Gröbnerbasis zu konvertieren. Wird eine der vorgestellten Varianten des Fractalwalkalgorithmus in diesem Fall angewendet, so ist der Zielgewichtsvektor $\tau = (1, 0, \ldots, 0)$ und der letzte dazwischenliegende Gewichtsvektor ω_t im Gröbnerwalkalgorithmus ist ebenfalls gleich τ. Das heißt, im letzten Schritt des Algorithmus muss das Initialformenideal $\langle I_\tau \rangle$ eines Ideals $I = \langle G' \rangle$ bezüglich τ und seine reduzierte Gröbnerbasis bezüglich der lexikographischen Ordnung berechnet werden. In der Tat gilt $in_\tau(f) = f$ für alle $f \in K[x_2, \ldots, x_n]$. Daraus folgt, dass die meisten Elemente des Erzeugendensystems G'_τ von $\langle I_\tau \rangle$ gleich der Elemente von G' sind. Also ist die Berechnung der reduzierten Gröbnerbasis von $\langle G'_\tau \rangle$ bezüglich der lexikographischen Ordnung aufwendig. Aus diesem Grund stört man nur diese Strecke [8)], das heißt, man stört den vorletzten dazwischenliegenden Gewichtsvektor ω_{t-1} und den ursprünglichen Zielgewichtsvektor τ vom nächsten Grad, also vom Grad 2. Danach wird der Algorithmus auf der neuen Strecke $\overline{\omega'_{t-1}\tau'}$ ausgeführt, um die lexikographische Gröbnerbasis von $\langle G' \rangle$ zu berechnen. Diese Basis ist die reduzierte lexikographische Gröbnerbasis des gegebenen Ideals.

In Algorithmus 4.2.2 stellen wir nur den Fractalwalkalgorithmus mit blinder Rekursion für den Fall dar, dass $\langle I_\sigma \rangle$ ein monomiales Ideal ist. Sollte $\langle I_\sigma \rangle$ kein monomiales Ideal sein, so müssen wir zu Beginn einen zusätzlichen Schritt für σ durchführen: Wir müssen den Startgewichtsvektor σ und den Zielgewichtsvektor τ jeweils vom Perturbationsgrad p, für alle $2 \leq p \leq n$, stören. In der Rekursionstiefe p wählen wir dann $\sigma'_{(p)}$ bzw. $\tau'_{(p)}$ als den aktuellen Start- bzw. Zielgewichtsvektor. Hierbei ist $\sigma'_{(p)}$ (bzw. $\tau'_{(p)}$) der gestörte Gewichtsvektor von σ (bzw. τ) vom Grad p.

Die vollständige Variante der blinden Rekursion zur Berechnung reduzierter lexikographischer Gröbnerbasen wird in Algorithmus 4.2.3 vorgestellt.

Beispiel 4.2.1. *(**Wang6** in http://www.calfor.lip6.fr/) Wir berechnen die reduzierte Gröbnerbasis des Ideals*

$$\begin{aligned} I = \langle & 2x_2x_4x_3x_1 - 4x_2x_4 + 2x_3x_1 + 4x_3x_1x_2 - 4x_4^2x_1 - 2x_4^2x_3x_1 + x_3^2x_4^2 - \\ & 2x_2^2x_1 + x_2^2x_1^2 + 2x_2x_4x_1 - 2x_2x_4x_1^2 + x_4^2x_1^2 - 2x_2x_4x_3 - 8x_3 + x_2^2 + 4x_4^2 - \\ & 2x_1 + 10x_3^2 + 2, \quad -2x_3^2 + 8x_3 - 2 - 2x_3x_1 - 2x_1, \\ & 2x_4x_3x_1 + 4x_4x_3^2 + x_4x_1 - 7x_4x_3 + x_2x_3 - 3x_3x_1x_2 + 4x_2, \\ & 4 - 4x_3 - 4x_1^2 + 3x_2^2x_1^2 - 6x_2^2x_1 + 3x_2^2 + 9x_3^2x_4^2 + 6x_4^2x_3x_1 - 3x_4^2x_1^2 - \\ & 24x_3x_4^2 + 12x_4^2 + 4x_3^2 + 12x_2x_4x_3 + 12x_2x_4x_1 + 12x_2x_4x_3x_1 - 12x_2x_4\rangle \end{aligned}$$

Gewichtsvektoren $\omega''_{(2)}, \omega''_{(3)}, \ldots, \omega''_{(n)}$ sind die gestörten Gewichtsvektoren des ursprünglichen Zielgewichtsvektors τ mit verschiedenen Graden von 2 bis n. Die Gewichtsvektoren $\omega''_{(i)}$ für $2 \leq i \leq n$ werden am Anfang des Algorithmus berechnet und während der Berechnung erneuert (siehe Fußnote 7). Aber die Gewichtsvektoren $\omega'_{(i)}$ für $2 \leq i \leq n$ müssen zum Beginn der ersten Rekursion bestimmt werden, wenn das Initialformenideal $\langle G_\omega \rangle$ kein monomiales Ideal ist. Ist $\langle G_\omega \rangle$ ein monomiales Ideal, setzen wir $\omega'_{(i)} := \omega$ für $2 \leq i \leq n$.

[7)] Wenn Ja, liefert diese Rekursion die bereits berechnete reduzierte Gröbnerbasis zurück. Im anderen Fall erhält man nur die reduzierte Gröbnerbasis bezüglich einer von der gewünschten Ordnung verschiedenen. Deswegen müssen die Zielgewichtsvektoren erneut gestört werden. Der Algorithmus setzt dann auf der so erhaltenen neuen Strecke fort.

[8)] Wegen dieser Störung braucht man nicht das Initialformenideal $\langle I_\tau \rangle$ zu berechnen.

im Polynomring $K[x_1, x_2, x_3, x_4]$ *bzgl. der lexikographischen Ordnung* $\succ_{lex}$ *auf* $K[x_1, x_2, x_3, x_4]$ *mit* $x_1 > x_2 > x_3 > x_4$.

Algorithmus 4.2.2. ***FractalWalk***$(G, \succ, \gg, 1)$;

Eingabe: *Zwei verschiedene monomiale Ordnungen* $\succ$ *und* $\gg$ *auf* $R := K[x_1, \ldots, x_n]$ *und die reduzierte Gröbnerbasis* G *von* $I \subset R$ *bzgl. der den Gewichtsvektor* σ *verfeinernden Ordnung* $\succ$.

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl. der durch eine nichtnegative rationale* $(n \times n)$*-Matrix* B *definierten Ordnung* $\gg$.

Initialisierung: $\tau'_{(i)} := PertVektor(G, B, i)$ *für* $1 \leq i \leq n$;

FractalWalk$(G, \gg_\sigma, \gg, p)$;

1. $\omega := NächsterVektor(G, \sigma, \tau'_{(p)})$;
2. ***if*** (ω *ist undefiniert*) ***then***
   (a) ***if*** $(\tau'_{(p)} \in cone(G, \gg))$ ***then return***(G);
   (b) $B' :=$ *eine Matrix, die die aktuelle Ordnung* $\gg$ *definiert;*
   (c) $\tau'_{(i)} := PertVektor(G, B', i)$ *für* $1 \leq i \leq n$;
   (d) ***go to*** (1);
3. $G_\omega := InitialForm(G, \omega)$;
4. ***if*** $(p == n)$ ***then*** $M := ReduziertGB(G_\omega, \gg_\omega)$;
5. ***else***
   (a) $p := p + 1$;
   (b) $M := FractalWalk(G_\omega, \gg_\sigma, \gg_\omega, p)$;
6. $F := LiftGB(G, G_\omega, M, \gg_\omega)$;
7. $G := Reduzierung(F, \gg_\omega)$;
8. $\sigma := \omega$;
9. ***go to*** (1);

Algorithmus 4.2.2: *Der Fractalwalkalgorithmus mit blinder Rekursion.*

Die reduzierte Gröbnerbasis von I bzgl. $\succ_{rlex}$ auf $K[x_1, x_2, x_3, x_4]$ mit $x_1 > x_2 > x_3 > x_4$ besteht aus 18 Polynomen mit 5, 8, 25, 25, 25, 9, 21, 25, 25, 25, 25, 18, 25, 25, 25, 18, 20 bzw. 25 Monomen. Mit Hilfe dieser Gröbnerbasis wird die reduzierte lexikographische Gröbnerbasis von I ausgerechnet.

Im letzten Schritt der Rekursionstiefe 1 ergibt sich ein Ideal, dessen Erzeugendensystem aus 23 Polynomen besteht. Eines dieser Polynome besteht aus 8 Monomen

Algorithmus 4.2.3. ***FractalWalk***$(G, \succ, \gg, 1)$;

Eingabe: *Zwei verschiedene monomiale Ordnungen* $\succ$ *und* $\gg$ *auf* $R := K[x_1, \ldots, x_n]$ *und die reduzierte Gröbnerbasis* G *von* $I \subset R$ *bzgl. der den Gewichtsvektor* σ *verfeinernden Ordnung* $\succ$.

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl. der durch eine nichtnegative rationale* $(n \times n)$*-Matrix* B *definierten Ordnung* $\gg$.

Initialisierung: $\tau'_{(i)} := PertVektor(G, B, i)$ *für* $1 \leq i \leq n$;

FractalWalk$(G, \gg_\sigma, \gg, p)$;

1. ***if*** $(p == 1)$ ***then***
   (a) ***if*** $(\langle G_\sigma \rangle$ *ist kein monomiales Ideal*$)$ ***then***
      i. $A :=$ *eine nichtnegative rationale* $(n \times n)$*-Matrix, die die Ordnung* $\gg_\sigma$ *definiert.*
      ii. $\sigma'_{(i)} := PertVektor(G, A, i)$ *für* $1 \leq i \leq n$;
   (b) ***else*** $\sigma'_{(i)} := \sigma$ *für* $1 \leq i \leq n$;
2. $\sigma := \sigma'_{(p)}$;
3. $\omega := NächsterVektor(G, \sigma, \tau'_{(p)})$;
4. ***if*** $(p == 1$ *und* $\omega == \tau'_{(1)})$ ***then***
   (a) $\tau'_{(i)} := PertVektor(G, B, i)$ *für* $1 \leq i \leq n$;
   (b) $p := p + 1$;
   (c) ***go to*** (2);
5. ***if*** $(\omega$ *ist undefiniert*$)$ ***then***
   (a) ***if*** $(\tau'_{(p)} \in cone(G, \gg))$ ***then return***(G);
   (b) $B' :=$ *eine Matrix, die die aktuelle Ordnung* $\gg$ *definiert;*
   (c) $\tau'_{(i)} := PertVektor(G, B', i)$ *für* $1 \leq i \leq n$;
   (d) ***go to*** (3);
6. $G_\omega := InitialForm(G, \omega)$;
7. ***if*** $(p == n)$ ***then*** $M := ReduziertGB(G_\omega, \gg_\omega)$;
8. ***else***
   (a) $p := p + 1$;
   (b) $M := FractalWalk(G_\omega, \gg_\sigma, \gg_\omega, p)$;
9. $F := LiftGB(G, G_\omega, M, \gg_\omega)$;
10. $G := Reduzierung(F, \gg_\omega)$;
11. $\sigma := \omega$; ***go to*** (3);

Algorithmus 4.2.3: *Der Fractalwalkalgorithmus mit blinder Rekursion zur Berechnung reduzierter lex. Gröbnerbasen.*

und jedes der anderen Polynome ist aus 25 Monomen zusammengesetzt. Diese Situation wird in Abbildung 4.1 an der Stelle P erreicht. Der Algorithmus setzt dann auf einer neuen Strecke fort, die durch die Störung des aktuellen dazwischenliegenden Gewichtsvektors und des Zielgewichtsvektors gegeben ist.

Abbildung 4.1) zeigt die Spur des Fractalwalkalgorithmus mit alternativer Rekursion.

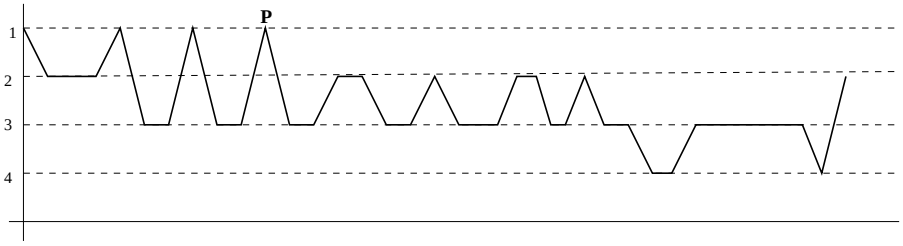

Abbildung 4.1: Die Spur des Fractalwalkalgorithmus mit alternativer Rekursion im Beispiel wang6.

4.3 Tranalgorithmus

In diesem Abschnitt wird der von Tran (2000) entwickelte Algorithmus vorgestellt, der ebenfalls auf dem Gröbnerwalkalgorithmus basiert.

In Kapitel 3 (siehe Tabelle 3.3) wird erwähnt, dass der letzte Schritt des Gröbnerwalkalgorithmus viel Zeit benötigt. Dies lässt sich insbesondere durch die Berechnung der reduzierten Gröbnerbasis eines τ-homogenen Ideals $\langle G_\tau \rangle$ erklären. Diese Berechnung tritt auf, falls der Zielgewichtsvektor τ in mehreren Gröbnerkegeln liegt und dadurch die Elemente von G_τ viele Terme haben. Um dieses Problem zu umgehen, haben Amrhein et al. die Start- und Zielgewichtsvektoren am Anfang des Perturbationwalkalgorithmus gestört. Am „Ende“ dieser Variante muss allerdings getestet werden, ob der gestörte Zielgewichtsvektor im richtigen Gröbnerkegel liegt. Dieser Test muss durchgeführt werden, da die verwendete Verfahrensweise zur Bestimmung des gestörten Zielgewichtsvektors nicht garantiert, dass der berechnete gestörte Zielgewichtsvektor im richtigen Gröbnerkegel liegt. Aus diesem Grund hat Tran eine neue Strategie entwickelt, so dass der „gestörte“ Zielgewichtsvektor immer im richtigen Gröbnerkegel liegt. Diese Methode wird im Gröbnerwalkalgorithmus ausgeführt, falls der berechnete dazwischenliegende Gewichtsvektor mit dem Zielgewichtsvektor übereinstimmt und die bereits berechnete Gröbnerbasis noch nicht die gewünschte Basis ist. Der Algorithmus wird dann auf der neuen Strecke vom vorletzten dazwischenliegenden Gewichtsvektor zum neuen Zielgewichtsvektor weitergeführt. Dieser Zielgewichtsvektor wird auch eine **Repräsentation** des ursprünglichen Zielgewichtsvektors τ genannt.

Um die von Tran entwickelte Variante des Gröbnerwalkalgorithmus zu beschreiben, nehmen wir an, dass im Gröbnerwalkalgorithmus die folgende Situation vorliegt: [9)]

[9)] Falls der Startgewichtsvektor σ in mehreren Gröbnerkegeln liegt, soll man einen neuen Startgewichtsvektor ω_0 wählen. Um zu eintscheiden, ob σ in mehreren Gröbnerkegeln liegt, kann man zum Beispiel die Anzahl der Monome der Elemente des Erzeugendensystems vom Ideal $\langle in_\sigma(I) \rangle$ betrachten.

Seien $\sigma =: \omega_0, \omega_1, \ldots, \omega_{t-1}, \omega_t := \tau$ die berechneten dazwischenliegenden Gewichtsvektoren und $G =: G_0, G_1, \ldots, G_{t-1}$ die reduzierten Gröbnerbasen des gegebenen Ideals $I \subset K[x_1, \ldots, x_n]$ bezüglich der Ordnung $\gg_{\omega_k}$ für $k = 0, 1, \ldots, t-1$. Das heißt, im t-ten Schritt ist der nächste dazwischenliegende Gewichtsvektor gleich dem Zielgewichtsvektor τ. An dieser Stelle muss getestet werden, ob die bereits berechnete reduzierte Gröbnerbasis G_{t-1} die gewünschte Gröbnerbasis ist. Wenn ja, terminiert der Algorithmus und liefert diese Basis zurück. Das heißt, in diesem Fall stimmt diese Variante mit dem Gröbnerwalkalgorithmus überein. Ansonsten wird der Zielgewichtsvektor τ durch seine Repräsentation τ' ersetzt. Dann setzt der Algorithmus auf der Strecke $\overline{\omega_{t-1}\tau'}$ fort.

Zur Bestimmung einer Repräsentation τ' von τ benötigt Tran zwei Informationen:

(1) die obere Schranke des Totalgrads der reduzierten Gröbnerbasis bezüglich der gegebenen Zielordnung $\gg$, die durch den Gewichtsvektor τ repräsentiert wird,

(2) den maximalen Eintrag einer Matrix A, die die Ordnung $\gg$ definiert.

Ist $G_{t-1}^{(h)}$ die homogenisierte Gröbnerbasis von G_{t-1}, so ist die obere Schranke des Totalgrads der nächsten homogenisierten Gröbnerbasis $G_t^{(h)}$:

$$b := 2\deg(G_{t-1}^{(h)})^2 + (n+1)\deg(G_{t-1}^{(h)})$$

mit $\deg(G_{t-1}^{(h)}) := \max\{\deg(g) \ : \ g \in G_{t-1}^{(h)}\}$[10] (siehe Kalkbrener, 1999).

Seien $\tau =: \tau_1, \ldots, \tau_n$ die Zeilen von A mit $\tau_i := (\tau_{i1}, \ldots, \tau_{in}) \in \mathbb{Q}_{\geq 0}^n$ für $i = 1, \ldots, n$ und $m := \max\{\tau_{ij} \ : \ 1 \leq i, j \leq n\}$. Ist $\gg_\tau$ die lexikographische Ordnung, so ist $m = 1$. Dann erhält man

$$\tau' := d^{n-1}\tau_1 + d^{n-2}\tau_2 + \cdots + \tau_n \text{ mit } d := bm$$

als eine Repräsentation von τ. Dieser Gewichtsvektor τ' liegt im Gröbnerkegel $cone(\tau)$ von I bezüglich τ (siehe Tran, 2000). Deswegen braucht man nicht mehr diesen Vektor zu testen wie in den von Amrhein et al. (1996, 1997, 1998) entwickelten Algorithmen. Algorithmus 4.3.1 liefert dann eine Repräsentation eines Gewichtsvektors, nämlich der ersten Zeile einer Matrix, die eine monomiale Ordnung definiert.

Der von Tran entwickelte Algorithmus läßt sich hiermit wie Algorithmus 4.3.2 darstellen und wird im Folgenden mit **Tranalgorithmus** bezeichnet. Die Terminierung und die Richtigkeit des Algorithmus hängen vom Gröbnerwalkalgorithmus und von der Definition einer Repräsentation des Zielgewichtsvektors ab.

Man kann auch eine Variante dieses Algorithmus wählen, in der der berechnete dazwischenliegende Gewichtsvektor ω_k für ein $1 < k < t$ durch seine Repräsentation ω_k' ersetzt wird, falls ω_k in mehreren Gröbnerkegeln liegt. Das heißt, der Algorithmus geht zuerst auf der Strecke $\overline{\omega_{k-1}\omega_k'}$ und dann auf der Strecke $\overline{\omega_k'\tau}$ weiter. Zur Bestimmung des Gewichtsvektors ω_k' benötigt man eine nichtnegative rationale $(n \times n)$-Matrix A_k, die die Ordnung $\gg_{\omega_k}$ definiert, und den maximalen Grad der Elemente der reduzierten Gröbnerbasis G_{k-1}.

Beispiel 4.3.3. *Wir wenden den Tranalgorithmus auf Beispiel 4.1.4 an.*

[10] $\deg(G_{t-1}^{(h)}) = \deg(G_{t-1})$.

Algorithmus 4.3.1. ***RepräsentationVektor***(G, A);

Eingabe: *Die reduzierte Gröbnerbasis G eines Ideals I in $K[x_1, \ldots, x_n]$ bezüglich $\gg_{\omega_{t-1}}$ und eine nichtnegative rationale $(n \times n)$-Matrix A mit erster Zeile ω_t, wobei ω_{t-1} und ω_t zwei im Gröbnerwalkalgorithmus verwendete dazwischenliegende Gewichtsvektoren sind. Diese Matrix A definiert die Ordnung $\gg_{\omega_t}$.*

Ausgabe: *Die Repräsentation τ' von ω_t.*

Initialisierung: *Für $i = 1, \ldots, n$ sei $\tau_i := (\tau_{i1}, \cdots, \tau_{in}) \in \mathbb{Q}^n_{\geq 0}$ die i-te Zeile von A mit $\tau_1 := \omega_t$.*

begin

1. $m := \max\{\tau_{ij} \ : \ 1 \leq i, j \leq n\}$;
2. $d_0 := \max\{\deg(g) \ : \ g \in G\}$;
3. $d := m(2d_0^2 + (n+1)d_0)$;
4. $\tau' := d^{n-1}\tau_1 + d^{n-2}\tau_2 + \cdots + d\tau_{n-1} + \tau_n$;
5. ***return***(τ');

end.

Algorithmus 4.3.1: *Repräsentationvektoralgorithmus.*

Zuerst geht der Algorithmus auf der Strecke von $(1,1,1)$ nach $\tau := (1,0,0)$. Im 15-ten Schritt wird der ursprüngliche Zielgewichtsvektor τ erreicht. Hier ergibt sich die reduzierte Gröbnerbasis G des eingegebenen Ideals bezüglich der ω-gewichteten lexikographischen Ordnung mit $\omega := (16,1,1)$. Diese Gröbnerbasis besteht aus zehn Polynomen. Sieben davon haben 144 Monome und jedes der anderen Polynome ist aus 44, 157 bzw. 145 Monomen zusammengesetzt.

Da die ausgerechnete reduzierte Gröbnerbasis G noch nicht die gesuchte reduzierte lexikographische Gröbnerbasis ist, muss die Repräsentation τ' von τ ausgerechnet werden, nämlich $\tau' = (1322500, 1150, 1)$. Dann geht der Algorithmus weiter auf der Strecke von $(16,1,1)$ nach τ'.

Von da aus benötigt der Algorithmus noch 26 Schritte. Im 26-ten Schritt, in dem der nächste dazwischenliegende Gewichtsvektor ω' von $(13230968, 11873, 383)$ nach τ' bestimmt wird, ist ω' undefiniert. Ein Test, ob die bereits berechnete Gröbnerbasis die reduzierte lexikographische Gröbnerbasis ist, fällt positiv aus. Der Algorithmus terminiert und liefert diese Basis zurück. Insgesamt benötigt der Algorithmus 41 Schritte, 10.38 Sekunden und 8572 Kilobyte Speicher. Diese Zeit verteilt sich auf die in Tabelle 4.7 aufgelisteten Prozeduren.

Tabelle 4.8 zeigt den Vergleich mehrerer Berechnungszeiten des Tranalgorithmus, der bereits in SINGULAR (Laptop PC Pentium II 266 MHz Prozessor mit 32 Megabyte RAM) und MATHEMATICA (Laptop PC mit Intel MMX-233 MHz Prozessor und 144

Algorithmus 4.3.2. ***TranWalk***$(G, \succ, \gg)$;

Eingabe: *Zwei verschiedene monomiale Ordnungen* $\succ$ *und* $\gg$ *auf* $R := K[x_1, \ldots, x_n]$, *und die reduzierte Gröbnerbasis* G *eines Ideals* I *in* R *bzgl.* $\succ$.

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl.* $\gg$.

Initialisierung:

- $B :=$ *eine nichtnegative rationale* $(n \times n)$*-Matrix, die die Ordnung* $\gg$ *definiert und die erste Zeile von* B *ist* τ.
- $\sigma :=$ *ein Gewichtsvektor, der durch* $\succ$ *verfeinert wird.*
- *Liegt* σ *in mehreren Gröbnerkegeln, muss man einen neuen Startgewichtsvektor nehmen. Setze* $\sigma :=$ *RepräsentationVektor*(G, A), *wobei* A *die Ordnung* $\succ$ *definiert. In diesem Fall kann man die Zeile (3) in Algorithmus 4.3.1 durch* $d := md$ *ersetzen, weil die benötigte Gröbnerbasis vorhanden ist.*

while *wahr* ***do***

1. $\omega :=$ *NächsterVektor*(G, σ, τ);
2. ***if*** $(\omega == \tau$ *oder* ω *ist undefiniert)* ***then***
   (a) ***if*** $(G$ *ist die reduzierte Gröbnerbasis von* I *bzgl.* $\gg)$ ***then***
   return(G);
   (b) ***if*** $(\tau$ *liegt in mehreren Gröbnerkegeln)* ***then***
   $\tau :=$ *RepräsentationVektor*(G, B);
   go to (1);
3. $G_\omega :=$*InitialForm*(G, ω);
4. $M :=$ *ReduziertGB*$(G_\omega, \gg_\omega)$;
5. $F :=$ *LiftGB*$(G, G_\omega, M, \gg_\omega)$;
6. $G :=$ *Reduzierung*$(F, \gg_\omega)$;
7. $\sigma := \omega$;

end.

Algorithmus 4.3.2: *Tranalgorithmus.*

Prozedur	Zeit	
	in Sek.	%
InitialForm	0.01	0.10
ReduziertGB	0.12	1.16
LiftGB	0.96	9.26
Reduzierung	9.20	88.72
NächsterVektor	0.02	0.19

Tabelle 4.7: Die Laufzeitanalyse des Tranalgorithmus.

Megabyte RAM, siehe Tran (2000)[11]) implementiert ist.

Bsp.	n	SINGULAR	MATHEMATICA
ex. 3.1	3	1884.58†	(?)
ex. 3.2	4	1.27	5.33
ex. 3.3	3	55.53	91.07
ex. 3.4	3	0.22	0.93

Tabelle 4.8: Berechnungszeiten in Sekunden des Tranalgorithmus ((?) := Die Berechnungszeit ist unbekannt; †Wir benutzten einen PC Pentium IV 2.40 GHz und 1 GB RAM; Im 121-ten Schritt wird der ursprüngliche Zielgewichtsvektor erreicht. Der Algorithmus setzt dann auf einer neuen Strecke fort, die durch den vorletzten dazwischenliegenden Gewichtsvektor und die Repräsentation des ursprünglichen Zielgewichtsvektors gegeben ist. Von da aus geht der Algorithmus nur bis auf den 162-ten Schritt, da in diesem Schritt eine Komponente des berechneten dazwischenliegenden Gewichtsvektors ω_{283} größer als 2^{31} ist. An dieser Stelle erhält man die reduzierte Gröbnerbasis G des gegebenen Ideals I bezüglich der ω_{282}-gewichteten lexikographischen Ordnung. Zur Berechnung der reduzierten lexikographischen Gröbnerbasis von I haben wir den Buchbergeralgorithmus auf dem Ideal $\langle G \rangle$ angewandt).

[11]) Tran hat seinen Algorithmus in MATHEMATICA implementiert. Seine Beispiel beziehen auf einen nicht weiter spezifierten Körper.

4.4 Alternativalgorithmen

Wir betrachten das Ideal (*virasoro* in http://www.calfor.lip6.fr/)

$$\begin{aligned} I = \langle & 8x_1^2 + 8x_1x_2 + 8x_1x_3 + 2x_1x_4 + 2x_1x_5 + 2x_1x_6 + 2x_1x_7 - \\ & 8x_2x_3 - 2x_4x_7 - 2x_5x_6 - x_1, \\ & 8x_1x_2 - 8x_1x_3 + 8x_2^2 + 8x_2x_3 + 2x_2x_4 + 2x_2x_5 + 2x_2x_6 + \\ & 2x_2x_7 - 2x_4x_6 - 2x_5x_7 - x_2, \\ & -8x_1x_2 + 8x_1x_3 + 8x_2x_3 + 8x_3^2 + 2x_3x_4 + 2x_3x_5 + 2x_3x_6 + \\ & 2x_3x_7 - 2x_4x_5 - 2x_6x_7 - x_3, \\ & 2x_1x_4 - 2x_1x_7 + 2x_2x_4 - 2x_2x_6 + 2x_3x_4 - 2x_3x_5 + 8x_4^2 + \\ & 8x_4x_5 + 2x_4x_6 + 2x_4x_7 + 6x_4x_8 - 6x_5x_8 - x_4, \\ & 2x_1x_5 - 2x_1x_6 + 2x_2x_5 - 2x_2x_7 - 2x_3x_4 + 2x_3x_5 + 8x_4x_5 - \\ & 6x_4x_8 + 8x_5^2 + 2x_5x_6 + 2x_5x_7 + 6x_5x_8 - x_5, \\ & -2x_1x_5 + 2x_1x_6 - 2x_2x_4 + 2x_2x_6 + 2x_3x_6 - 2x_3x_7 + 2x_4x_6 + \\ & 2x_5x_6 + 8x_6^2 + 8x_6x_7 + 6x_6x_8 - 6x_7x_8 - x_6, \\ & -2x_1x_4 + 2x_1x_7 - 2x_2x_5 + 2x_2x_7 - 2x_3x_6 + 2x_3x_7 + 2x_4x_7 + \\ & 2x_5x_7 + 8x_6x_7 - 6x_6x_8 + 8x_7^2 + 6x_7x_8 - x_7, \\ & -6x_4x_5 + 6x_4x_8 + 6x_5x_8 - 6x_6x_7 + 6x_6x_8 + 6x_7x_8 + 8x_8^2 - x_8 \rangle \end{aligned} \tag{4.3}$$

im Polynomring $R := \mathbb{Q}[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]$ und die lexikographische monomiale Ordnung $\succ_{lex}$ auf R mit

$$x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > x_7 > x_8.$$

Mit keinem der in den vergangenen Abschnitten beschriebenen Algorithmen erhalten wir die reduzierte Gröbnerbasis von I bzgl. $\succ_{lex}$, weil die Berechnung einer solchen Gröbnerbasis zu viel Speicher und Zeit benötigt. Jeder Algorithmus davon geht wie folgt vor:

1. Der Gröbnerwalkalgorithmus braucht nur zwei Schritte, um den Zielgewichtsvektor $\tau := (1, 0, \ldots, 0) \in \mathbb{Z}^8$ zu erreichen. Im letzten Schritt ergibt sich eine Gröbnerbasis G_τ eines τ-homogenen Ideals bzgl. $\succ_{(\omega, lex)}$ mit $\omega := (1, 1, \ldots, 1) \in \mathbb{Z}^8$. Es muss noch die reduzierte Gröbnerbasis des Ideals $\langle G_\tau \rangle$ bzgl. $\succ_{(\tau, lex)}$ mit Hilfe des Hilbert-driven Algorithmus ausgerechnet werden. Die Berechnung benötigt aber zu viel Speicher und bricht nach ca. 26 Minuten ab.

2. Der Perturbationwalkalgorithmus bricht nach einiger Zeit für jedes Perturbationgradpaar (p, p) mit $1 \leq p \leq 4$ ab, wenn die reduzierte Gröbnerbasis eines $\tau'_{(p)}$-homogenen Ideals $\langle G_{\tau'_{(p)}} \rangle$ bezüglich der $\tau'_{(p)}$-gewichteten lexikographischen Ordnung ausgerechnet werden muss (siehe Tabelle 4.9). Der Gewichtsvektor $\tau'_{(p)}$ selbst ist durch die Störung des ursprünglichen Zielgewichtsvektors $\tau := (1, 0, \ldots, 0) \in \mathbb{Z}^8$ vom Grad p gegeben. Für die anderen Perturbationgradpaare sind eine oder mehrere Komponenten eines dazwischenliegenden Gewichtsvektors oder des gestörten Start- und Zielgewichtsvektors größer als 2^{31}. Der erste Fall tritt für die Paare $(5, 5)$ bzw. $(6, 6)$ auf und der zweite für die Paare $(7, 7)$ bzw. $(8, 8)$.

	Perturbationgradpaar			
	(1,1)	(2,2)	(3,3)	(4,4)
B.-Zeit	> 26M	> 17M	> 37M	> 51M
Schritte	2	7	30	55

Tabelle 4.9: Zeit- und Schrittverbrauch des Perturbationwalkalgorithmus.

3. Der Fractalwalkalgorithmus geht wie folgt vor: Im ersten Schritt der Rekursionstiefe 1 wird der Zielgewichtsvektor $\tau = (1, 0, \ldots, 0)$ erreicht. Man stört dann die Strecke $\overline{\sigma\tau}$ mit $\sigma = (1, \ldots, 1)$ vom Grad 2. Das heißt, die Rekursionstiefe 2 wird aufgerufen und benötigt nur einen Schritt, um den aktuellen Zielgewichtsvektor $\tau'_{(2)}$ zu erreichen. Zur Berechnung der reduzierten Gröbnerbasis des Ideals $\langle I_{\tau'_{(2)}} \rangle$ bezüglich der $\tau'_{(2)}$-gewichteten lexikographischen Ordnung wird die Rekursionstiefe 3 aufgerufen. Wie in der Rekursionstiefe 2 wird im ersten Schritt der aktuelle Zielgewichtsvektor $\tau'_{(3)}$ erreicht. Dann wird die Rekursionstiefe 4 aufgerufen. Im ersten Schritt sind eine oder mehrere Komponenten des berechneten dazwischenliegenden Gewichtsvektors größer als 2^{31}. Die Anwendung des Buchbergeralgorithmus zur Berechnung der reduzierten Gröbnerbasis von $\langle (I_{\tau'_{(2)}})_{\tau'_{(3)}} \rangle$ bezüglich der $\tau'_{(3)}$-gewichteten lexikographischen Ordnung ist sehr aufwendig. Nach circa 26 Minuten bricht die Berechnung wegen des Speichermangels ab.

4. Der Tranalgorithmus benötigt 12 Schritte, um den ursprünglichen Zielgewichtsvektor $\tau = (1, 0, \ldots, 0) \in \mathbb{N}^8$ zu erreichen. An dieser Stelle erhält man die reduzierte Gröbnerbasis G von I bezüglich der ω_{11}-gewichteten lexikographischen Ordnung mit

   $$\omega_{11} = (723349, 597871, 597870, 597861, 597780, 597051, 590490, 531441).$$

   Der maximale Totalgrad von G ist 81. Also ist die obere Schranke der reduzierten lexikographischen Gröbnerbasis 243. Dann muss die Repräsentation τ' von τ ausgerechnet werden. Eine Komponente dieser Repräsentation ist allerdings größer als 2^{31}. Deswegen kann man nicht diesen „fehlenden“ Gewichtsvektor zur Anwendung in SINGULAR bringen.

Amrhein et. al und Tran wählten verschiedene Verfahrensweisen zur Bestimmung einer „effizienten“ Strecke, auf der der Gröbnerwalkalgorithmus ausgeführt wird. Diese Strecke ist durch eine Störung bzw. die Repräsentation eines Gewichtsvektors gegeben.

In jeder dieser beiden Verfahrensweisen spielt die gewählte ganze Zahl - entweder $\frac{1}{\epsilon}$ (vgl. Gleichung (4.1)) oder d (vgl. Zeile 3 von Algorithmus 4.3.1) - eine große Rolle. Dann muss noch in jedem der von Amrhein et. al bzw. Tran entwickelten Algorithmen getestet werden, ob $\frac{1}{\epsilon}$ richtig oder die bereits berechnete Gröbnerbasis die gesuchte Gröbnerbasis ist.

Für große Perturbationgradpaare bzw. Ideale mit vielen Variablen, zum Beispiel mehr als 4 Variablen, tritt in der Praxis (in SINGULAR) ein Problem häufig auf. Es besteht darin, dass eine oder mehrere Komponenten eines gestörten Gewichtsvektors oder der Repräsentation des Zielgewichtsvektors groß oder größer als 2^{31} sind. Die „große“ gewählte ganze Zahl $\frac{1}{\epsilon}$ (oder d) ist eine der Ursachen dafür.

Damit dieses Problem nicht häufig auftritt, wählen wir eine „kleine“ ganze Zahl $\frac{1}{\epsilon}$ zur Störung eines Gewichtsvektors vom Grad p für $1 \leq p \leq n$, wobei n die Anzahl der Variablen ist. Diese Zahl ist als der Quotient der Division von

$$\deg(G)(\max(\omega_2) + \cdots + \max(\omega_p)) + 1$$

durch p definiert, falls gilt

$$3 < p < \deg(G)(\max(\omega_2) + \cdots + \max(\omega_p)) + 1,$$

wobei G die bereits berechnete reduzierte Gröbnerbasis bezüglich der ω_1-gewichteten Ordnung $\gg_{\omega_1}$ ist. Diese Ordnung wird durch eine Matrix M mit Zeilen $\omega_1, \omega_2, \ldots, \omega_n$ definiert.

Also erhalten wir

$$\omega_1' := (\frac{1}{\epsilon})^{p-1}\omega_1 + (\frac{1}{\epsilon})^{p-2}\omega_2 + \cdots + \omega_p$$

als den gestörten Gewichtsvektor des Vektors ω_1 vom Grad p.

Diese Verfahrensweise wird auf die in diesem Abschnitt vorgestellten Algorithmen angewendet, die auch auf dem Gröbner- und Perturbationwalkalgorithmus basieren.

Die Grundidee des Alternativalgorithmus ist wie folgt: Zuerst wenden wir den Gröbnerwalkalgorithmus auf das gegebene Ideal an. Wenn der aktuelle Zielgewichtsvektor erreicht wird und die berechnete Gröbnerbasis noch keine gewünschte Gröbnerbasis ist, verwenden wir eine Rekursion des Perturbationwalkalgorithmus, bis die gewünschte Gröbnerbasis erreicht wird.

4.4.1 Erster Alternativalgorithmus

Der erste Alternativalgorithmus funktioniert folgendermaßen: Zuerst wenden wir den Gröbnerwalkalgorithmus auf das gegebene Ideal an. In jedem Schritt muss getestet werden, ob der erreichte dazwischenliegende Gewichtsvektor gleich dem Zielgewichtsvektor ist. Wenn ja, wird entweder der Buchbergeralgorithmus oder der Perturbationwalkalgorithmus vom Perturbationgradpaar $(1, p)$ mit $1 \leq p \leq n$ aufgerufen, um die gewünschte Gröbnerbasis auszurechnen, wobei n die Anzahl der Variablen ist. Der Buchbergeralgorithmus wird aufgerufen, falls $p = 1$ ist. Falls der Perturbationwalkalgorithmus vom Perturbationgradpaar $(1, p)$ mit $p \neq 1$ aufgerufen wird und eine Komponente des berechneten dazwischenliegenden Gewichtsvektors größer als 2^{31} ist oder der gestörte Zielgewichtsvektor nicht im entsprechenden Gröbnerkegel liegt, wird der Perturbationwalkalgorithmus vom Perturbationgradpaar $(1, p-1)$ aufgerufen. Falls dann im Perturbationwalkalgorithmus vom Perturbationgradpaar $(1, p-1)$ der erreichte dazwischenliegende Gewichtsvektor oder der gestörte Zielgewichtsvektor nicht im entsprechenden Gröbnerkegel liegt, wird der Perturbationwalkalgorithmus vom Perturbationgradpaar $(1, p-2)$ aufgerufen. Ist $p = 2$, so wird die gewünschte Gröbnerbasis durch den Buchbergeralgorithmus ausgerechnet. Dies wird wiederholt, bis die gewünschte reduzierte Gröbnerbasis erreicht oder der Perturbationgradpaar des aufgerufenen Perturbationwalkalgorithmus gleich $(1, 1)$ ist.

Den Algorithmus 4.4.1 bezeichnen wir als **ersten Alternativalgorithmus**, falls $p = n - 1$ ist.

Beispiel 4.4.3. *Wir wenden den ersten Alternativalgorithmus auf Beispiel 4.1.5 (cyclic 6) an.*

Algorithmus 4.4.1. ***Ab_Rec_Walk***(G, σ, τ, p);

Eingabe: *Zwei Gewichtsvektoren* σ *und* τ,
die reduzierte Gröbnerbasis G *eines Ideals* I *mit* n *Variablen bzgl. der Ordnung* $\succ$, *die den Vektor* σ *verfeinert,*
eine positive ganze Zahl p *mit* $1 \leq p \leq n$.

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl. der Ordnung* $\gg$, *die den Vektor* τ *verfeinert.*

Initialisierung: $\omega := \sigma$;

while *wahr* ***do***

1. $G_\omega := InitialForm(G, \omega)$;
2. $M := ReduziertGB(G_\omega, \gg_\omega)$;
3. $F := LiftGB(G, G_\omega, M, \gg_\omega)$;
4. $G := Reduzierung(F, \gg_\omega)$;
5. $\omega := NächsterVektor(G, \sigma, \tau)$;
6. ***if*** $(\omega == \tau)$ ***then***
   return$(Ab\_Rec\_Pert(G, \sigma, \tau, p))$;
7. ***if*** (ω *ist undefiniert*) ***then***
   return(G);
8. $\sigma := \omega$;

Algorithmus 4.4.1: *Der erste Alternativalgorithmus vom Grad* p.

Die Hauptprozedur Ab_Rec_Walk benötigt nur einen Schritt, um den Zielgewichtsvektor $\tau := (1,0,0,0,0,0)$ zu erreichen. Dann ruft sie den absteigenden Rekursivalgorithmus (Ab_Rec_Pert) vom Grad 5 auf. Der Algorithmus geht auf der Strecke von $(1,1,1,1,1,1)$ nach $(2401,343,49,7,1,0)$ und braucht 68 Schritte, um die lexikographische reduzierte Gröbnerbasis auszuliefern.

Der erste Alternativalgorithmus benötigt insgesamt 1.71 Sekunden und 3333 Kilobyte Speicher. Davon werden 0.37 Sekunden (21.64%) zur Berechnung der reduzierten Gröbnerbasis von I bzgl. $\succ_{rlex}$ gebraucht. Die restliche Zeit verteilt sich auf die in Tabelle 4.10 aufgelisteten Prozeduren. □

Nach 2675 Minuten liefert der verbesserte Buchbergeralgorithmus keine Ausgabe zurück. In diesem Fall ist der erste Alternativalgorithmus viel schneller als die direkte Berechnung der lexikographischen Gröbnerbasis.

Beispiel 4.4.4. *Wenden wir den ersten Alternativalgorithmus auf Beispiel* dessin2 *mit 10 Variablen an, geht die Hauptprozedur Ab_Rec_Walk auf der Strecke von* $(1,1,\ldots,1)$ *nach* $\tau := (1,0,\ldots,0)$ *und braucht 3 Schritte, um den Zielgewichtsvek-*

Algorithmus 4.4.2. ***Ab_Rec_Pert***(G, σ, τ, d);

Eingabe: *Zwei Gewichtsvektoren* σ *und* τ*,*
die reduzierte Gröbnerbasis G *eines Ideals* I *bzgl. der Ordnung* $\gg_\sigma$ *und eine ganze Zahl* d*.*

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl. der Ordnung* $\gg_\tau$*.*

if (G *ist die reduzierte Gröbnerbasis von* I *bzgl.* $\gg_\tau$)

then return(G);

Initialisierung:

- $A :=$ *ist eine Matrix, die die Ordnung* $\gg_\tau$ *definiert.*
- $\tau' :=$ *PertVektor*(G, A, d);
- *endstep* $:= 0$;

while *wahr* ***do***

1. $\omega :=$ *NächsterVektor*(G, σ, τ');
2. ***if*** (ω *ist undefiniert*) ***then break;***
3. ***if*** $(\omega == \tau')$ ***then*** *endstep* $:= 1$;
4. ***if*** (*eine oder mehrere Komponenten von* ω *sind größer als* 2^{31}) ***then***

   if $(d \neq 1)$ ***then***
   return$(Ab\_Rec\_Pert(G, \sigma, \tau, d-1))$;
   else return$(RedGB\_BA(G, \succ_{\omega_2}))$;
5. $G_\omega :=$ *InitialForm*(G, ω);
6. $M :=$ *ReduziertGB*$(G_\omega, \gg_\omega)$;
7. $F :=$ *LiftGB*$(G, G_\omega, M, \gg_\omega)$;
8. $G :=$ *Reduzierung*$(F, \gg_\omega)$;
9. ***if*** *(endstep* $== 1$*)* ***then break;***
10. $\sigma := \omega$;

if $(\tau' \notin cone(G, \gg_\tau))$ ***then***

1. ***if*** $(d \neq 1)$ ***then***

   return$(Ab\_Rec\_Pert(G, \sigma, \tau, d-1))$;
2. ***else return***$(RedGB\_BA(G, \gg_\tau))$;

return(G);

Algorithmus 4.4.2: *Absteigender Rekursivalgorithmus vom Grad d.*

Prozedur	Zeit	
	in Sek.	%
InitialForm	0.03	1.75
ReduziertGB	0.42	24.56
LiftGB	0.53	30.99
Reduzierung	0.29	16.96
NächsterVektor	0.03	1.75

Tabelle 4.10: Die Laufzeitanalyse des ersten Alternativalgorithmus für *cyclic6*.

tor τ zu erreichen. Dann ruft sie den absteigenden Rekursivalgorithmus vom Grad 9 auf. Der letzte Algorithmus geht der auf der Strecke von

$$(2,1,1,1,1,1,1,1,1,1)$$

nach

$$\tau'_{(9)} := (65536, 16384, 4096, 1024, 256, 64, 16, 4, 1, 0),$$

wobei $\tau'_{(9)}$ der gestörte Gewichtsvektor des Zielgewichtsvektors τ vom Grad 9 ist. Er benötigt 38 Schritte, um $\tau'_{(9)}$ zu erreichen. Da $\tau'_{(9)}$ nicht im entsprechenden Gröbnerkegel liegt, ruft er den absteigenden Rekursivalgorithmus vom Grad 8 auf.

Der letzte Rekursivalgorithmus geht auf der Strecke von

$$(65536, 16384, 4096, 1024, 256, 64, 16, 4, 1, 0)$$

nach dem gestörten Gewichtsvektor des Zielgewichtsvektors τ vom Grad 8

$$\tau'_{(8)} := (170859375, 11390625, 759375, 50625, 3375, 225, 15, 1, 0, 0).$$

Im neunten Schritt sind eine oder mehrere Komponenten des berechneten dazwischenliegenden Gewichtsvektors allerdings größer als 2^{31}. Deswegen ruft der Algorithmus den absteigenden Rekursivalgorithmus vom Grad 7 auf. Der letzte Rekursivalgorithmus benötigt nur zwei Schritte, um die reduzierte lexikographische Gröbnerbasis zurückzuliefern.

Der erste Alternativalgorithmus benötigt insgesamt 67.13 Sekunden, 52 Schritte und 8946 Kilobyte Speicher. Davon werden 1.18 Sekunden zur Berechnung der reduzierten Gröbnerbasis bzgl. $\succ_{rlex}$ gebraucht. Die restliche Zeit verteilt sich auf die in Tabelle 4.11 aufgelisteten Prozeduren. □

Prozedur	Zeit	
	in Sek.	%
InitialForm	0.02	0.03
ReduziertGB	3.53	5.26
LiftGB	54.41	81.05
Reduzierung	7.91	11.79
NächsterVektor	0.00	0.00

Tabelle 4.11: Die Laufzeitanalyse des ersten Alternativalgorithmus für *dessin2*.

Beispiel 4.4.5. *Wenden wir den ersten Alternativalgorithmus auf das in (4.3) auf Seite 80 beschriebene Ideal I zur Berechnung der reduzierten Gröbnerbasis bzgl. $\succ_{lex}$ an, wird die gewünschte reduzierte Gröbnerbasis in 181.23 Sekunden erreicht. Davon werden 116.51 Sekunden (64.29%) zur Berechnung der reduzierten Gröbnerbasis von I bzgl. $\succ_{rlex}$ gebraucht. Die restliche Zeit verteilt sich auf die Prozedur InitialForm (0.55%), ReduziertGB (3.97%), LiftGB (11.95%), Reduzierung (18.28%) und NächsterVektor (0.31%).*

Der erste Alternativalgorithmus funktioniert wie folgt: Die Hauptprozedur Ab_Rec_Walk benötigt einen Schritt, um den Zielgewichtsvektor $\tau = (1, 0, \ldots, 0) \in \mathbb{N}^8$ zu erreichen. Sie ruft dann die Prozedur Ab_Rec_Pert vom Grad 7 auf. Der absteigende Rekursivalgorithmus vom Grad 7 geht auf der Strecke von

$$(1, 1, 1, 1, 1, 1, 1, 1)$$

nach

$$\tau'_{(7)} = (117649, 16807, 2401, 343, 49, 7, 1, 0)$$

und benötigt 63 Schritte, um $\tau'_{(7)}$ zu erreichen. Da dieser Vektor im richtigen Gröbnerkegel liegt, terminiert der Algroithmus und liefert die gewünschte reduzierte Gröbnerbasis zurück. □

Im Allgemeinen können wir feststellen:

1. Der erste Alternativalgorithmus vom Grad $(n-1)$ ist schneller als von den anderen Graden (siehe z.B. Tabelle 4.12).

2. Je größer der Grad ist, desto mehr Schritte braucht der erste Alternativalgorithmus (siehe z.B. Tabelle 4.13).

3. Die meiste Zeit ist zur Reduzierung einer Gröbnerbasis nötig (siehe Tabellen 4.14 und F.3 für den ersten Alternativalgorithmus vom Grad $(n-1)$).

Bsp.	n	Erster Alternativalgorithmus vom Grad						
		1	2	3	4	5	6	7
ex2	3	1698.66	**0.58**	0.59	—	—	—	—
ex3	3	6999.66	**0.48**	0.49	—	—	—	—
kats6†	6	>3364.77	>2751.51	>7012.00	3427.52	**1.07**	1.10	—
s7	7	>1500.27	>1680.25	>1215.35	>1247.08	3.35	0.82	**0.51**

Tabelle 4.12: Berechnungszeiten in Sekunden (†$> t$ bedeutet, dass nach t Sekunden bricht die Berechnung der reduzierten Gröbnerbasis eines ω-homogenen Ideals wegen des Speichermangels ab;).

Bsp.	n	Erster Alternativalgorithmus vom Grad						
		1	2	3	4	5	6	7
ex2	3	11	10+32	10+24†+10	—	—	—	—
ex3	3	13	12+34	12+29†+7	—	—	—	—
kats6	6	2	1+3	1+6	1+13	1+43	1+27†+19	—
s7	7	2	1+5	1+10	1+22	1+59	1+105	1+91†+2

Tabelle 4.13: Die Anzahl der Schritte ($\dagger\tau'$ liegt nicht im richtigen Gröbnerkegel).

Bsp.	Grad	Laufzeit (%) zur					Speicher
		InFo	sG_ω	Lift	InRd	NäVe	
ex2	2	0.00	6.90	12.07	77.59	0.00	1317
ex3	2	4.17	6.25	8.33	70.83	4.17	2346
kats6	5	0.65	2.61	9.15	80.39	0.65	1920
s7	6	7.32	32.93	17.07	24.39	6.10	3341

Tabelle 4.14: Die Laufzeitanalyse des ersten Alternativalgorithmus.

4.4.2 Zweiter Alternativalgorithmus

Beispiel 4.4.6. *(**chemkin** in http://www.symbolicdata.org) Wenden wir den ersten Alternativalgorithmus zur Berechnung der reduzierten Gröbnerbasis des Ideals*

$$\begin{aligned} I = \langle\, & z_2+z_3+z_4+z_5,\ 33461c_3-94642,\ c_2+y_2+y_3+y_4+y_5,\\ & 9c_2-8,\ 36926c_1-208885,\ -3c_3y_2+3x_3+3x_4+8,\\ & -c_1y_2+9y_2^2+z_2,\ z_5^2+y_5^2-c_2,\ 3z_4z_5+3y_4y_5+x_4-1,\\ & z_4^2+y_4^2+x_4^2-1,\ 3z_3z_4+3y_3y_4+3x_3x_4-1,\\ & z_3^2+y_3^2+x_3^2-1,\ -3c_3y_2x_3+3z_2z_3+3y_2y_3+3x_3-1\,\rangle \end{aligned}$$

in $R := \mathbb{Q}[c_1,c_2,c_3,z_2,z_3,z_4,z_5,y_2,y_3,y_4,y_5,x_3,x_4]$ *bzgl.* $\succ_{lex}$ *auf* R *mit*

$$c_1 > c_2 > c_3 > z_2 > z_3 > z_4 > z_5 > y_2 > y_3 > y_4 > y_5 > x_3 > x_4$$

an, liefert der Algorithmus nach 1956 *Minuten noch kein Ergebnis zurück.*

Im ersten Schritt der Hauptprozedur Ab_Rec_Walk wird der Zielgewichtsvektor $\tau := (1,0,\ldots,0) \in \mathbb{N}^{13}$ erreicht. Deswegen wird die Prozedur Ab_Rec_Pert vom Grad 12 aufgerufen und geht auf der Strecke von $\sigma := (1,1,\ldots,1)$ nach dem durch die Störung von τ mit Grad 12 gegebenen Gewichtsvektor

$$\tau'_{(12)} := (177147,59049,19683,6561,2187,729,243,81,27,9,3,1,0).$$

Diese Prozedur ist leider sehr aufwendig und liefert nach 1956 Minuten noch kein Ergebnis zurück

In diesem Abschnitt stellen wir nun den zweiten Alternativalgorithmus vor. Der Algorithmus funktioniert folgendermaßen: Zuerst wird der Gröbnerwalkalgorithmus aufgerufen. Wenn ein dazwischenliegender Gewichtsvektor gleich dem Zielgewichtsvektor ist, wird der Perturbationwalkalgorithmus vom Perturbationgradpaar $(1,2)$ aufgerufen. Falls im Perturbationwalkalgorithmus vom Perturbationgradpaar $(1,2)$

eine von der drei Möglichkeiten auftritt: (1) der erreichte dazwischenliegende Gewichtsvektor ist gleich dem aktuellen Zielgewichtsvektor, (2) eine Komponente des berechneten dazwischenliegenden Gewichtsvektors ist größer als 2^{31} oder (3) der gestörte Zielgewichtsvektor liegt nicht im entsprechenden Gröbnerkegel, wird der Perturbationwalkalgorithmus vom Perturbationgradpaar $(1,3)$ aufgerufen. Dies wird wiederholt, bis die gewünschte Gröbnerbasis erreicht oder das Perturbationgradpaar gleich $(1,n)$ ist, wobei n die Anzahl der Variablen ist. Bei jedem Aufruf eines solchen Perturbationwalkalgorithmus wird getestet, ob die Eingabe des Algorithmus bereits die gewünschte Gröbnerbasis ist. Wenn ja, terminiert der Algorithmus und liefert seine Eingabe zurück. Schließlich wird der Buchbergeralgorithmus erst aufgerufen, falls das Perturbationgradpaar gleich $(1,n)$ ist und einer der zwei folgenden Fällen auftritt:

1. Der letzte dazwischenliegende Gewichtsvektor ist gleich dem gestörten Zielgewichtsvektor.

2. Der gestörte Zielgewichtsvektor liegt nicht im entsprechenden Gröbnerkegel.

Den Algorithmus 4.4.7 bezeichnen wir als **zweiten Alternativalgorithmus**.

Algorithmus 4.4.7. ***Auf_Rec_Walk***(G,σ,τ);

Eingabe: *Zwei Gewichtsvektoren σ und τ, die reduzierte Gröbnerbasis G eines Ideals I bzgl. der Ordnung $\succ$.*

Ausgabe: *Die reduzierte Gröbnerbasis von I bzgl. der Ordnung $\gg$, die τ verfeinert.*

Initialisierung: $\omega := \sigma$;

while *wahr* ***do***

1. $G_\omega := InitialForm(G,\omega)$;
2. $M := ReduziertGB(G_\omega, \gg_\omega)$;
3. $F := LiftGB(G, G_\omega, M, \gg_\omega)$;
4. $G := Reduzierung(F, \gg_\omega)$;,
5. $\omega := NächsterVektor(G,\sigma,\tau)$;
6. ***if*** $(\omega == \tau)$ ***then***

   return$(Auf\_Rec\_Pert(G,\sigma,\tau,2))$;
7. ***if*** (ω *ist undefiniert*) ***then return***(G);
8. $\sigma := \omega$;

Algorithmus 4.4.7: *Der zweite Alternativalgorithmus.*

Algorithmus 4.4.8. ***Auf_Rec_Pert***(G, σ, τ, d);

Eingabe: *Zwei Gewichtsvektoren* σ *und* τ, *die reduzierte Gröbnerbasis* G *eines Ideals* I *mit* n *Variablen bzgl. der Ordnung* $\gg_\sigma$ *und eine ganze Zahl* $1 \leq d \leq n$.

Ausgabe: *Die reduzierte Gröbnerbasis von* I *bzgl. der Ordnung* $\gg_\tau$.

if (G *ist die reduzierte Gröbnerbasis von* I *bzgl.* $\gg_\tau$)

then return(G);

Initialisierung:

- $A :=$ *ist eine Matrix, die die Ordnung* $\gg_\tau$ *definiert.*
- $\tau' :=$ $PertVektor(G, A, d)$;
- $\omega := \sigma$;

while *wahr* ***do***

1. $\omega :=$ $NächsterVektor(G, \sigma, \tau')$;
2. ***if*** ($\omega ==$ τ' *oder* (*eine oder mehrere Komponenten von* ω *sind größer als* 2^{31})) ***then***
   (a) ***if*** $(d \neq n)$ ***then***
   return$(Auf\_Rec\_Pert(G, \sigma, \tau, d+1))$;
   (b) ***else*** $G :=$ $RedGB\_BA(G, \gg_\tau)$; ***break;***
3. ***if*** (ω *ist undefiniert*) ***then break;***
4. $G_\omega :=$ $InitialForm(G, \omega)$;
5. $M :=$ $ReduziertGB(G_\omega, \gg_\omega)$;
6. $F :=$ $LiftGB(G, G_\omega, M, \gg_\omega)$;
7. $G :=$ $Reduzierung(F, \gg_\omega)$;
8. $\sigma := \omega$;

if $(\tau' \notin cone(G, \gg_\tau))$ ***then***

1. ***if*** $(d \neq n)$ ***then***
   return$(Auf\_Rec\_Pert(G, \sigma, \tau, d+1))$;
2. ***else return***$(RedGB\_BA(G, \gg_\tau))$;

return(G);

Algorithmus 4.4.8: *Aufsteigender Rekursivalgorithmus vom Grad* d.

Prozedur	Zeit	
	in Sek.	%
InitialForm	0.02	0.76
ReduziertGB	1.14	43.35
LiftGB	0.61	23.19
Reduzierung	0.21	7.98
NächsterVektor	0.02	0.76
andere	0.20	4.32

Tabelle 4.15: Die Laufzeitanalyse des zweiten Alternativalgorithmus für *cyclic6*.

Beispiel 4.4.9. *Wir betrachten wieder das Ideal im Beispiel 4.1.5 (cyclic 6) und berechnen die lexikographische reduzierte Gröbnerbasis mit Hilfe des zweiten Alternativalgorithmus.*

Die Hauptprozedur Auf_Rec_Walk braucht nur einen Schritt, um den Zielgewichtsvektor $\tau := (1,0,0,0,0,0)$ zu erreichen. Dann ruft sie die Prozedur Auf_Rec_Pert vom Grad 2 auf.

Die Prozedur Auf_Rec_Pert vom Grad 2 geht auf der Strecke von $(1,1,1,1,1,1)$ nach $\tau'_{(2)} := (10,1,0,0,0,0)$ und braucht 2 Schritte, um $\tau'_{(1)}$ zu erreichen. Diese Prozedur ruft dann die Prozedur Auf_Rec_Pert vom Grad 3 auf.

Die Prozedur Auf_Rec_Pert vom Grad 3 geht auf der Strecke von $(61,7,1,1,1,1)$ nach $\tau'_{(3)} := (441,21,1,0,0,0)$ und braucht nur 4 Schritte, um $\tau'_{(2)}$ zu erreichen. Diese Prozedur ruft dann die Prozedur Auf_Rec_Pert vom Grad 4 auf.

Die Prozedur Auf_Rec_Pert vom Grad 4 geht auf der Strecke von $(2707,133,7,1,1,1)$ nach $\tau'_{(4)} := (729,81,9,1,0,0)$ und braucht nur 8 Schritte, um τ_3 zu erreichen. Diese Prozedur ruft dann die Prozedur Auf_Rec_Pert vom Grad 5 auf.

Die Prozedur Auf_Rec_Pert vom Grad 5 geht auf der Strecke von $(15829,1591,169,19,1,1)$ nach $\tau'_{(5)} := (130321,6859,361,19,1,0)$ und braucht nur 27 Schritte, um $\tau'_{(5)}$ zu erreichen. Sie ruft dann die Prozedur Auf_Rec_Pert vom Grad 6 auf.

Da die Eingabe der Prozedur Auf_Rec_Pert vom Grad 6 bereits die gewünschte reduzierte lexikographische Gröbnerbasis ist, terminiert der Algorithmus und liefert diese Gröbnerbasis zurück.

Der Algorithmus braucht insgesamt 2.63 Sekunden und 2308 Kilobyte Speicher. Davon sind 0.57 Sekunden (21.67%) zur Berechnung der reduzierten Gröbnerbasis von I bzgl. $\succ_{rlex}$ nötig. Tabelle 4.15 zeigt den restlichen Zeitverbrauch. □

Nach 2675 Minuten liefert der verbesserte Buchbergeralgorithmus noch keine Ausgabe. In diesem Fall ist der zweite Alternativalgorithmus viel schneller als die direkte Berechnung der lexikographischen Gröbnerbasis.

Beispiel 4.4.10. *Wird der zweite Alternativalgorithmus auf Beispiel 4.4.6 (chemkin) angewandt, liefert er in 456.84 Sekunden die reduzierte lexikographische Gröbnerbasis, die aus 13 Polynomen mit 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 2, 2 bzw. 2 Monomen besteht. Davon sind 9.74 Sekunden (2.13%) zur Berechnung der reduzierten Gröbnerbasis bzgl. $\succ_{rlex}$ nötig. Die restliche Zeit verteilt sich auf die*

Prozedur InitialForm (0.02%), ReduziertGB (70.09%), LiftGB (16.89%), Reduzierung (10.82%) und NächsterVektor (0.01%).

Der Algorithmus geht wie folgt vor: Die Hauptprozedur Auf_Rec_Walk geht auf der Strecke von $\sigma := (1, 1, \ldots, s) \in \mathbb{N}^{13}$ *nach* $\tau := 1, 0, \ldots, 0) \in \mathbb{N}^{13}$ *und benötigt nur einen Schritt, um den Zielgewichtsvektor* τ *zu erreichen. Dann ruft sie die Prozedur Auf_Rec_Pert vom Grad 2 auf.*

Diese Prozedur geht von σ *nach* $\tau'_{(2)} := (5, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. *Im ersten Schritt wird* $\tau'_{(2)}$ *erreicht. Deswegen ruf sie dann die Prozedur Auf_Rec_Pert vom Grad 3 auf. Diese ruft dann die Prozedur Auf_Rec_Pert vom Grad 4 auf, da* $\tau'_{(3)} := (81, 9, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ *im ersten Schritt erreicht wird.. Der aktuelle Zielgewichtsvektor* $\tau'_{(4)} := (27, 9, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ *wird im ersten Schritt erreicht.*

Die Prozedur Auf_Rec_Pert vom Grad 5 wird dann aufgerufen. Sie geht auf der Strecke von σ *nach* $\tau'_{(5)} := (81, 27, 9, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0)$ *und benötigt zwei Schritte, um* $\tau'_{(5)}$ *zu erreichen.*

Die Prozedur Auf_Rec_Pert vom Grad 6 wird aufgerufen und geht auf der Strecke von

$$\omega_1^{(5)} := (82, 28, 10, 4, 2, 1, 1, 1, 1, 1, 1, 1, 1)$$

nach

$$\tau'_{(6)} := (243, 81, 27, 9, 3, 1, 0, 0, 0, 0, 0, 0, 0).$$

Sie benötigt drei Schritte, um $\tau'_{(6)}$ *zu erreichen.*

Die Prozedur Auf_Rec_Pert vom Grad 7 wird aufgerufen und geht auf der Strecke von

$$\omega_2^{(6)} := (568, 190, 64, 22, 8, 3, 1, 1, 1, 1, 1, 1, 1)$$

nach

$$\tau'_{(7)} := (729, 243, 81, 27, 9, 3, 1, 0, 0, 0, 0, 0, 0).$$

Sie benötigt auch drei Schritte, um $\tau'_{(7)}$ *zu erreichen.*

Die Prozedur Auf_Rec_Pert vom Grad 8 geht auf der Strecke von

$$\omega_2^{(7)} := (2755, 919, 307, 103, 35, 12, 4, 1, 1, 1, 1, 1, 1)$$

nach

$$\tau'_{(8)} := (16384, 4096, 1024, 256, 64, 16, 4, 1, 0, 0, 0, 0, 0)$$

und benötigt nur einen Schritt, um $\tau'_{(8)}$ *zu erreichen.*

Dewegen geht die aufgerufene Prozedur Auf_Rec_Pert vom Grad 9 auf der Strecke von $\omega_2^{(7)}$ *nach*

$$\tau'_{(9)} := (65536, 16384, 4096, 1024, 256, 64, 16, 4, 1, 0, 0, 0, 0)$$

und benötigt auch nur einen Schritt, um $\tau'_{(9)}$ *zu erreichen.*

Die Prozedur Auf_Rec_Pert vom Grad 10 wird aufgerufen und geht auf der Strecke von $\omega_2^{(7)}$ *nach*

$$\tau'_{(10)} := (262144, 65536, 16384, 4096, 1024, 256, 64, 16, 4, 1, 0, 0, 0).$$

Im 12-ten Schritte wird $\tau'_{(10)}$ erreicht. Dann ruft sie die Prozedur Auf_Rec_Pert vom Grad 11 auf. In der letzten Prozedur ist eine Komponente des gestörten Zielgewichstvektor aber größer als 2^{31}. An dieser Stelle erhält man nur die reduzierte Gröbnerbasis F des gegebenen Ideals bezüglich der $\omega_{11}^{(10)}$-gewichteten lexikographischen Ordnung mit

$$\omega_{11}^{(10)} := (69148562, 13869402, 2783834, 559266, 112482, 22655, 4571, 924, 188, 39, 8, 1, 1).$$

Auf das Ideal $\langle F \rangle$ verwenden wir den Buchbergeralgorithmus zur Berechnung der reduzierten lexikographischen Gröbnerbasis.

Beispiel 4.4.11. *Betrachten wir wieder das Beispiel 4.1.4 und wenden die beiden Alternativalgorithmen auf dem Ideal (4.2) an, gehen die Algorithmen auf einer ähnlichen Strecke. Die Hauptprozedur geht zuerst von $(1,1,1)$ nach $(1,0,0)$. Im 15-ten Schritt, in dem der nächste dazwischenliegende Gewichtvektor von $(16,1,1)$ nach $(1,0,0)$ bestimmt werden muss, wird der Zielgewichtsvektor erreicht. Aus diesem Grund gehen die beiden Algorithmen von $(16,1,1)$ nach $(24,1,0)$, der durch die Störung von $(1,0,0)$ mit Grad 2 gegeben ist. Da dieses Unterprogramm die gesuchte reduzierte Gröbnerbasis liefert, terminiert der Algorithmus.*

Kapitel 5

Analyse der Ergebnisse

Den Gröbnerwalkalgorithmus und die im Kapitel 4 vorgestellten Algorithmen haben wir im Computeralgebrasystem SINGULAR implementiert und mit zahlreichen Beispielen getestet. Gleichzeitig haben wir auch diese Algorithmen mit den folgenden in SINGULAR implementierten Varianten des Buchbergeralgorithmus verglichen:

1. Der Singularbefehl **std** liefert eine nicht notwendig reduzierte Gröbnerbasis eines beliebigen Ideals oder Moduls bzgl. einer monomialen Ordnung zurück. Eine solche Gröbnerbasis wird direkt mit einer Variante des Buchbergeralgorithmus ausgerechnet.

2. Der Singularbefehl **stdhilb** liefert eine durch den Hilbert-driven Buchbergeralgorithmus ausgerechnete Gröbnerbasis eines homogenen Ideals I zurück.

3. Der Singularbefehl **stdfglm** liefert eine durch den FGLM Algorithmus ausgerechnete Gröbnerbasis eines *nulldimensionalen* Ideals zurück.

4. Der Singularbefehl **groebner** liefert eine Gröbnerbasis eines beliebigen Ideals I durch eine heuristisch gewählte Methode zurück. Die auf einer solchen Methode basierende Variante nennen wir dann *Groebneralgorithmus*.

   Zum Beispiel, zur Berechnung einer Gröbnerbasis von I bzgl. der lexikographischen Ordnung $\succ_{lex}$ auf dem Polynomring $K[x_1, \ldots, x_n]$ über dem Körper K geht **groebner**(I) folgendermaßen vor: Ist I ein homogenes Ideal, so wird die gewünschte Gröbnerbasis durch den Hilbert-driven Buchbergeralgorithmus ausgerechnet. Ansonsten wird I in einen Ring mit Ordnung $\succ_{rlex}$ abgebildet und homogenisiert. Wir bezeichnen mit I_h das homogenisierte Ideal. In diesem Ring wird die Hilbert-Reihe H von I_h ausgerechnet. Mit Hilfe des Hilbert-driven Buchbergeralgorithmus wird dann eine Gröbnerbasis G von I bezüglich $\succ_{lex}$ ausgerechnet, deren Hilbert-Reihe H ist. Die gewünschte lexikographische Gröbnerbasis wird durch die Dehomogenisierung von G erreicht.

In unseren Experimenten verwendeten wir stets den Singularbefehl **option(redSB)** der sicherstellt, dass die Eingabe **std**(I), **stdhilb**(I), **stdfglm**(I) und **groebner**(I) die reduzierte Gröbnerbasis eines Ideals I zurückliefert.

SINGULAR besitzt eine eigene Interpretersprache. Aus Geschwindigkeitgründen wurde jedoch von Beginn an eine Implementierung der Algorithmen in **C** oder **C++**

im SINGULAR-Kern angestrebt. Deswegen haben wir alle in Kapitel 3 und 4 vorgestellten Algorithmen direkt als **C**-Kernfunktionen implementiert.

Gewichtsvektoren spielen in den auf dem Gröbnerwalkalgorithmus basierenden Algorithmen eine große Rolle. In großen Beispielen werden oft eine oder mehrere Komponenten eines dazwischenliegenden Gewichtsvektors größer als 2^{31} sein. Damit unsere Implementierungen in diesem Fall noch die gesuchte Gröbnerbasis zurückliefern könnten, können wir den Buchbergeralgorithmus oder den Algorithmus 4.4.2 auf das von der bereits berechneten Gröbnerbasis erzeugte Ideal verwenden. Um eine dieser Möglichkeiten zu wählen, kann der Benutzer eine ganze Zahl d in jedem in der Standardbibliothek *groebnerwalk.lib* bereitgestellten Befehl eingeben. Diese Zahl, falls sie gegeben ist, gehört zu einem Singularobjekt L vom Typ *list*. Ist $d = 0$, so wird der Buchbergeralgorithmus gewählt. Ansonsten ist es Algorithmus 4.4.2. L ist eine geordnete Menge $\{\}, \{d\}, \{\sigma\}, \{\sigma, d\}, \{\sigma, \tau\}, \{\sigma, \tau, d\}$, wobei σ und τ zwei nichtnegative ganze Gewichtsvektoren sind. Wir bezeichnen dann σ bzw. τ als den Start- bzw. Zielgewichtsvektor.

Bemerkung 5.0.1. *Ist σ eingegeben, so berechnet der aufgerufene Algorithmus zuerst die reduzierte Gröbnerbasis des gegebenen Ideals I bzgl. der σ-gewichteten lexikographischen Ordnung. Ansonsten wird die reduzierte Gröbnerbasis von I bzgl. der Ordnung $\succ_{rlex}$ ausgerechnet. Dann formt der Algorithmus diese Gröbnerbasis in die reduzierte Gröbnerbasis bzgl. der Zielordnung um, nämlich entweder $\succ_{lex}$ oder der τ-gewichteten lexikographischen Ordnung, falls τ eingegeben ist.*

Falls in der obigen Berechnung eine oder mehrere Komponenten eines verwendeten dazwischenliegenden Gewichtsvektors größer als 2^{31} sind oder der gestörte Zielgewichtsvektor nicht im richtigen Gröbnerkegel liegt, wird der Algorithmus 4.4.2 auf das von der bereits berechneten Gröbnerbasis erzeugte Ideal zur Berechnung der gesuchten Gröbnnerbasis angewendet. Dies wird ausgeführt, falls $d \neq 0$ eingegeben ist. Ansonstens wird der Algorithmus 4.4.2 durch den Buchbergeralgorithmus ersetzt.

Um unsere Implementierungen durchführen zu können, werden in *groebnerwalk.lib* die folgenden Befehle bereitgestellt:

1. **ideal = gwalk(ideal** I**, list** L**);**
   Dies ruft die Variante *ngwalk* des Gröbnerwalkalgorithmus (siehe Seite 99) auf.

2. **ideal = pwalk(ideal** I**, int** $\mathbf{d_1}$**, int** $\mathbf{d_2}$**, list** L**);**
   Dies ruft den Perturbationwalkalgorithmus vom Perturbationgradpaar (d_1, d_2) auf. Ist $d_1 = d_2 = 1$, ruft dies den „originalen“ Gröbnerwalkalgorithmus, nämlich Algorithmus 3.3.1, auf.

3. **ideal = fwalk(ideal** I**, list** L**);**
   Dies ruft den Fractalwalkalgorithmus auf.

4. **ideal = twalk(ideal** I**, list** L**);**
   Dies ruft den Tranalgorithmus auf.

5. **ideal = awalk1(ideal** I**, int** p**, list** L**);**
   Dies ruft den ersten Alternativalgorithmus vom Grad p auf. Ist die ganze Zahl p nicht eingegeben und n ist die Anzahl der Varablen von I, so wird die Stelle p durch $n - 1$ ersetzt.

6. **ideal = awalk2(ideal I, list L);**
   Dies ruft den zweiten Alternativalgorithmus auf.

In diesem Abschnitt präsentieren wir nun die Ergebnisse unserer Experimente mit den in SINGULAR implementierten Algorithmen zur Berechnung der reduzierten Gröbnerbasis eines Ideals bzgl. einer lexikographischen monomialen Ordnung. Dies sind die Implementierung des Buchberger-, Hilbert-driven-, FGLM-, Groebneralgorithmus und der auf dem Gröbnerwalkalgorithmus basierenden Algorithmen. Genauer formuliert: Sei ein Erzeugendensystem eines Ideals $I \subset R := K[x_1, \ldots, x_n]$ mit $K := \mathbb{Q}$ oder $K := \mathbb{Z}_{32003} := \mathbb{Z}/32003\mathbb{Z}$ gegeben. Dann berechne daraus die reduzierte Gröbnerbasis von I bzgl. der lexikographischen Ordnung $\succ_{lex}$ auf R. Falls nicht anders angegeben, wird zunächst im Fall der vom Buchbergeralgorithmus verschiedenen Algorithmen die reduzierte Gröbnerbasis von I bzgl. $\succ_{rlex}$ berechnet. Dies wird ausgeführt, da gewöhnlich das eingegebene Erzeugendensystem von I noch keine Gröbnerbasis ist. Deswegen enthält die Berechnungzeit jedes oben angegebenen Algorithmus auch die Berechnungzeit der Gröbnerbasis bzgl. $\succ_{rlex}$.

In den von Amrhein et. al bzw. Tran entwickelten Algorithmen werden einer oder zwei neue Gewichtsvektoren gewählt, um eine „effiziente“ Strecke zu finden. Diese Vektoren hängen nicht nur von der gewählten ganzen Zahl - entweder $\frac{1}{\epsilon}$ oder d - ab, sondern auch von dem Perturbationgrad bzw. der Anzahl der Variablen. Deswegen sind in mehreren Beispielen, in denen die Anzahl der Variablen größer als 3 ist, eine oder mehrere Komponenten eines während der Berechnung ergebenden Gewichtsvektors größer als 2^{31}. Das heisst, in SINGULAR kann die Berechnung nicht mit diesem großen Gewichtsvektor fortgesetzt werden (siehe Bemerkung 3.2.5).

Deswegen änderten wir in unseren Experimenten, deren Ergebnisse in diesem Abschnitt vorgestellt werden, den Algorithmus 4.1.1 zur Bestimmung des gestörten Gewichtsvketors vom Grad d bzw. der Repräsentation des Zielgewichtsvektors wie folgt ab: Die Zeile 4 des Algorithmus 4.1.1 wird durch den Quotient der Division von $me + 1$ durch d (bzw. n) ersetzt, falls $me + 1 \geq d \geq 4$ (bzw. $me + 1 \geq n \geq 4$) ist.

Diese Verfahrensweise garantiert jedoch nicht, dass jede Komponente eines berechneten dazwischenliegenden Gewichtsvektors bzw. eines gestörten Gewichtsvektors immer kleiner als 2^{31} ist. Falls ein Gewichtsvektor mit einer oder mehreren übertretenen Komponenten auftritt, kann man noch die im vorletzten Schritt berechneten Ergebnisse zur Berechnung der gesuchten Gröbnerbasis verwenden. Hierzu können wir den Buchbergeralgorithmus oder den rekursiven Algorithmus 4.4.2 verwenden (vgl. Bemerkung 5.0.1). Aus Effizienzgründen haben wir den zweiten Algorithmus gewählt.

Mehrere der in unseren Experimenten verwendeten Beispiele wurden auch in den Arbeiten von Amrhein et al. (1996, 1997, 1998)[1)], Tran (1998, 2000)[2)], Faugère et al. (1993)[3)] und Wichmann (1997)[4)] benutzt. Die Autoren haben festgestellt, dass in den Beispielen ihre Algorithmen im Allgemeinen schneller als der direkte

[1)] Sie benutzten Beispiele zum Vergleich des Gröbner-, Perturbation- und Fractalwalkalgorithmus mit dem Buchberger- oder FGLM-Algorithmus.

[2)] Er benutzte Beispiele zum Vergleich des „Tranalgorithmus“ mit dem Buchbergeralgorithmus.

[3)] Sie benutzten Beispiele zum Vergleich des FGLM-Algorithmus mit dem Buchbergeralgorithmus.

[4)] Er hat den FGLM-Algorithmus in SINGULAR implementiert und gezeigt, dass im Allgemeinen der FGLM-Algorithmus schneller ist als der Buchbergeralgorithmus.

Buchbergeralgorithmus sind. Um einen möglichst repräsentativen Vergleich anstellen zu können, benutzen wir zahlreiche Beispiele, die in den Webseiten [40] und [41][5)] zur Verfügung stehen und darüber hinaus Beispiele, die von uns konstruiert wurden. Ein paar Beispiele davon sind im Anhang A abgedruckt.

Falls nicht anders erwähnt, ist der Zeit- bzw. maximale Speicherverbrauch in Sekunden bzw. Kilobytes angegeben, und die Berechnungen wurden auf einem Computer mit Pentium IV 2.40 GHz Prozessor und 1 Gegabyte RAM durchgeführt. Wir benutzen die folgenden Bezeichnungen:

1. $> t$ M (bzw. St) bedeutet, dass nach t Minuten (bzw. Stunden) die Berechnung wegen des Speichermangels abbricht. Diese Bezeichnungen tauchen nur in der Spalte *Berechnungzeiten* einer Tabelle auf.

2. $>_z t$ M (bzw. St) bedeutet, dass nach t Minuten (bzw. Stunden) die Berechnung noch keine Ausgabe zurückliefert hat und von uns abgebrochen wurde. Dies wurde durchgeführt, weil dieser Algorithmus viel mehr Zeit als die anderen Algorithmen benötigt.

3. ∞, die in der Spalte *Speicherverbrauch* auftaucht, bedeutet, dass der verfügbare Speicher ausgenutzt ist und die Berechnung bricht wegen des Speichermangels ab.

4. $-$, die in der Spalte *fglm* auftaucht, bedeutet, dass das Ideal kein nulldimensionales Ideal ist.

Außerdem benutzen wir die folgenden Abkürzungen:

1. n ist die Anzahl der Variablen eines Polynomrings.

2. Bsp. ist die Abkürzung für Beispiele.

3. *gwalk* bzw. *ngwalk* stehen für den originalen Gröbnerwalkalgorithmus bzw. den neuen Gröbnerwalkalgorithmus.

4. *bpwalk* steht für den Perturbationwalkalgorithmus vom besten Perturbationgradpaar.

5. *fglm* bzw. *hilb* steht für den Singularbefehl **stdfglm** bzw. **stdhilb**.

5.1 Analyse des Gröbnerwalkalgorithmus

Der Gröbnerwalkalgorithmus wurde von Collart et al. (1997) entwickelt und von Amrhein et al. (1996a) in SACLIB implementiert. Gleichzeitig haben Amrhein et al. den Gröbnerwalkalgorithmus anhand mehrerer Beispiele mit dem Buchbergeralgorithmus zur Berechnung einer reduzierten lexikographischen Gröbnerbasis verglichen. Dabei haben sie festgestellt, dass der Gröbnerwalkalgorithmus viel schneller als der

[5)] Eines von mehreren Zielen des SymbolicData Projekts ist es, Begriffe und Serviceprogramme (tools) zu entwickeln und in verschiedenen Bereichen der Computeralgebra aufgetretene Daten zu sammeln. Diese Begriffe und Serviceprogramme werden zu Benchmark-Tests in der Computeralgebra benutzt.

Buchbergeralgorithmus ist, falls die „Schwierigkeit“ [6] der Beispiele ansteigt (siehe Tabelle 5.1).

In der Tabelle 5.1 sind die Berechnungzeiten des Gröbnerwalk- und Buchbergeralgorithmus abgedruckt. Die beiden Algorithmen wurden jeweils in SINGULAR (Pentium II 266 MHz Prozessor mit 32 Megabyte RAM) und SACLIB (SPARCstation 10/40 mit 32 Megabyte RAM) implementiert. Die Berechnungzeiten von SACLIB sind nach Amrhein et al. (1996a) zitiert. Die Beispiele zeigen, dass die Implementierung des Buchbergeralgorithmus in SINGULAR sehr effizient ist. Eine Ausnahme bildet das Beispiel **bsp6**.

Beispiele	n	SINGULAR		SACLIB	
		gwalk	std	gwalk	std
bsp1	3	0.02	0.00	3.33	0.03
bsp2	3	0.02	0.01	14.43	6.36
bsp3	3	0.25	0.02	2.32	27.32
bsp4	3	0.06	0.12	4.39	2951
trinks1	6	65.45	3.07	54.5	>300
canny	5	0.96	0.19	133	>300
canny2	5	4.06	0.13	269	>10 M
s6	6	1.10	0.11	3395	>2 St
bsp5	3	0.08	0.03	1687	(?)
bsp6	3	1.71	>10 St	29630	(?)
bsp7	3	12.76	42.49	1141†	(?)

Tabelle 5.1: Berechnungzeiten des Gröbnerwalk- und Buchbergeralgorithmus ((?) Die Berechnungzeit ist unbekannt, mindestens mehrere Stunden; † Die Berechnungzeit des Gröbnerwalkalgorithmus mit der Störung der beiden gegebenen Gewichtsvektoren).

Betrachten wir den Zeitverbrauch des Gröbnerwalkalgorithmus und der in SINGULAR implementierten Algorithmen, können wir folgendes feststellen:

1. Für nulldimensionale Ideale ist der Gröbnerwalkalgorithmus im Allgemeinen langsamer als der FGLM-Algorithmus (siehe z.B. Tabellen 5.2 und 5.6).

2. Im Allgemeinen ist der Gröbnerwalkalgorithmus viel schneller als der Buchbergeralgorithmus (siehe z.B. Tabelle 5.2, obere Hälfte).

   In unseren Experimenten haben wir ein paar Beispiele gefunden, in denen er noch langsamer ist als der Buchbergeralgorithmus (siehe z.B. Tabelle 5.2, untere Hälfte) oder die beiden Algorithmen keine Ausgabe zurückliefern können, weil die Berechnungen mehr Speicher als verfügbar benötigen. In diesen Fällen brechen sie dann ab (siehe Tabelle 5.6).

3. In jedem Schritt des Gröbnerwalkalgorithmus muss die reduzierte Gröbnerbasis eines ω-homogenen Ideals $\langle I_\omega \rangle$ bzgl. der Ordnung $\gg_\omega$ ausgerechnet werden. Im letzten Schritt des Algorithmus haben wir die folgende Situation: G_{t-1} ist die bereits berechnete reduzierte Gröbnerbasis des gegebenen Ideals I

[6] Die Schwierigkeit eines Beispiels bezieht sich auf den Zeitverbrauch des Buchbergeralgorithmus für das Beispiel. Das Beispiel A ist *schwieriger* als B, falls der Buchbergeralgorithmus für A mehr Zeit als für B benötigt, vgl. Amrhein et al. (1996).

Bsp.	n	std	groebner	fglm	gwalk
tran1	3	564.31	0.09	—	**0.05**
bsp6	3	$>_z$106 M	**0.12**	—	0.23
bsp7	3	8.68	**0.94**	—	2.28
rose	3	150.33	1.79	2.44	**0.98**
qnt3	3	350.05	**0.01**	0.04	0.75
rose2	3	$>_z$120 M	26.17	8.20	**2.98**
cassou	4	$>_z$120 M	0.36	**0.27**	0.75
exam14	4	$>_z$445 M	**0.06**	0.24	171.20
vermeer76	5	$>_z$8984.70	0.55	—	**0.10**
jhd6inf	6	>6272.80	1.36	—	**0.17**
eco7	7	$>_z$158 M	3.82	**0.56**	138.08
omdi	9	261.41	205.30	**0.29**	0.40
el44	9	995.72	$>_z$337 M	**6.78**	6.67
hieta1	10	>2370.77	258.55	**0.40**	1.79
hairer	13	114.10	454.62	—	**5.94**
arnborg7	3	0.03	**0.00**	0.02	0.14
exam12	3	0.99	**0.06**	0.37	145.71
exam46	4	1.44	**0.17**	0.46	4.74
noon4	4	2.30	**0.23**	0.58	22.49
issac97	4	0.31	**0.19**	0.38	524.57
Mod	5	0.03	0.02	0.01	6.52
s6	6	0.02	0.02	**0.01**	0.16
trinks1	6	1.05	0.06	**0.08**	18.96
kats5	6	0.19	**0.02**	0.02	683.79
redeco7	7	1.03	**0.21**	0.27	4.13
s7	7	161.92	**0.14**	0.31	>1500.27

Tabelle 5.2: Berechnungzeiten: Gröbnerwalkalgorithmus und andere Algorithmen.

bzgl. der ω_{t-1}-gewichteten lexikographischen Ordnung $\gg_{\omega_{t-1}}$, wobei ω_{t-1} der vorletzte berechnete dazwischenliegende Gewichtsvektor ist. In diesem Schritt wird der nächste berechnete dazwischenliegende Gewichtsvektor ω_t gleich dem Zielgewichtsvektor $\tau = (1, 0, \ldots, 0) \in \mathbb{Z}^n$ sein. Dann berechnet man die reduzierte Gröbnerbasis M_t des ω_t-homogenen Ideals $\langle I_{\omega_t} \rangle = \langle (G_{t-1})_{\omega_t} \rangle$ bzgl. der Ordnung $\gg_{\omega_t}$. Da die meisten Polynome von G_{t-1} nicht die erste Variable enthalten, gehören diese Polynome zu $(G_{t-1})_{\omega_t}$. Außerdem stimmen die Initialmonome mehrerer Polynome von $(G_{t-1})_{\omega_t}$ bzgl. $\gg_{\omega_t}$ überein (siehe Anhang B und C). Deswegen ist diese Berechnung sehr aufwendig. Also benötigt der Gröbnerwalkalgorithmus in diesem Schritt viel Zeit und Speicher. In den meisten von uns berechneten Beispielen bricht die Berechnung nach einiger Zeit wegen des Speichermangels ab (siehe z.B. Tabellen 5.3 und 5.4). Deshalb sucht man nach Alternativen um den Gröbnerwalkalgorithmus schneller und speicherschonender zu machen. Es wurden die folgenden Alternativen entwickelt:

(a) Amrhein et al. (1996, 1997 und 1998) entwickelten den Perturbation- bzw. Fractalwalkalgorithmus. Im letzten Algorithmus wird die reduzierte Gröbnerbasis von $\langle I_\omega \rangle$ bzgl. $\gg_\omega$ durch den Perturbationwalkalgorithmus

mit dem rekursiven Perturbationgradpaar ausgerechnet. Im Allgemeinen sind die beiden Algorithmen effizienter als der Gröbnerwalkalgorithmus (siehe Tabellen 5.6 und 5.7).

Falls die im letzten Schritt aufgetretenen Gröbnerbasen $(G_{t-1})_{\omega_t}$ und M_t große Koeffizienten besitzen, benötigt das Lifting viel Zeit. Die Begründung hierfür liegt in der Berechnung der Quotienten jedes Polynoms in M_t bei Division durch $(G_{t-1})_{\omega_t}$.

(b) Tran (2000) entwickelte einen Algorithmus, in dem der ursprüngliche Zielgewichtsvektor erst dann durch seine Repräsentation ersetzt wird, falls der berechnete dazwischenliegende Gewichtsvektor gleich dem aktuellen Zielgewichtsvektor und die bereits berechnete reduzierte Gröbnerbasis noch nicht die gesuchte Gröbnerbasis ist. Danach wird die Berechnung mit dem neuen Zielgewichtsvektor fortgesetzt. Das Ziel des Algorithmus ist es, zu vermeiden, dass die im letzten Schritt des Gröbnerwalkalgorithmus aufgetretene Gröbnerbasis $(G_{t-1})_{\omega_t}$ zu groß wird. Im Allgemeinen ist der Algorithmus viel effizienter als der Gröbnerwalkalgorithmus (siehe z.B. Tabellen 5.6 und 5.7).

Ist die Anzahl der Variablen groß, so kann es sein, dass die Repräsentation des Zielgewichtsvektors einen oder mehrere große Einträge hat. Dies ist eine der Ursachen davon, dass eine oder mehrere Komponenten eines nächsten berechneten dazwischenliegenden Gewichtsvektors größer als 2^{31} sind. An dieser Stelle erhält man nur die berechnete Gröbnerbasis F des gegebeben Ideals I bzgl. einer von lexikographischen Ordnung verschiedenen. Dann muss die reduzierte lexikographische Gröbnerbasis von $\langle F\rangle$ durch zum Beispiel den Buchbergeralgorithmus ausgerechnet werden. Im Allgemeinen benötigt diese Berechnung viel Zeit und Speicher (siehe z.B. Tabelle 5.14).

(c) Zur Verbesserung des Gröbnerwalkalgorithmus haben wir den Hilbert-driven Algorithmus nur im letzten Schritt durch den Unteralgorithmus 4.4.8 ersetzt. In allen Schritten bis auf den letzten Schritt ist der Hilbert-driven Algorithmus sehr effizient. Die reduzierte Gröbnerbasis $(G_{t-1})_{\omega_t}$, die im letzten Schritt des Gröbnerwalkalgorithmus erreicht wird, ist dann die Eingabe dieses Unteralgorithmus. Diese Variante bezeichnen wir dann als *neuen Gröbnerwalkalgorithmus* (*ngwalk*).

Beim neuen Gröbnerwalkalgorithmus wird die reduzierte Gröbnerbasis M_t von $\langle(G_{t-1})_{\omega_t}\rangle$ bzgl. $\gg_{\omega_t}$ nach kurzer Zeit erreicht (siehe z.B. Tabelle 5.3 für Ideale in $\mathbb{Q}[x_1,\ldots,x_n]$ und Tabelle 5.4 für $\mathbb{Z}_{32003}[x_1,\ldots,x_n]$). Somit ist der Unteralgorithmus 4.4.8 sehr effizient, um die reduzierte Gröbnerbasis des im letzten Schritt des Gröbnerwalkalgorithmus aufgetretenen ω_t-homogenen Ideals auszurechnen.

Bei der Berechnung vieler Beispiele konnten wir folgendes feststellen:

i. Der neue Gröbnerwalkalgorithmus ist viel effizienter als der „originale“ Gröbnerwalkalgorithmus, weil er im Allgemeinen wenig Zeit und Speicher benötigt (siehe Tabellen 5.3 und 5.4).

ii. Für Ideale in $\mathbb{Q}[x_1,\ldots,x_n]$ ist der neue Gröbnerwalkalgorithmus leider noch nicht effizient genug, insbesondere in Fällen, in denen die Polynome von $(G_{t-1})_{\omega_t}$ bzw. M_t große Koeffizienten und viele Monomen besitzen. In diesem Fall benötigt die Anwendung von Satz 3.1.9, nämlich die Prozedur *LiftGB*, viel Zeit und Speicher zur Berechnung

der Quotienten jedes Polynoms von M_t bei Division durch $(G_{t-1})_{\omega_t}$. Im Beispiel **dessin1** bricht es sogar nach 9 Stunden wegen des Speichermangels ab.
In Anhang B bzw. C sind die Größen und die Initialterme von $(G_{t-1})_{\omega_t}$ bzw. M_t mehrerer Ideale abgedruckt, bei denen die Berechnung der Quotienten jedes Polynoms in M_t bei Division durch $(G_{t-1})_{\omega_t}$ sehr aufwendig ist.

iii. Tabelle 5.6 zeigt, dass für Beispiele mit mehreren Variablen der neue Gröbnerwalkagorithmus auch noch schneller ist als der Fractalwalkalgorithmus.

(d) Da die obigen Algorithmen noch nicht effizient genug sind, haben wir neue Alternativalgorithmen entwickelt. In den Alternativalgorithmen haben wir die Ideen des Fractalwalk- und Tranalgorithmus verwendet, damit:

i. die Gröbnerbasen $(G_{t-1})_{\omega_t}$ und M_t nicht zu groß werden und

ii. die Komponenten des berechneten dazwischenliegenden Gewichtsvektors nicht zu groß sind.

Diese Alternativalgorithmen sind viel effizienter als der Gröbnerwalkalgorithmus (siehe z.B. Tabelle 5.6).

4. Betrachten wir die Strecke des neuen Gröbnerwalkalgorithmus, können wir feststellen, dass das letzte Stück $\overline{\omega_{t-1}\tau}$ der Strecke $\overline{\sigma\tau}$ des Gröbnerwalkalgorithmus oder der Zielgewichtsvektor τ im Durchschnitt mehrerer Gröbnerkegel liegt (siehe Tabelle 5.5).

5.2 Analyse des Perturbationwalkalgorithmus

Am Anfang des Perturbationwalkalgorithmus müssen die beiden gegebenen Gewichtsvektoren, nämlich der Start- und Zielgewichtsvektor, mit dem eingegebenen Perturbationgrad gestört werden. Dafür benötigt man eine entsprechende reduzierte Gröbnerbasis, insbesondere ihren maximalen Totalgrad. Sind dieser Totalgrad bzw. der Perturbationgrad groß, kann es sein, dass eine der folgenden Möglichkeiten auftritt:

- Der gestörte Gewichtsvektor bzw. einer der berechneten dazwischenliegenden Gewichtsvektoren sind sehr groß. Das heißt, eine oder mehrere Komponenten dieses Gewichtsvektors sind größer als 2^{31}. In diesem Fall kann man nicht die Berechnung mit diesem großen Gewichtsvektor fortsetzen.

- Der berechnete gestörte Gewichtsvektor liegt nicht im richtigen Gröbnerkegel.

An dieser Stelle erhält man nur die reduzierte Gröbnerbasis des gegebenen Ideals bzgl. einer von der Zielordnung verschiedenen. Mit Hilfe dieser Gröbnerbasis kann man dann die gesuchte Gröbnerbasis ausrechnen. Hier wird zum Beispiel der Buchbergeralgorithmus oder andere Strategien, etwa der absteigende Rekursivalgorithmus (vgl. Algorithmus 4.4.2), angewendet. Zum Vergleich dieses Algorithmus mit den anderen Algorithmen verwendeten wir Algorithmus 4.4.2, da im Allgemeinen er effizienter als der Buchbergeralgorithmus ist.

Beispiele	n	Berechnungzeiten				Speicherverbrauch	
		gwalk		ngwalk		gwalk	ngwalk
		M_t†	G_{lex}	M_t	G_{lex}		
ex2	3	1440.87	1698.66	0.35	248.59	227716	18332
ex3	3	6681.11	6999.66	0.27	305.52	635999	17907
niermann1	3	5829.09	6940.75	0.50	1080.06	774538	52088
tran2	3	8972.48	9090.33	0.69	145.38	814897	14163
qnt2	3	3132.02	4585.21	0.99	1527.27	231355	191829
issac97	4	506.75	524.57	0.23	21.47	48785	4726
wang6	4	>8987.56	—	0.25	1077.06	∞	61153
casmod1	4	>17459.65	—	1.27	3150.14	∞	112411
ubikker	4	>6433.29	—	4.66	628.76	∞	136227
cyc5mod1	5	>2621.67	—	3.14	1683.47	∞	482425
noon5	5	>1354.00	—	11.77	202.28	∞	57105
kats5	6	683.56	683.79	0.04	0.24	70422	12.55
kats6	6	>3364.77	—	1.49	176.02	∞	54243
MK5	6	>4097.89	—	2.39	1502.33	∞	73385
cyclic6	6	>3060.01	—	0.86	1062.84	∞	109192
s7	7	>1500.27	—	0.31	307.85	∞	118105
redeco8	8	>1390.53	—	0.56	350.63	∞	33650
dessin1	8	>1999.06	—	72.43	>32132.76	∞	∞
dessin2	10	>5457.57	—	16.28	16422.00	∞	919011

Tabelle 5.3: Vergleich: Gröbnerwalkalg. mit dem neuen Gröbnerwalkalg. (in der Spalte M_t (bzw. G_{lex}) sind die gebrauchten Zeiten zur Berechnung der reduzierten Gröbnerbasis von $\langle I_{\omega_t}\rangle$ (bzw. I) bzgl. $\succ_{(\omega_t, lex)}$ (bzw. $\succ_{lex}$) abgedruckt; †$> s$ bedeutet, dass nach s Sekunden bricht die Berechnung wegen des Speichermangels ab).

Wir nennen das Perturbationgradpaar (p, p) **größer** als (d, d), falls $p > d$ ist. Betrachten wir den Perturbationwalkalgorithmus, in dem der gestörten Zielgewichtsvektor im richtigen Gröbnerkegel liegt und jede Komponente der dazwischenliegenden Gewichtsvektoren kleiner als 2^{31} ist, so können wir feststellen: Falls das Perturbationgradpaar des Algorithmus größer wird, so benötigt der Algorithmus mehr Schritte (siehe z.B. Tabellen 4.5 und D.2). Im Allgemeinen erfordert der Perturbationwalkalgorithmus vom größeren Perturbationgradpaar wenig Zeit (siehe Tabelle 4.4 und D.1), weil das Erzeugendensystem des letzten ω-homogenen Ideals viel kleiner als das Erzeugendensystem beim Perturbationwalkalgorithmus vom kleineren Perturbationgradpaar ist (siehe Tabelle D.3 und D.4).

Im Allgemeinen ist der Perturbationwalkalgorithmus vom besten Perturbationgradpaar viel schneller als der Buchberger-, Gröbnerwalk- bzw. Fractalwalkalgorithmus (siehe Tabelle 5.6). Hierbei ist die meiste Zeit zur Reduzierung einer Gröbnerbasis gebraucht (siehe Tabelle **??**). Es gibt keine allgemeingültige Methode, um das beste Perturbationgradpaar zu finden.

In unseren Experimenten haben wir einige Beispiele gefunden, in denen der Perturbationwalkalgorithmus mit einem der Perturbationgradpaare keine Ausgabe zurückliefert hat, weil die Berechnungen wegen Speichermangels abbrechen (z.B. *redeco8* in Tabelle 5.6, *virasoro*, *schwarz_9* bzw. *chemkin* in Tabelle 5.7, und Tabellen in

Beispiele	n	Berechnungzeiten				Speicherverbrauch	
		gwalk		ngwalk		gwalk	ngwalk
		M_t	G_{lex}	M_t	G_{lex}		
ex2	3	0.08	0.14	0.04	0.07	1927	1415
ex3	3	0.16	0.22	0.04	0.11	2440	1927
niermann1	3	0.27	0.31	0.08	0.14	3481	1915
tran2	3	0.20	0.25	0.05	0.09	1927	1927
qnt2	3	0.03	0.10	0.01	0.07	1403	1915
issac97	4	0.02	0.03	0.02	0.04	902	902
wang6	4	0.54	0.59	0.03	0.08	3485	1415
casmod1	4	3.62	3.67	0.02	0.05	10172	1403
ubikker	4	17.32	17.37	0.06	0.11	33847	1927
cyc5mod1	5	$>$ 13 M	—	0.27	24.75	∞	11653
noon5	5	66.23	67.63	0.94	2.27	213006	19340
kats5	6	0.07	0.08	0.02	0.02	910	910
kats6	6	$>$ 8 M	—	0.06	0.88	∞	3986
MK5	6	$>$ 2 M	—	0.06	0.14	∞	2440
cyclic6	6	60.49	60.83	0.15	0.46	225430	3469
s7	7	$>$ 2 M	—	0.20	233.45	∞	32173
redeco8	8	9.72	368.77	0.56	350.63	34307	33650
dessin1	8	$>$ 2 M	—	0.18	545.23	∞	30075
dessin2	10	16.65	464.06	0.17	459.85	49194	34175

Tabelle 5.4: Vergleich: Gröbnerwalkalg. mit dem neuen Gröbnerwalkalg. bei Idealen in $\mathbb{Z}_{32003}[x_1, \ldots, x_n]$.

Anhang D).

5.3 Analyse des Fractalwalkalgorithmus

In diesem Abschnitt stellen wir die Ergebnisse unserer Experimente mit dem Fractalwalkalgorithmus vor. In diesem Algorithmus wird rekursiv der Perturbationwalkalgorithmus mit den Perturbationgradpaaren (j, j) mit $2 \leq j \leq n$ aufgerufen. Hiermit wird die reduzierte Gröbnerbasis eines ω-homogenen Ideals berechnet. Das heißt, das Lifting wird sehr oft durchgeführt. Deswegen hängt die Effizienz des Fractalwalkalgorithmus nicht nur von der Anzahl der Variablen, sondern auch vom Grundkörper des Polynomrings ab, in dem das gegebene Ideal definiert ist.

Aus den experimentellen Daten ergeben sich die Folgerungen:

1. Im Allgemeinen ist der Fractalwalkalgorithmus viel schneller als der Buchberger- bzw. Gröbnerwalkalgorithmus (siehe Tabellen 5.6).

2. Bei Idealen mit 3 Variablen ist der Fractalwalkalgorithmus sehr effizient (siehe Tabelle 5.6).

   (a) Der Algorithmus ist viel schneller als der neue Gröbnerwalkalgorithmus.

   (b) Der Algorithmus ist so schnell wie der Perturbationwalkalgorithmus mit dem besten Perturbationgradpaar und die beiden Alternativalgorithmen.

Bsp.	n	Zur Berechnung		
		τ†	$(G_{t-1})_{\omega_t}$‡	G_{lex}
ex2	3	11	31	42
ex3	3	13	33	46
niermann1	3	12	52	64
tran2	3	11	44	55
qnt2	3	15	27	42
issac97	4	3	16	19
wang6	4	5	26	31
casmod1	4	4	14	28
ubikker	4	4	43	47
cyc5mod1	5	2	18, 61	81
noon5	5	7	46, 50	103
kats5	6	2	6, 12	20
kats6	6	2	6, 9, 26	43
MK5	6	2	6, 9, 26	43
cyclic6	6	2	6, 8, 27	43
s7	7	2	9, 9, 17, 24	43
redeco8	8	2	7, 9,, 17, 1, 67	103
dessin1	8	2	5, ,6, 10, 43	66
dessin2	10	4	3, 1, 5, 6, 10, 37	66

Tabelle 5.5: Anzahl der Schritte des neuen Gröbnerwalkalgorithmus zur Berechnung der reduzierten lexikographischen Gröbnerbasis G_{lex} (†Die Anzahl der Schritte wird gebraucht, um den ursprünglichen Zielgewichtsvektor zu erreichen; ‡Die Anzahl der Schritte wird gebraucht, um nur die reduzierte lexikographische Gröbnerbasis von $(G_{t-1})_{\omega_t}$ zu berechnen).

Analoges gilt für den Vergleich mit dem Groebner- bzw. FGLM Algorithmus.

(c) Die meiste Zeit wird zur Reduzierung einer Gröbnerbasis gebraucht (siehe Tabelle F.1).

3. Bei Idealen mit mehr als 3 Variablen ist der Fractalwalkalgorithmus sehr langsam. In diesem Fall ist er auch langsamer als der neue Gröbnerwalkalgorithmus (vgl. Tabelle 5.6). Die Gründe hierfür sind, dass (1) der Zielgewichtsvektor von großen Perturbationgraden gestört werden muss und (2) das Lifting sehr oft durchgeführt wird und die Koeffizienten der auftretenden Gröbnerbasen sehr groß sind.

   Aus dem ersten Grund tritt ein dazwischenliegenden Gewichtsvektor mit großen Komponenten auf. Der Buchbergeralgorithmus wird dann an dieser Stelle verwendet, um die aktuelle Rekursion zu ersetzen. Diese Berechnung ist allerdings noch sehr aufwendig. In ein paar Beispielen bricht sie wegen des Speichermangels ab. Aus dem anderen Grund benötigt der Algorithmus sehr viel Zeit, aber wenig Speicher. Die meiste Zeit davon wird entweder für das Lifting oder für die Reduzierung einer Gröbnerbasis gebraucht (siehe Tabelle F.1). Falls wir uns mit Idealen im Polynomring über $\mathbb{Z}_{32003}$ beschäftigen, lässt sich das Lifting schnell durchführen (siehe Tabelle 5.8).

Bsp.	n	std	groebner	fglm	gwalk	ngwalk	bpwalk	fwalk	twalk	awalk1	awalk2
ex2	3	$>_z$711 M	0.17	0.62	1698.66	248.59	0.58 (2)	0.82	0.84	0.60	0.56
ex3	3	$>_z$250 M	0.36	1.01	6999.66	305.52	0.49 (2)	0.68	0.50	0.47	0.46
niermann1	3	>178 M	0.32	0.73	6940.75	1080.06	0.63 (2)	0.94	0.66	0.64	0.64
tran2	3	>71353.88	0.34	1.58	9090.33	145.38	0.86 (2)	0.92	1.24	0.89	0.88
qnt2	3	>18192.93	17.46	—	4585.21	1527.27	10.96 (3)	12.44	10.46	10.68	10.46
issac97	4	0.30	0.18	0.37	524.57	21.47	0.30 (4)	190.82	0.30	0.21	0.21
wang6	4	>97 M	0.35	0.76	>8987.56	1077.06	0.35 (4)	106.67	0.55	0.40	0.38
casmod1	4	$>_z$119 M	56.57	8.03	>17459.65	3150.14	4.38 (3)	1956.26	5.68	3.55	3.82
ubikker	4	$>_z$249 M	3.93	9.62	>6433.29	628.76	7.29 (3)	$>_z$409 M	7.03	4.84	5.05
cyc5mod1	5	>215 M	1.89	1.43	>2621.67	1683.47	7.83 (5)	>3430.92	5.17	2.31	2.44
noon5	5	>198 M	923.50	46.92	>1354.00	202.28	31.79 (3)	80.36	28.78	24.77	15.84
kats5	6	0.19	0.03	0.04	683.79	0.24	0.08 (4)	0.49	0.04	0.05	0.05
kats6	6	>155 M	1.69	0.92	>3364.77	176.02	1.12 (5)	1693.62	1.16	1.53	1.76
MK5	6	>108 M	1.09	3.03	>4097.89	1502.33	2.75 (5)	4702.99	1.86	1.81	2.75
cyclic6	6	$>_z$157 M	1.35	0.74	>3060.01	1062.84	2.64 (5)	4106.64	1.84	1.70	2.64
s7	7	161.92	0.13	0.207	>1500.27	307.85	0.70 (7)	496.20	0.53	0.55	0.48
redeco8	8	9759.28	7.87	7.33	>1390.53	350.63	(?)†	>5375.56	22495.33	8.91	12.72
dessin1	8	$>_z$164 M	5761.20	67.18	>1999.06	72.43	86.27 (6)	>13385.90	3029.86	437.29	101.01
dessin2	10	>45 M	824.26	37.80	>5457.57	16422.00	721.52 (8)	>5519.28	8363.92	66.59	24.79

Tabelle 5.6: Berechnungzeiten: Gröbnerwalkalg. und mehrere Algorithmen (†Der Perturbationwalkalgorithmus mit einem der Perturbationgradpaare liefert keine Ausgabe zurück, weil die Berechnungen wegen Speichermangels abbrechen).

Bsp.	n	std	groebner	fglm	gwalk	ngwalk	bpwalk	fwalk	twalk	awalk1	awalk2	tdp
qnt1	3	83.26	0.41	—	74.32	3.16	1.45 (2)	0.14	0.93	0.98	1.58	0.05
qnt2	3	0.03	0.03	—	0.10	0.07	0.07 (2)	0.14	0.09	0.10	0.10	0.01
lichtblau	3	52.02	0.11	—	0.23	0.13	0.31 (2)	0.29	0.28	0.28	0.28	0.01
ubikker	4	0.11	0.02	0.01	17.37	0.11	0.08 (3)	0.09	0.06	0.07	0.06	0.00
casmod1	4	0.33	0.11	0.03	3.67	0.05	0.06 (4)	0.05	0.06	0.06	0.07	0.02
cylic5mod1	5	17.43	0.09	0.04	>13 M	32.65	0.25 (4)	9.79	0.21	0.23	0.22	0.02
noon5	5	3.31	1.18	0.61	67.63	2.27	2.19 (2)	2.52	3.65	3.89	1.95	0.03
noon6	6	1396.14	104.65	14.86	>3 M	681.24	93.73 (2)	68.62	95.23	106.03	19.24	0.14
cylic6	6	0.23	0.11	0.06	60.83	0.46	1.05 (5)	0.61	0.36	0.33	0.23	0.03
exam54	6	0.33	0.11	0.07	48.60	0.46	0.64 (4)	0.42	0.36	0.38	0.23	0.04
katsura7	7	>7 M	0.16	0.08	>5 M	$>_z$178 M	0.68 (6)	$>_z$26 M	0.49	0.50	0.45	0.04
schwarz7	7	0.19	0.15	0.03	>2 M	233.45	0.58 (6)	338.74	0.36	0.36	0.23	0.01
schwarz8	8	12.66	0.38	0.12	>2 M	18.22	6.07 (6)	761.13	2.68	2.61	1.19	0.04
vsoro	8	205.15	8.03	1.82	>2 M	6.30	(?)†	>9 M	6.04	5.55	3.85	1.13
redeco8	8	0.99	0.16	0.06	368.77	350.63	1.25 (5)	581.90	0.48	0.51	0.40	0.03
exam57	8	>7 M	0.45	0.04	>3 M	1078.53	0.31 (7)	119.42	0.22	0.22	0.18	0.02
dessin1	8	>9 M	0.71	0.03	>2 M	545.23	0.26 (7)	41.20	0.39	0.37	0.37	0.03
filter9	9	2.00	25.88	0.48	0.62	0.67	1.91 87)	2.72	0.93	0.95	0.66	0.45
schwarz9	9	>8 M	28.63	4.34	>1 M	>2 M	(?)†	$>_z$45 M	>5 M	445.72	89.93	0.84
el44	9	0.92	$>_z$20 M	0.03	0.04	0.06	0.04 (8)	0.05	0.05	0.04	0.05	0.03
dessin2	10	0.04	2.51	0.07	464.06	459.85	0.29 (9)	4048.69	0.24	0.23	0.25	0.03
f744	12	67.83	5.97	—	>2 M	6.85	0.64 (9)	>5 M	0.77	0.77	1.06	0.40
bmarkD1	12	85.46	3.05	0.08	>2 M	$>_z$26 M	0.97 (8)	$>_z$62 M	0.42	0.43	0.36	0.07
chemkin	13	0.55	0.55	0.05	>2 M	656.48	(?)†	$>_z$53 M	0.36	0.32	0.32	0.04

Tabelle 5.7: Berechnungszeiten der Algorithmen bei Idealen aus $\mathbb{Z}_{32003}[x_1, \ldots, x_n]$ (†Der Perturbationwalkalgorithmus mit einem der Perturbationgradpaare liefert keine Ausgabe zurück, weil die Berechnungen wegen Speichermangels abbrechen).

Bsp.	n	$\mathbb{Q}$	$\mathbb{Z}_{32003}$
qnt2	3	12.44	0.14
lichtblau	3	726.58	0.29
ubikker	4	$>_z$409 M	0.09
issac97	4	190.82	0.03
wang6	4	106.67	0.06
exam59	4	209.24	0.03
casmod1	4	1956.26	0.05
cyclic5mod1	5	>3430.92	9.79
noon5	5	80.36	2.52
noon6	6	$>_z$794 M	68.62
cylic6	6	4106.64	0.61
exam54	6	3733.23	0.42
kats6	6	1693.62	0.79
MK5	6	4702.99	0.31
schwarz7	7	496.20	338.74
redeco8	8	>5379.50	581.90
dessin1	8	>13385.90	41.20
dessin2	10	>5519.28	4048.69

Tabelle 5.8: Berechnungzeiten des Fractalwalkalgorithmus bei Idealen in $K[x_1, \ldots, x_n]$ mit $K := \mathbb{Q}$ bzw. $K := \mathbb{Z}_{32003}$.

5.4 Analyse des Tranalgorithmus

Beim Tranalgorithmus muss der aktuelle Zielgewichtsvektor durch die Repräsentation des ursprünglichen Zielgewichtsvektor ersetzt werden, wenn der berechnete dazwischenliegende Gewichtsvektor gleich dem aktuellen Zielgewichtsvektor und die berechnete Gröbnerbasis noch nicht die gesuchte Gröbnerbasis ist. Diese Repräsentation hängt nicht nur von der vorhandenen reduzierten Gröbnerbasis, sondern auch von der Anzahl der Variablen ab. Falls diese Repräsentation große Komponenten hat, kann auch der nächste berechnete dazwischenliegende Gewichtsvektor große Komponenten besitzen. Das heißt, eine oder mehrere Komponenten dieses Gewichtsvektors sind größer als 2^{31}. Also kann man nicht den Algorithmus mit diesem Gewichtsvektor fortsetzen. Hierbei erhält man nur die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. einer von der gegebenen Zielordnung verschiedenen monomialen Ordnung. Gemäß unseren Erfahrungen ist diese Gröbnerbasis im Allgemeinen groß (siehe Anhang E) und die weitere Berechnung sehr aufwendig. In diesem Fall wird der Tranalgorithmus nicht effizient sein.

Damit die gewünschte Gröbnerbasis erreicht wird, kann man den Buchbergeralgorithmus oder andere Strategien auf das Ideal $\langle F \rangle$ anwenden. Tabelle 5.14 zeigt, dass der Tranalgorithmus, in dem der Buchbergeralgorithmus auf das Ideal $\langle F \rangle$ angewandt wird, langsamer ist als der zweite Alternativalgorithmus.

Tabellen 5.6 und 5.7 zeigen, dass der Tranalgorithmus im Allgemeinen viel schneller ist als der Buchberger-, Gröbnerwalk-, neue Gröbnerwalk- bzw. Fractalwaklkalgorithmus. Für Ideale mit wenig Variablen ist der Tranalgorithmus genauso schnell wie der Groebner- bzw. FGLM-Algorithmus, wie der Perturbationwalkalgorithmus vom besten Perturbationgradpaar bzw. wie die beiden Alternativalgorithmen.

5.5 Analyse der Alternativalgorithmen

Aus den von uns verwendeten Beispielen zum Vergleich zwischen den beiden Alternativalgorithmen können wir folgendes feststellen:

1. Für Ideale mit 3 Variablen gehen die beiden Alternativalgorithmen auf der gleichen Strecke, weil die erste aufgerufene Rekursion des Perturbationwalkalgorithmus vom Grad 2 die gesuchte Gröbnerbasis zurückliefern kann.

2. Im Allgemeinen sind die beiden Alternativalgorithmen gleichschnell. Ausnahme bilden die in Tabelle 5.9 aufgelisteten Beispiele. In den Beispielen **chemkin** bzw. **noon6** benötigt der erste Alternativalgorithmus sehr viel Zeit. Nach 1589 Minuten (für **noon6**) bzw. 1956 Minuten (für **chemkin**) hat er keine Ausgabe zurückliefert, weil der aktuelle Zielgewichtsvektor noch nicht erreicht ist.

Beispiele	n	Berechnungzeiten		Speicherverbrauch	
		awalk1	awalk2	awalk1	awalk2
cyc5mod1	5	2.31	2.44	4494	3982
MK5	6	1.81	2.75	2028	2028
kats7	7	68.03	96.19	9511	9735
exam57	8	558.04	603.99	11637	10664
redeco8	8	8.91	12.72	8586	6156
el44	9	4.93	5.82	1956	1952
filter9	9	87.76	113.26	5035	5055
benchmarkD1	12	423.39	642.64	12358	14582
lichtblau	3	694.78	684.84	27897	27845
qnt1	3	1626.87	1618.69	41182	41082
noon5	5	24.77	15.84	33959	17527
noon6	6	$>_z$1589 M	724.63	$\cdots$	223832
eco7	7	15.13	0.54	3525	1787
virasoro	8	181.23	178.58	41346	26003
dessin1	8	437.29	101.01	20254	5255
hairer	13	5.90	5.25	3849	2833
chemkin	13	$>_z$1956 M	456.83	$\cdots$	24506

Tabelle 5.9: Vergleich zwischen dem ersten und zweiten Alternativalgorithmus.

3. Die meiste Zeit ist zur Reduzierung einer Gröbnerbasis nötig, falls im absteigenden bzw. aufsteigenden Rekursivalgorithmus (vgl. Algorithmus 4.4.2 bzw. 4.4.8) die Prozedur *RedGB_BA* nicht durchgeführt wird. Ansonsten wird die meiste Zeit zur Berechnung der reduzierten lexikographischen Gröbnerbasis eines Ideals benötigt. Diese Gröbnerbasis wird durch den Buchbergeralgorithmus ausgerechnet (siehe Tabelle F.3 für den ersten Alternativalgorithmus und Tabelle F.4 für den zweiten).

Durch die Anwendung der Startordung $\succ_{rlex}$ bzw. $\succ_{glex}$ zur Berechnung der ersten Gröbnerbasis für homogene Ideale sind die vom Perturbationwalkalgorithmus verschiedenen Varienten des Gröbernerwalkalgorithmus nicht effizient. Hierbei sind sie langsamer als der Buchberger- bzw. Hilbert-driven Buchbergeralgorithmus. Im ersten Abschnitt dieses Unterkapitels stellen wir unseren Versuch vor, den zweiten

Alternativalgorithmus schneller als den Buchbegeralgorithmus zu machen. In den restlichen Abschnitten geben wir dann den Vergleich der beiden Alternativalgorithmen mit den anderen Algorithmen zur Berechnung der reduzierten lexikographischen Gröbnerbsasis mehrerer Ideale an. Hier betrachten wir den Zeit- bzw. Speicherverbrauch jedes Algorithmus.

5.5.1 Homogene Ideale

Um die reduzierte Gröbnerbasis des Ideals $I \subset R := K[x_1, \ldots, x_n]$ bzgl. der lexikographischen Ordnung $\succ_{lex}$ auf R durch eine Variante des Gröbnerwalkalgorithmus auszurechnen, gibt man nur ein Erzeugendensystem von I ein. Im Allgemeinen stellt dieses System keine reduzierte Gröbnerbasis von I dar. Ist dieses Erzeugendensystem keine reduzierte Gröbnerbasis, so verwendet man gewöhnlich eine der graduierten monomialen Ordnungen auf R, etwa $\succ_{rlex}$ oder $\succ_{glex}$ zur Berechnung der ersten reduzierten Gröbnerbasis. Falls man diese Vorgehensweise auf homogene Ideale anwendet, benötigt jede vom Perturbationwalkalgorithmus verschiedene Variante des Gröbnerwalkalgorithmus nur einen Schritt, da der erste dazwischenliegende Gewichtsvektor undefiniert ist. Das heißt, es gibt nur eine Basiskonvertierung, nämlich von $\succ_{rlex}$ nach $\succ_{(\sigma,\, lex)}$ mit $\sigma = (1, 1, \ldots, 1)$, und der Zielvektor $\tau = (1, 0, \ldots, 0)$ liegt im Gröbnerkegel von I bzgl σ.

In diesem Unterabschnitt präsentieren wir nun unsere Experimente mit dem zweiten Alternativalgorithmus für homogene Ideale. In diesem Fall verwenden wir drei verschiedene ω_i-gewichtete lexikographische Ordnungen $\gg_{\omega_i}$ als Startordnungen zur Berechnung der ersten reduzierten Gröbnerbasis des gegebenen Ideals mit $\omega_0 := (1, 1, \ldots, 1), \omega_1 := (n-1, 1, \ldots, 1, 0)$ und $\omega_2 := (3, 2, 1, \ldots, 1, 0)$.

Aus Tabelle 5.10 wird ersichtlich, dass für homogene Ideale der Buchberger- bzw. Hilbert-driven Buchbergeralgorithmus schneller ist als die Varianten des Gröbnerwalkalgorithmus mit der Startordnung $\succ_{rlex}$. Mit jeder der drei obigen Startordnungen ist allerdings der zweite Alternativalgorithmus effizienter als mit der Startordnung $\succ_{rlex}$ bzw. der Buchbergeralgorithmus.

Bsp.	std	hilb	ngwalk	Zweiter Alternativalg. mit			
				$\succ_{rlex}$	$\omega_0^{(\mathrm{lp})}$	$\omega_1^{(\mathrm{lp})}$	$\omega_2^{(\mathrm{lp})}$
liu	2.39	0.04	2.83	1.84	1.55	0.97	0.99
sym47	4.33	5.63	11.72	7.11	1.49	10.45	4.21
Fate	10.00	0.09	14.07	15.12	5.96	3.35	3.21
sym34	12.29	0.14	21.84	17.29	18.27	4.86	4.90
hf744	64.00	41.75	64.49	59.45	9.26	25.81	25.64
gerhard1	53.13	1.20	109.89	110.22	45.34	23.01	15.72
weispf94	52.33	0.25	58.78	65.54	43.37	11.16	10.52

Tabelle 5.10: Berechnungzeiten: Zweiter Alternativalgorithmus und andere Algorithmen bei homogenen Idealen.

5.5.2 Vergleich mit std, stdfglm und groebner

In diesem Unterabschnitt präsentieren wir den Vergleich zwischen den beiden Alternativalgorithmen und den in SINGULAR bereits implementierten Algorithmen,

nämlich der Buchberger-, FGLM- und Groebneralgorithmus.

Um aussagen zu können, dass die beiden Alternativalgorithmen viel effizienter als der Buchberger-, FGLM- und Groebneralgorithmus sind, geben wir mehrere Tabellen an, in denen der Zeit- und maximale Speicherverbrauch der verglichen Algorithmen abgedruckt ist.

Zusammenfassend können wir folgendes feststellen:

1. Bei nichthomogenen Idealen sind die beiden Alternativalgorithmen viel effizienter als der Buchbergeralgorithmus (siehe z.B. Tabellen 5.6 und 5.7).

2. Bei den Berechnungen, bei denen wenig Zeit benötigt wird, ist der zweite Alternativalgorithmus in der Regel etwas langsamer als der andere Algorithmus (siehe Tabelle 5.11 zum Vergleich mit dem FGLM-Algorithmus und Tabelle 5.12 mit dem Groebneralgorithmus, obere Hälfte).

3. Bei den Berechnungen, bei denen viel Zeit benötigt wird, ist der zweite Alternativalgorithmus schneller als der andere Algorithmus (siehe Tabelle 5.11 zum Vergleich mit dem FGLM-Algorithmus und Tabelle 5.12 mit dem Groebneralgorithmus, untere Hälfte).

5.5.3 Vergleich mit den Varianten des Gröbnerwalkalgorithmus

In diesem Unterabschnitt vergleichen wir die Alternativalgorithmen mit den auf dem Gröbnerwalkalgorithmus basierenden Algorithmen. Es zeigt sich, dass im Allgemeinen die beiden Alternativalgorithmen effizienter als die anderen Algorithmen sind:

1. Beim Gröbnerwalkalgorithmus benötigt der Hilbert-driven Algorithmus zur Berechnung der reduzierten Gröbnerbasis M des im letzten Schritt des Algorithmus aufgetretenen ω-homogenen Ideals $\langle G_\omega \rangle$ viel Zeit und Speicher. Wird die Berechnung der Gröbnerbasis M abgeschlossen, so braucht das Lifting viel Zeit und Speicher (siehe z.B. Tabellen 5.6 und 5.3), falls G_ω bzw. M groß ist (vgl. der neue Gröbnerwalkalgorithmus).

   Die beiden Alternativalgorithmen können diese Schwierigkeit vermeiden und sind immer viel effizienter als der Gröbnerwalkalgorithmus bzw. der neue Gröbnerwalkalgorithmus (siehe Tabellen 5.6, 5.7 und 5.13).

2. Bei Beispielen mit mehreren Variablen sind die beiden Alternativalgorithmen viel effizienter als der Perturbationwalkalgorithmus vom besten Perturbationgradpaar (siehe z.B. Tabelle 5.6).

3. Zum Vergleich mit dem Fractalwalkalgorithmus siehe Tabellen 5.6. Bei Idealen mit 3 Variablen sind alle drei Algorithmen gleichschnell. Ansonsten sind die beiden Alternativalgorithmen im Allgemeinen viel effizienter als der Fractalwalkalgorithmus.

4. Zum Vergleich mit dem Tranalgorithmus siehe Tabellen 5.6 und 5.14. Bei Beispielen mit mehreren Variablen ist der zweite Alternativalgorithmus schneller als der Tranalgorithmus.

Beispiele	n	Berechnungzeiten			Speicherverbrauch		
		fglm	awalk1	awalk2	fglm	awalk1	awalk2
exam7	3	1.42	1.75	1.71	1407	1896	1896
cyc5mod1	5	1.43	2.31	2.44	1439	4494	3982
kats6	6	0.92	1.53	1.76	1371	1920	1920
cyclic6	6	0.74	1.70	2.64	1779	3333	2308
exam54	6	1.16	1.79	2.01	1775	3325	2300
redeco7	7	0.17	0.36	0.36	742	1787	1787
kats7	7	53.21	68.03	96.19	5343	9511	9735
s7	7	0.20	0.55	0.48	1267	3341	1795
schwarz8	8	1.45	3.93	1.95	2800	14664	4591
redeco8	8	7.33	8.91	12.72	2484	8586	6156
dessin1	8	67.18	437.29	101.01	5923	20254	5255
exam57	8	535.34	558.04	603.99	13150	11637	10664
filter9	9	71.54	87.76	113.26	5771	5035	5055
exam60	3	1.20	0.88	0.96	1315	1787	1787
rose	3	2.44	0.96	0.95	1767	1787	1787
rose2	3	8.20	1.46	2.05	2372	2404	2404
m41	3	19.64	6.43	6.91	3217	3713	3713
m52470	3	661.86	323.99	371.21	20277	17218	17230
m51330	3	1807.87	732.30	612.63	43468	16934	16926
m51390	3	92.37	68.45	69.33	8041	10224	10228
m63580	3	3557.36	1764.80	1784.92	51139	38427	38515
ubikker	4	9.62	4.84	5.05	1799	2036	2108
noon5	5	46.92	24.77	15.84	5883	33959	17527
casmod1	5	8.03	3.55	3.82	1975	1591	1591
MK5	6	3.03	1.81	2.75	1443	2028	2028
noon6	6	4862.99	$>_z$1589 M	724.63	77097	$\cdots$	223832
vsoro	8	195.11	181.23	178.58	16462	41346	26003
dessin2	10	37.80	66.59	24.79	4758	8946	4358
bmD1	12	738.66	423.39	642.64	16874	12358	14582
chemkin	13	516.68	$>_z$1956 M	456.83	14568	$\cdots$	24506

Tabelle 5.11: Alternativalgorithmen und FGLM-Algorithmus.

Bsp.	n	Berechnungzeiten			Speicherverbrauch		
		groebner	awalk1	awalk2	groebner	awalk1	awalk2
bsp6	3	0.17	0.67	0.63	1775	3065	3065
bsp7	3	0.91	1.59	1.77	2796	3737	3737
ex2	3	0.17	0.60	0.56	1339	1315	1315
ex3	3	0.36	0.47	0.46	1351	1836	1836
tran2	3	0.34	0.89	0.88	2292	1791	1791
exam7	3	0.80	1.75	1.71	2288	1896	1896
niermann1	3	0.32	0.64	0.64	2300	1787	1787
m41	3	5.73	6.43	6.91	5895	3713	3713
lichtblau	3	516.97	694.78	684.84	117308	27897	27845
m51330	3	315.27	732.30	612.63	165483	16934	16926
ubikker	4	3.93	4.84	5.05	4029	2036	2108
cyc5mod1	5	1.89	2.31	2.44	3841	4494	3982
MK5	6	1.09	1.81	2.75	2296	2028	2028
exam54	6	1.37	1.67	1.73	1795	3329	2304
cyclic6	6	1.35	1.70	1.70	1799	3333	2308
redeco7	7	0.13	0.36	0.36	1267	1787	1787
s7	7	0.13	0.55	0.48	806	3341	1795
schwarz8	8	0.55	3.93	1.95	1891	14664	4591
redeco8	8	7.87	8.91	12.72	9935	8586	6156
rose2	3	27.51	1.46	2.05	47556	2404	2404
m51390	3	97.91	68.45	69.33	37875	10224	10228
m52470	3	891.75	323.99	371.21	89213	17218	17230
m52430	3	643.77	376.62	451.23	114633	18764	18804
m63580	3	6700.04	1764.80	1784.92	239071	38427	38515
qnt1	3	2002.27	1626.87	1618.69	354136	41182	41082
m62480	3	8709.57	4284.11	3558.73	629873	59175	58975
casmod1	4	56.57	3.55	3.82	5159	1591	1591
vermeer76	5	0.52	0.05	0.09	1783	762	762
hieta1	5	234.09	1.65	0.99	14448	1519	1403
noon5	5	923.50	24.77	15.84	167297	33959	17527
noon6	6	>8 St	$>_z$1589 M	724.63	∞	$\cdots$	223832
kats7	7	222.31	68.03	96.19	42947	9511	9735
vsoro	8	1285.39	181.23	178.58	34888	41346	26003
dessin1	8	5761.20	437.29	101.01	85153	20254	5255
exam57	8	8972.64	558.04	603.99	77465	11637	10664
filter9	9	4238.57	87.76	113.26	270974	5035	5055
omdi	9	199.07	0.21	0.21	361.81	762	762
el44	9	$>_z$337 M	4.93	5.82	512 MB	1956	1952
dessin2	10	824.26	66.59	24.79	61261	8946	4358
f744	12	32.20	6.77	10.62	9543	5067	8486
bmarkD1	12	16829.77	423.39	642.64	267403	12358	14582
h2bgk	13	351.06	5.90	5.25	74150	3849	2833
chemkin	13	2608.76	$>_z$1956 M	456.83	38355	$\cdots$	24506

Tabelle 5.12: Alternativalgorithmen und Groebneralgorithmus.

Beispiele	n	Berechnungzeiten			Speicherverbrauch		
		ngwalk	awalk1	awalk2	ngwalk	awalk1	awalk2
ex2	3	248.59	0.60	0.56	18332	1315	1315
ex3	3	305.52	0.47	0.46	17907	1836	1836
niermann1	3	1080.06	0.64	0.64	52088	1787	1787
tran2	3	145.38	0.89	0.88	14163	1791	1791
qnt2	3	1527.27	10.68	10.46	191829	6542	5499
issac97	4	21.47	0.21	0.21	4726	762	762
wang6	4	1077.06	0.40	0.38	61153	1275	1275
casmod1	4	3150.14	3.55	3.82	112411	1591	1591
ubikker	4	628.76	4.84	5.05	136227	2036	2108
cyc5mod1	5	1683.47	2.31	2.44	482425	4494	3982
noon5	5	202.28	24.77	15.84	57105	33959	17527
kats5	6	0.24	0.05	0.05	12.55	770	770
kats6	6	176.02	1.53	1.76	54243	1920	1920
MK5	6	1502.33	1.81	2.75	73385	2028	2028
cyclic6	6	1062.84	1.70	2.64	109192	3333	2308
s7	7	307.85	0.55	0.48	118105	3341	1795
redeco8	8	350.63	8.91	12.72	33650	8586	6156
dessin1	8	>32132.76	437.29	101.01	∞	20254	5255
dessin2	10	16422.00	66.59	24.79	919011	8946	4358

Tabelle 5.13: Alternativalgorithmen sind viel schneller als der neuen Gröbnerwalkalgorithmus.

Beispiele	n	Berechnungzeiten			Speicherverbrauch		
		twalk	awalk1	awalk2	twalk	awalk1	aalk2
m52470	3	316.03	323.99	371.21	17743	17218	17230
m51330	3	554.48	732.30	612.63	16926	16934	16926
lichtblau	3	621.85	694.78	684.84	67207	27897	27845
qnt1	3	1884.58	1626.87	1618.69	41094	41182	41082
cyc5mod1	5	5.15	2.31	2.44	5523	4494	3982
noon5	5	28.78	24.77	15.84	32421	33959	17527
noon6	6	4472.06	$>_z$1589 M	724.63	1006805	$\cdots$	223832
s7	7	0.51	0.55	0.48	2792	3341	1795
kats7	7	$>_z$688 M	68.03	96.19	$\cdots$	9511	9735
schwarz8	8	6.01	3.93	1.95	15201	14664	4591
redeco8	8	22495.33	8.91	12.72	86146	8586	6156
virasoro	8	133.71	181.23	178.58	45414	41346	26003
exam57	8	817.83	558.04	603.99	36081	11637	10664
filter9	9	96.43	87.76	113.26	5079	5035	5055
dessin2	10	8363.92	66.59	24.79	2068.92	8946	4358
bmarkD1	12	42819.13	423.39	642.64	294485	12358	14582
chemkin	13	$>_z$780 M	$>_z$1956 M	456.83	$\cdots$	$\cdots$	24506

Tabelle 5.14: Alternativalgorithmen und Tranalgorithmus.

Anhang A

Beispiele

In diesem Anhang sind mehrere von den von unseren Experimenten verwendeten Beispielen abgedruckt. Die von uns mit Hilfe SINGULAR konstruierten Beispiele bezeichnen wir mit **mx**, wobei **x** eine Zahl ist, z.B. m51330. Die anderen Beispiele befinden sich in verschiedenen Arbeiten und auch im Internet, zum Beispiel:

1. bsp1,..., bsp7 in Amrhein et al. (1996a),
2. ex1,..., ex9, s6 und s7 in Amrhein et al. (1996b),
3. cyc5, cyc6,cyc7, caprasse, Aux und ModF in Faugère et al. (1993),
4. tran1, tran2, canny1, canny2 und trinks1 in Tran (1998),
5. qnt1, qnt2 und qnt3 in Tran (2000),
6. dessin1, dessin2, cyclic5mod1, cassoumod1 und symmetric47 in Wichmann (1997),
7. ubikker, issac97, lichtblau, el44, vermeer76, hairer, liu, wang6, rose, rose2, f633, f744, hf633, hf744, filter9, virasoro, benchmarkD1, Fate und MK5 in http://www.calfor.lip6.fr/~jcf/benchs,
8. chemkin, eco**x**, noon**x**, redeco**x**, weispf94, wang**x**, vermeer**x**, sym**x**, exam**x**, gerhard1 und weispfenning94 in http://www.symbolicdata.org/SD_HTML, wobei **x** eine Zahl ist.

A.1 Keine homogenen Ideale

1. **ex2** mit $z > y > x$:
   Ideal $I = \langle x + 3xy^3 + y4 + yz^2, -x^2z + 2y^3z + z^2 + 2yz^2 + 3xyz^2, 3x^3 + xy^2 + yz^2 - 2xz^3 \rangle$.
2. **ex3** mit $z > y > x$:
   Ideal $I = \langle x^2 + y^4 + x^3z + yz - 2xz^3, x^2y^2 + y^3z + z^3 + 3yz^3, y^4 - x^2z + 2y^2z - 2xyz^2 \rangle$.
3. **bsp7** mit $z > y > x$:
   Ideal $I = \langle 2x^4 + 2y^2 + 3x^2yz + xz^2 + 3z^4, 1 + x^3 + 3xyz + y^2z + xyz^2 \rangle$.

4. **qnt1** (Beispiel 3.1 in Tran, 2000) mit $x > y > z$:
   Ideal $I = \langle 5x^4+13y^2z+11x^4yz^3+12x^2z^4+2x^4z^4+5yz^4+13x^3yz^4, 11xy+15y^3+4x^2y^4z+2xz^2+18x^2z^2+19x^2yz^3, 3xy+16xz^2+20x^3yz^2+3yz^3+4xy^2z^3+2x^4y^2z^3\rangle$.

5. **qnt2** (Beispiel 3.3 in Tran, 2000) mit $x > y > z$:
   Ideal $I = \langle 16+3x^3+16x^2z+14x^2y^3, 6+y^3z+17x^2z^2+7xy^2z^2+13x^3z^2\rangle$.

6. **tran2** mit $x > y > z$:
   Ideal $I = \langle 2x^2y+x^3y+2xy^2z, xy^3+y^4+yz^2-z^3-2xz^3, 2-3x^2y+2x^3y+yz^3\rangle$.

7. **lichtblau** mit $u > x > y$:
   Ideal $I = \langle x-110u^2+495u^3-1320u^4+2772u^5-5082u^6+7590u^7-8085u^8+5555u^9-2189u^{10}+374u^{11}, y-22u+110u^2-330u^3+1848u^5-3696u^6+3300u^7-1650u^8+550u^9-88u^{10}-22u^{11}\rangle$.

8. **rose** mit $z > y > x$:
   Ideal $I = \langle 7y^4-20x^2, 2160z^4x^2+1512z^4x+315z^4-4000x^2-2800x-490, 67200000y^3x^5+94080000y^3x^4+40924800y^3x^3+2634240y^3x^2-2300844y^3x-432180y^3+40320000zy^2x^6+28800000zy^2x^5+21168000zy^2x^3+4939200zy^2x^2+347508zy^2x-23520000z^2yx^4-41395200z^2yx^3-26726560z^2yx^2-7727104z^2yx-852355z^2y-10080000z^3x^4-28224000z^3x^3-15288000z^3x^2-1978032z^3x-180075z^3\rangle$.

9. **exam7** mit $x > y > z$:
   Ideal $I = \langle 2x^2+3y^2+7xz+9yz+5z^2+4x, 3x^4+6y^3+xyz+3xz^2+2yz^2+4z^2+5, 3y^4z+7x^3+10z^3+8xy+12y^2+18xz+12\rangle$.

10. **niermann1** mit $x > y > z$: Ideal $I = \langle x^2+xy^2z-2xy+y^4+y^2+z^2, -x^3y^2+xy^2z+xyz^3-2xy+y^4, -2x^2y+xy^4+yz^4-3\langle$.

11. **m41** mit $a > b > c$:
    Ideal $I = \langle 2a^4+a^2c+2ab^2+2abc+b^3c+2b^2c^2+bc^3, 2a^4+2a^3c+2a^2+ab^3+2b^3+2bc^2+c^4+2c^3+c^2, a^3b+a^3c+2abc^2+abc+2ac^2+2ac+2b^3c+2b^2c+b+2c^2\rangle$.

12. **m43** mit $a > b > c$:
    Ideal $I = \langle a^4+2a^2b^2+2a^2bc+a^2b+a^2c^2+ab^2+abc^2+2abc+ac^3+2ac^2+2b^2c^2+2bc^2+c^4, a^3b+a^3c+a^2b+2a^2c^2+ac^3+ac^2+b^3c+2b^3+2b^2c+2b^2+bc^3+bc+2c^2, 2a^4+a^3b+2a^3+2a^2b+2ab^3+2abc+ac+b^3c+2bc^3+2c^2+c\rangle$.

13. **m44** mit $a > b > c$:
    Ideal $I = \langle 2a^4+a^3c+2a^2bc+2a^2b+2a^2c^2+2a^2c+ab^3+2abc^2+2abc+2ac^2+b^2c^2+b^2c+2bc^3+2bc^2+2c^4, a^3b+2a^3c+a^3+2a^2b^2+2a^2bc+2a^2c^2+2a^2c+a^2+2abc^2+ac^2+2ac+2b^2c^2+2bc^2+c^4+c^3+2c^2, 2a^3c+a^3+2a^2b^2+2a^2c^2+2ab^2+abc+2b^4+2b^3+2b^2c^2+2b^2+2bc+2b+2c^4+c^2+2c\rangle$.

14. **m48** mit $a > b > c$:
    Ideal $I = \langle 2a^4+a^3b+a^3c+a^2b^2+2a^2bc+2a^2b+a^2c^2+2a^2c+ab^3+ab^2c+ab^2+2abc^2+2abc+2ac^3+2ac^2+2b^4+2b^3c+b^3+b^2c+bc^3+bc^2+c^4+c^3, 2a^4+a^3b+a^3c+a^3+2a^2b^2+a^2bc+a^2b+a^2c^2+2a^2c+2a^2+ab^3+2ab^2c+2ab^2+abc^2+abc+2ab+2ac^2+ac+b^4+b^3c+b^3+2b^2c^2+2b^2c+2b^2+2bc^3+bc^2+c^4+2c^2, 2a^4+a^3b+2a^3+a^2b^2+2a^2bc+a^2b+a^2c^2+2a^2c+a^2+2ab^3+ab^2c+2ab^2+2abc^2+2ab+ac^3+ac^2+ac+b^4+2b^3c+2b^3+2b^2c^2+b^2c+b^2+2bc^3+bc^2+bc+2b+c^4+c^3+c\rangle$.

15. **m51390** mit $a > b > c$:
    Ideal $I = \langle 2a^5+a^3c+2a^3+2a^2bc+b^4c+b^3c^2, a^3b^2+a^3c+2ab^4+b^5+b^2+bc^2+c^4, 2a^4b+2a^3+a^2bc^2+2a^2bc+2ab+2a+b^3c^2+b^3c\rangle$.

16. **m51330** mit $a > b > c$:
Ideal $I = \langle a^5+2a^4+2a^3b^2+2a^3bc+2a^3b+2a^3+a^2b^2c+a^2b^2+2a^2bc^2+2a^2b+2a^2c^3+ab^4+ab^3+ab^2c^2+2ab^2c+ab^2+2abc^3+abc^2+abc+2ac^4+2ac^3+ac^2+2b^5+b^4c+b^4+b^3c^2+b^3c+b^2c^3+2b^2c+bc^3+bc^2+c^5+c^4, a^5+2a^4c+a^3b^2+2a^3b+a3c^2+2a^3c+2a^3+2a^2b^3+2a^2b^2c+a^2bc^2+2a^2bc+a^2b+2a^2c^2+2a^2c+2ab^3c+ab^3+ab^2c+ab^2+2abc^3+2abc^2+abc+ab+2ac^4+ac^2+ac+b^5+b^4c+2b^4+b^3c^2+b^3c+2b^3+2b^2c^3+2b^2c^2+b^2+bc^4+bc+2c^4+2c^2, 2a^4b+a^4c+2a^4+2a^3b^2+a^3b+a^3c^2+a^3c+a^3+2a^2b^2c+2a^2b^2+a^2bc^2+2a^2bc+2a^2b+2a^2c^2+2a^2c+2a^2+ab^4+2ab^3c+ab^2c^2+ab^2c+ab^2+abc^3+abc^2+2ab+ac^2+2ac+a+2b^4c+2b^2c^3+2b^2c^2+b^2c+bc^4+bc+b+c^5+c^4+c^3+c^2+2c\rangle$.

17. **m52470** mit $a > b > c$:
Ideal $I = \langle a^4b+2a^4c+2a^3b+a^3c+2a^2b^2c+a^2c^3+2ab^3+2ac^3+2b^5+2b^2c^2+2bc^4+2c^5, 2a^4c+2a^3b^2+a^3bc+2a^3c^2+2a^2b^3+2a^2bc^2+2a^2bc+a^2c+ab^3+abc^3+2ac^3+ac^2+2b^2c^2+2bc^3+c^3, a^5+a^4b+2a^4+2a^2b^3+a^2bc+ab^2c+2ab^2+abc+2ac^4+2ac+b^4c+2b^3c+2b^2c^2+b^2+2bc^4+2c^5+2c^3\rangle$.

18. **m52430** mit $a > b > c$:
Ideal $I = \langle 2a^5 + a^4b + 2a^4c + 2a^4 + a^3b^2 + 2a^3bc + 2a^3b + a^3c^2 + 2a^2b^2 + a^2bc^2 + 2a^2bc + a^2c^3 + a^2c^2 + ab^4 + 2abc^3 + abc^2 + ac^3 + b^5 + 2b^4c + b^4 + 2b^2c^3 + 2b^2c^2 + 2c^5 + 2c^4, 2a^5 + 2a^4b + 2a^4 + 2a^3b^2 + 2a^3bc + 2a^3b + a^3 + 2a^2b^2c + 2a^2b^2 + a^2bc + a^2b + a^2c^3 + 2a^2c^2 + ab^4 + 2ab^2c^2 + 2ab^2c + 2ab^2 + abc^3 + 2abc^2 + 2abc + ac^4 + 2ac^3 + ac^2 + 2b^5 + b^4c + 2b^3c^2 + b^3c + 2b^3 + b^2c^2 + 2bc^4 + bc^3 + c^5 + 2c^3, 2a^4b + 2a^4c + 2a^3b^2 + 2a^3b + 2a^3c^2 + a^3c + a^3 + 2a^2b^3 + a^2b^2c + a^2b^2 + 2a^2bc^2 + a^2b + 2a^2c^3 + 2a^2c^2 + 2a^2c + 2a^2 + 2ab^3c + 2ab^3 + ab^2c + 2abc^3 + ab + ac^4 + ac^3 + b^5 + 2b^4 + 2b^3c^2 + b^3 + b^2c^3 + 2b^2c^2 + b^2c + 2bc^3 + 2bc^2 + bc + c^5 + c^4 + c^3 + c^2\rangle$.

19. **m62480** mit $a > b > c$:
Ideal $I = \langle 2a^6 + a^5b + a^4b + 2a^3bc + 2a^2c^2 + ab^4c + ab^4 + ab^3c^2 + 2ab^3c + 2ab^2c^3 + 2abc^2 + ac^4 + 2b^3c^3 + 2b^3c, a^3bc^2 + 2a^3b + a^3c^3 + 2a^3 + 2a^2b^2 + a^2c^4 + a^2c^3 + 2ab^4 + 2ab^2 + abc^2 + ac^5 + 2ac^4 + 2b^4c^2 + 2b^2c^3 + bc^4 + 2c^6, 2a^4bc + 2a^4b + a^3b^2 + 2a^3 + a^2b^4 + a^2b^3c + a^2bc^3 + 2a^2bc + a^2c^4 + ab^5 + 2abc^3 + 2ab + 2b^4 + bc^4 + 2bc + c^5 + 2c^4 + c^3\rangle$.

20. **m63580** mit $a > b > c$:
Ideal $I = \langle a^4bc+2a^4c+a^3b^3+a^3c^3+2a^2b^3+2a^2b^2c+2a^2bc^3+ab^5+2b^6+b^3c^2+2c^5, a^2c^3+2a^2c^2+ab^5+ab^3c^2+ab^2c^3+2abc^2+2b^6+b^5c+2b^5+2b^4c+b^3c^3+b^2c^3+2bc^3+2c^5, 2a^5b+a^4b+a^4c+a^3c^3+2a^3c+2a^3+2a^2b^4+a^2bc^2+a^2c^2+ab^3c^2+abc^3+2ac^5+2b^4c+b^2c^2+2b^2c+2bc^5\rangle$.

21. **issac97** mit $x > y > z > w$:
Ideal $I = \langle 8w^2 + 5xw + 2zw + 3w + 5x^2 + 7y^2 + 7z^2 - 7x + 2xy - 7y - 4wy - 8z - 7xz - 8yz + 8, 3w^2 + 9w + 4x^2 + 9y^2 + 7z^2 + 7x - 5wx + 2xy + 5y - 3wy + 6yz + 7z - 6wz - 2xz + 5, -2w^2 + 9xw + 9yw + 8x^2 + 6y^2 - 4w + 8x + 9xy + 4y + 8z - 7wz - 3xz - 7yz - 6z^2 + 2, 7w^2 + 5xw + 3yw + 2x^2 - 5w + 4x + 9xy + 6y - 4y^2 - 9z - 5wz - 7xz - 5yz - 4z^2 + 2\langle$.

22. **ubikker** mit $x > y > z > t$:
Ideal $I = \langle x^2 + xy + y^2 - 2xz - 4yz + 3z^2 - 3xt + 2yt + t^2 - 3x - 2y + 3z - 2t - 2, 2x^2 - xy + y^2 - xz - yz - 6z^2 - xt + yt - 5zt - 3t^2 - 5x + y + 5z + 2t + 5, -3 - 3xy + 2xz + xt^2 - 5xz^2 - 5z^2t - 3xt - 2zt + xyz + xyt - x^2z + x^2 - y^2 + 2z^2 + 11z - 2t - x + y + x^3 + y^3 - 3z^3 + 2t^3 - 3t^2 - 5y^2z + 7yz^2, -15 + 2xy + 11xt^2 + 5xz^2 - zt - 4xyz + 6xyt - x^2z + 3x^2 + 2y^2 - z^2 + 4z - 10t - 35x - 14y - x^3 + 6y^3 + 15z^3 + 4t^3 + 5t^2 + 6y^2z + 4yz^2 - xzt + 6x^2y - 12xy^2 - 7y^2t + 2yt\rangle$.

23. **wang6** mit $q > c > p > d$:
Ideal $I = \langle 2cdpq - 4cd + 2pq + 4pqc - 4d^2q - 2d^2pq + p^2d^2 - 2c^2q + c^2q^2 + 2cdq - 2cdq^2 + d^2q^2 - 2cdp - 8p + c^2 + 4d^2 - 2q + 10p^2 + 2, 2dpq + 4dp^2 + dq - 7dp + cp - 3pqc + 4c, -2p^2 + 8p - 2 - 2pq - 2q, 4 - 4p - 4q^2 + 3c^2q^2 - 6c^2q + 3c^2 + 9p^2d^2 + 6d^2pq - 3d^2q^2 - 24pd^2 + 12d^2 + 4p^2 + 12cdp + 12cdq + 12cdpq - 12cd\rangle$.

24. **cassoumod1** mit $a > b > c > d$:
Ideal $I = \langle 15a^4bc^2 + 6a^4b^3 + 21a^4b^2c - 144a^2b - 8a^2b^2d - 28a^2bcd - 648a^2c + 36c^2d + 9a^4c^3 - 120, 30b^3a^4c - 32cd^2b - 720ca^2b - 24b^3a^2d - 432b^2a^2 + 576db - 576cd + 16ba^2c^2d + 16c^2d^2 + 16d^2b^2 + 9b^4a^4 + 5184 + 39c^2a^4b^2 + 18c^3a^4b - 432c^2a^2 + 24c^3a^2d - 16b^2a^2cd - 240b, 216ca^2b - 162c^2a^2 - 81b^2a^2 + 5184 + 1008db - 1008cd + 15b^2a^2cd - 15b3a^2d - 80cd^2b + 40c^2d^2 + 40d^2b^2, 261 + 4ca^2b - 3c^2a^2 - 4b^2a^2 + 22db - 22cd\rangle$.

25. **cyclic5mod1** mit $a > b > c > d > e$:
Ideal $I = \langle a+b+c+d+e, ab+bc+cd+ae+de, abc+bcd+abe+ade+cde, bcd+abce+abde+acde+bcde, abcde-1\rangle$.

26. **noon5** mit $x_1 > x_2 > x_3 > x_4 > x_5$:
Ideal $I = \langle 10x_1^2x_5 + 10x_2^2x_5 + 10x_3^2x_5 + 10x_4^2x_5 - 11x_5 + 10, 10x_1^2x_4 + 10x_2^2x_4 + 10x_3^2x_4 + 10x_4x_5^2 - 11x_4 + 10, 10x_1^2x_3 + 10x_2^2x_3 + 10x_3x_4^2 + 10x_3x_5^2 - 11x_3 + 10, 10x_1x_2^2 + 10x_1x_3^2 + 10x_1x_4^2 + 10x_1x_5^2 - 11x_1 + 10, 10x_1^2x_2 + 10x_2x_3^2 + 10x_2x_4^2 + 10x_2x_5^2 - 11x_2 + 10\rangle$.

27. **noon6** mit $x_1 > x_2 > x_3 > x_4 > x_5 > x_6$:
Ideal $I = \langle 10x_1^2x_6 + 10x_2^2x_6 + 10x_3^2x_6 + 10x_4^2x_6 + 10x_5^2x_6 - 11x_6 + 10, 10x_1^2x_5 + 10x_2^2x_5 + 10x_3^2x_5 + 10x_4^2x_5 + 10x_5x_6^2 - 11x_5 + 10, 10x_1^2x_4 + 10x_2^2x_4 + 10x_3^2x_4 + 10x_4x_5^2 + 10x_4x_6^2 - 11x_4 + 10, 10x_1^2x_3 + 10x_2^2x_3 + 10x_3x_4^2 + 10x_3x_5^2 + 10x_3x_6^2 - 11x_3 + 10, 10x_1x_2^2 + 10x_1x_3^2 + 10x_1x_4^2 + 10x_1x_5^2 + 10x_1x_6^2 - 11x_1 + 10, 10x_1^2x_2 + 10x_2x_3^2 + 10x_2x_4^2 + 10x_2x_5^2 + 10x_2x_6^2 - 11x_2 + 10\rangle$.

28. **exam54** mit $a > b > c > d > x > f$:
Ideal $I = \langle a+b+c+d+x+f, ab+bc+cd+dx+af+xf, abc+bcd+cdx+abf+axf+dxf, abcd+bcdx+abcf+abxf+adxf+cdxf, abcdx+abcdf+abcxf+abdxf+acdxf+bcdxf, abcdxf-1\rangle$.

29. **MK5** mit $x > y > z > t > u > v$:
Ideal $I = \langle 2x^2+2y^2+2z^2+2t^2+2u^2+v^2-v, xy+yz+2zt+2tu+2uv-u, 2xz+2yt+2zu+u^2+2tv-t, 2xt+2yu+2tu+2zv-z, t^2+2xv+2yv+2zv-y, 2x+2y+2z+2t+2u+v-1\rangle$.

30. **cyclic6** mit $x_1 > x_2 > x_3 > x_4 > x_5 > x_6$:
Ideal $I = \langle x_1+x_2+x_3+x_4+x_5+x_6, x_1x_2+x_1x_6+x_2x_3+x_3x_4+x_4x_5+x_5x_6, x_1x_2x_3+x_1x_2x_6+x_1x_5x_6+x_2x_3x_4+x_3x_4x_5+x_4x_5x_6, x_1x_2x_3x_4+x_1x_2x_3x_6+x_1x_2x_5x_6+x_1x_4x_5x_6+x_2x_3x_4x_5+x_3x_4x_5x_6, x_1x_2x_3x_4x_5+x_1x_2x_3x_4x_6+x_1x_2x_3x_5x_6+x_1x_2x_4x_5x_6+x_1x_3x_4x_5x_6+x_2x_3x_4x_5x_6, x_1x_2x_3x_4x_5x_6-1\rangle$.

31. **kats6** (katsura_6) mit $x_0 > x_1 > x_2 > x_3 > x_4 > x_5$:
Ideal $I = \langle x_0+2x_1+2x_2+2x_3+2x_4+2x_5-1, x_0^2+2x_1^2+2x_2^2+2x_3^2+2x_4^2+2x_5^2-x_0, 2x_0x_1+2x_1x_2+2x_2x_3+2x_3x_4+2x_4x_5-x_1, x_1^2+2x_0x_2+2x_1x_3+2x_2x_4+2x_3x_5-x_2, 2x_1x_2+2x_0x_3+2x_1x_4+2x_2x_5-x_3, x_2^2+2x_1x_3+2x_0x_4+2x_1x_5-x_4\rangle$.

32. **kats7** (katsura_7) mit $x_0 > x_1 > x_2 > x_3 > x_4 > x_5 > x_6$:
Ideal $I = \langle x_0+2x_1+2x_2+2x_3+2x_4+2x_5+2x_6-1, x_0^2+2x_1^2+2x_2^2+2x_3^2+2x_4^2+2x_5^2+2x_6^2-x_0, 2x_0x_1+2x_1x_2+2x_2x_3+2x_3x_4+2x_4x_5+2x_5x_6-x_1, x_1^2+2x_0x_2+2x_1x_3+2x_2x_4+2x_3x_5+2x_4x_6-x_2, 2x_1x_2+2x_0x_3+2x_1x_4+2x_2x_5+2x_3x_6-x_3, x_2^2+2x_1x_3+2x_0x_4+2x_1x_5+2x_2x_6-x_4, 2x_2x_3+2x_1x_4+2x_0x_5+2x_1x_6-x_5\rangle$.

33. **dessin1** mit $a > b > u > v > w > x > y > z$:
Ideal $I = \langle 6zab+10vax+8yau-162a2u+16uw+14xb+48aw, 15zau-162a2v-312ab+24aw+27xu+24yb+18vay+30vw+84xa, -240a+420z-64v+112y, 180za-284va-162a2+60vy+50ya+70w+55zu+260x-112b, 66za+336y+90x+78vz-1056a-90u, 136z-136, 4vaw+2yab+6bw-162a2b+3xua, 28vaz+192w+128ya+36xb+36zb-300au+40yu-648a2+44vx\rangle$.

34. **exam57** mit $a > b > x > y > z > u > v > w$:
Ideal $I = \langle 36z - 136, -240a + 112y + 420z - 64v, 66az + 78zv - 1056a + 90x + 336y - 90u, -162a^2 + 50ay + 180az + 55zu - 284av + 60yv - 112b + 260x + 70w, 28azv - 648a^2 + 36bx + 128ay + 36bz - 300au + 40yu + 44xv + 192w, 15azu - 162a^2v + 18ayv - 312ab + 84ax + 24by + 27xu + 24aw + 30vw, 6abz - 162a^2u + 8ayu + 10axv + 14bx + 48aw + 16uw, -162a^2b + 2aby + 3axu + 4avw + 6bw \rangle$.

35. **virasoro** mit $x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > x_7 > x_8$:
Ideal $I = \langle 8x_1^2 + 8x_1x_2 + 8x_1x_3 + 2x_1x_4 + 2x_1x_5 + 2x_1x_6 + 2x_1x_7 - 8x_2x_3 - 2x_4x_7 - 2x_5x_6 - x_1, 8x_1x_2 - 8x_1x_3 + 8x_2^2 + 8x_2x_3 + 2x_2x_4 + 2x_2x_5 + 2x_2x_6 + 2x_2x_7 - 2x_4x_6 - 2x_5x_7 - x_2, -8x_1x_2 + 8x_1x_3 + 8x_2x_3 + 8x_3^2 + 2x_3x_4 + 2x_3x_5 + 2x_3x_6 + 2x_3x_7 - 2x_4x_5 - 2x_6x_7 - x_3, 2x_1x_4 - 2x_1x_7 + 2x_2x_4 - 2x_2x_6 + 2x_3x_4 - 2x_3x_5 + 8x_4^2 + 8x_4x_5 + 2x_4x_6 + 2x_4x_7 + 6x_4x_8 - 6x_5x_8 - x_4, 2x_1x_5 - 2x_1x_6 + 2x_2x_5 - 2x_2x_7 - 2x_3x_4 + 2x_3x_5 + 8x_4x_5 - 6x_4x_8 + 8x_5^2 + 2x_5x_6 + 2x_5x_7 + 6x_5x_8 - x_5, -2x_1x_5 + 2x_1x_6 - 2x_2x_4 + 2x_2x_6 + 2x_3x_6 - 2x_3x_7 + 2x_4x_6 + 2x_5x_6 + 8x_6^2 + 8x_6x_7 + 6x_6x_8 - 6x_7x_8 - x_6, -2x_1x_4 + 2x_1x_7 - 2x_2x_5 + 2x_2x_7 - 2x_3x_6 + 2x_3x_7 + 2x_4x_7 + 2x_5x_7 + 8x_6x_7 - 6x_6x_8 + 8x_7^2 + 6x_7x_8 - x_7, -6x_4x_5 + 6x_4x_8 + 6x_5x_8 - 6x_6x_7 + 6x_6x_8 + 6x_7x_8 + 8x_8^2 - x_8 \rangle$.

36. **redeco8** mit $x_1 > x_2 > x_3 > x_4 > x_5 > u_8 > x_6 > x_7$:
Ideal $I = \langle -7u_8 + x_7, x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + 1, x_1x_7 - 6u_8 + x_6, x_1x_6 + x_2x_7 + x_5 - 5u_8, x_1x_5 + x_2x_6 + x_3x_7 + x_4 - 4u_8, x_1x_4 + x_2x_5 + x_3x_6 + x_4x_7 + x_3 - 3u_8, x_1x_3 + x_2x_4 + x_3x_5 + x_4x_6 + x_5x_7 + x_2 - 2u_8, x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_1 - u_8 \rangle$.

37. **schwarz8** mit $x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > x_7 > x_8$:
Ideal $I = \langle 2x_4x_5 + 2x_3x_6 + 2x_2x_7 + 2x_1x_8 + x_8, x_4^2 + 2x_3x_5 + 2x_2x_6 + 2x_1x_7 + x_8^2 + x_7, 2x_3x_4 + 2x_2x_5 + 2x_1x_6 + 2x_7x_8 + x_6, x_3^2 + 2x_2x_4 + 2x_1x_5 + x_7^2 + 2x_6x_8 + x_5, 2x_2x_3 + 2x_1x_4 + 2x_6x_7 + 2x_5x_8 + x_4, x_2^2 + 2x_1x_3 + x_6^2 + 2x_5x_7 + 2x_4x_8 + x_3, 2x_1x_2 + 2x_5x_6 + 2x_4x_7 + 2x_3x_8 + x_2, x_1^2 + x_5^2 + 2x_4x_6 + 2x_3x_7 + 2x_2x_8 + x_1 \rangle$.

38. **filter9** mit $a > b > m_1 > m_2 > m_3 > m_4 > m_5 > m_6 > m_7$:
Ideal $I = \langle m_2m_4m_6 - 1/100, am_4b - 7/500, a^2 + m_1^2 - 2/25, b^2 + m_7^2 - 37/50, m_3^2 + m_5^2 + m_4^2 + m_2^2 + m_6^2 - 9401/10000, m_4^2m_6^2 + m_2^2m_4^2 + m_3^2m_6^2 + m_2^2m_5^2 + m_3^2m_5^2 + m_2^2m_6^2 - 38589/1000000, m_1m_3m_5m_7 - m_6m_1m_3b + m_2m_6ab - m_2am_5m_7 + 81/10000, -m_1m_2m_3m_4b - am_4m_5m_6m_7 + am_4bm_6^2 + am_2^2m_4b + 39/25000, -m_4^2m_7^2 - m_3^2m_7^2 + 2m_5m_6bm_7 - m_2^2m_7^2 - m_5^2m_7^2 - m_3^2b^2 - b^2m_6^2 - m_2^2b^2 + 27173/40000 \rangle$.

39. **dessin2** mit $a > b > c > d > u > v > w > x > y > z$:
Ideal $I = \langle 16aw + 18bv + 20cu, -80d + 180y + 855z, 7av + 8bu, 210z - 210, 40ay + 44bx + 48cw + 52dv + 280u, 27ax + 30bw + 33cv + 36du, 55az + 60by + 65cx + 70dw + 80u + 375v, 78bz + 84cy + 90dx - 170a + 102v + 480w, 136dz - 114c + 152x + 720y, 105cz + 112dy - 144b + 126w + 595x \rangle$.

40. **benchmarkD1** mit $x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > x_7 > x_8 > x_9 > x_{10} > x_{11} > x_{12}$:
Ideal $I = \langle -10000x_3x_{10}x_{11} - 10000x_5x_{10}x_{11} - 10000x_7x_{10}x_{11} + 10000x_4x_{12} + 10000x_6x_{12} + 10000x_8x_{12} - 4077, 2000x_2x_4x_9 + 2000x_2x_6x_9 + 2000x_2x_8x_9 + 2000x_1x_{10} - 3823, 10000x_3x_9 + 10000x_5x_9 + 10000x_7x_9 - 19791, 3750x_2x_4 + 2500x_2x_6 + 1250x_2x_8 - 5077, 7500x_1x_4 + 5000x_1x_6 + 2500x_1x_8 - 4293, 30000x_3 + 20000x_5 + 10000x_7 - 39701, x_{11}^2 + x_{12}^2 - 1, x_9^2 + x_{10}^2 - 1, x_7^2 + x_8^2 - 1, x_5^2 + x_6^2 - 1, x_3^2 + x_4^2 - 1, x_1^2 + x_2^2 - 1 \rangle$.

41. **chemkin** mit $c_1 > c_2 > c_3 > z_2 > z_3 > z_4 > z_5 > y_2 > y_3 > y_4 > y_5 > x_3 > x_4$:
Ideal $I = \langle z_2 + z_3 + z_4 + z_5, 33461c_3 - 94642, c_2 + y_2 + y_3 + y_4 + y_5, 9c_2 - 8, 36926c_1 - 208885, -3c_3y_2 + 3x_3 + 3x_4 + 8, -c_1y_2 + 9y_2^2 + z_2, z_5^2 + y_5^2 - c_2, 3z_4z_5 + 3y_4y_5 + x_4 - 1, z_4^2 + y_4^2 + x_4^2 - 1, 3z_3z_4 + 3y_3y_4 + 3x_3x_4 - 1, z_3^2 + y_3^2 + x_3^2 - 1, -3c_3y_2x_3 + 3z_2z_3 + 3y_2y_3 + 3x_3 - 1 \rangle$.

A.2 Homogene Ideale

1. **Fate** mit $p > q > r > s$:
   Ideal $I = \langle s^3+2r^3+2q^3+2p^3, s^5+2r^5+2q^5+2p^5, -s^5+(r+q+p)s^4+(r^2+(2q+2p)r+q^2+2pq+p^2)s^3+(r^3+q^3+p^3)s^2+(3r^4+(2q+2p)r^3+(4q^3+4p^3)r+3q^4+2pq^3+4p^3q+3p^4)s+(4q+4p)r^4+(2q^2+4pq+2p^2)r^3+(4q^3+4p^3)r^2+(6q^4+4pq^3+8p^3q+6p^4)r+4pq^4+2p^2q^3+4p^3q^2+6p^4q\rangle$.

2. **sym34** mit $x > y > z > h$:
   Ideal $I = \langle yz^4+xh^4-2h^5, x^4z+yh^4-2h^5, xy^4+zh^4-2h^5\rangle$.

3. **weispfenning94** mit $x > y > z > h$:
   Ideal $I = \langle y^4+xy^2z+x^2h^2-2xyh^2+y^2h^2+z^2h^2, xy^4+yz^4-2x^2yh^2-3h^5, -x^3y^2+xyz^3+y^4h+xy^2zh-2xyh^3\rangle$.

4. **symmetric47** mit $a > b > c > d$:
   Ideal $I = \langle a^7+b^7, b^7+c^7, c^7+d^7, a6b+b6c+c6d+d6a\rangle$.

5. **gerhard1** mit $w > t > x > y > z$:
   Ideal $I = \langle 7w^2y+4x^2y+xy^2+2wtz, 3w^3tz+3w^2yz^2+2yz^4, 2t^2x^5y^3+5w^4t^3x^2z\rangle$.

6. **liu** mit $x > y > z > t > a > h$:
   Ideal $I = \langle yz-yt-xh+ah, zt-zx-yh+ah, tx-yt-zh+ah, xy-zx-th+ah\rangle$.

7. **hf744** mit $U_7 > U_6 > U_5 > U_4 > U_3 > U_2 > u_7 > u_6 > u_5 > u_4 > u_3 > u_2 > h$:
   Ideal $I = \langle U_7u_7-1h^2, U_6u_6-1h^2, U_5u_5-1h^2, U_4u_4-1h^2, U_3u_3-1h^2, U_2u_2-1h^2, 16U_4u_5u_3+16U_4u_5u_2+16U_3u_5u_2+16U_3u_4u_2+8U_4u_5h+8U_3u_5h+8U_2u_5h+8U_3u_4h+8U_2u_4h+8U_2u_3h+18u_5h^2+18u_4h^2+18u_3h^2+18u_2h^2+11h^3, 16U_5U_3u_4+16U_5U_2u_4+16U_5U_2u_3+16U_4U_2u_3+8U_5u_4h+8U_5u_3h+8U_4u_3h+8U_5u_2h+8U_4u_2h+8U_3u_2h+18U_5h^2+18U_4h^2+18U_3h^2+18U_2h^2+11h^3, 8U_7u_6+8U_6u_6+8U_7u_5+8U_6u_5+8U_5u_5+8U_7u_4+8U_6u_4+8U_5u_4+8U_4u_4+8U_7u_3+8U_6u_3+8U_5u_3+8U_4u_3+8U_3u_3+8U_7u_2+8U_6u_2+8U_5u_2+8U_4u_2+8U_3u_2+8U_2u_2-17h^2, 8U_6u_7+8U_5u_7+8U_4u_7+8U_3u_7+8U_2u_7+8U_6u_6+8U_5u_6+8U_4u_6+8U_3u_6+8U_2u_6+8U_5u_5+8U_4u_5+8U_3u_5+8U_2u_5+8U_4u_4+8U_3u_4+8U_2u_4+8U_3u_3+8U_2u_3+8U_2u_2-17h^2, 2U_7+2U_6+2U_5+2U_4+2U_3+2U_2+h, 2u_7+2u_6+2u_5+2u_4+2u_3+2u_2+h\rangle$.

A.3 Eigenschaften der Beispiele

In den folgenden Tabellen werden Eigenschaften von dem gegebenen Erzeugendensystem des Ideals I mehrerer Beispiele und der reduzierten Gröbnerbasis von I bzgl. $\succ_{rlex}$ bzw. $\succ_{lex}$ aufgelistet. Dies sind die Anzahl ihrer Polynome (#P) bzw. ihrer Monome (#M), ihr höchster Totalgrad (Td), die Anzahl der Ziffern ihres größten Leitkoeffizientens (#Z) und der Speicherbedarf (Bt) in Byte sie abzuspeichern.

Beispiele	n	Erzeugendensysteme					Red. Gröbnerbasen bzgl. $\succ_{rlex}$					Red. Gröbnerbasen bzgl. $\succ_{lex}$				
		#P	#M	Td	#Z	Bt	#P	#M	Td	#Z	Bt	#P	#M	Td	#Z	Bt
ex2	3	3	13	4	1	55	15	379	7	13	3942	5	182	36	249	38309
ex3	3	3	13	4	1	53	15	477	7	10	5086	8	268	35	178	42926
bsp6†	3	2	10	4	1	39	4	74	10	1	353	10	2679	35	34	108130
bsp7†	3	2	10	4	1	41	5	79	8	2	419	7	2032	44	136	228307
tran1†	3	2	10	4	1	35	3	49	6	2	235	7	1146	20	17	23668
tran2	3	3	12	4	1	50	13	295	8	15	2658	3	138	45	293	28027
qnt2†	3	2	9	5	2	47	5	44	8	4	287	4	1429	42	609	604355
trinks1	6	6	22	3	2	97	13	124	3	54	3573	6	66	10	297	17335
issac97	4	4	60	2	1	216	12	180	5	79	5440	4	68	16	423	22420
wang6	4	4	47	4	1	213	18	374	4	28	6404	4	100	24	267	20706
ubikker	4	4	81	3	1	302	17	519	7	113	22159	4	146	36	1240	138809
MK5	6	6	36	2	1	120	22	559	6	58	11105	6	198	32	555	95966
cyc6	6	6	32	6	1	482	45	1135	9	11	22151	17	330	48	17	9395
s7	7	7	35	2	1	119	64	621	8	6	4043	8	82	127	107	6078
kats6	6	6	36	2	1	325	22	528	6	19	9769	6	193	32	402	69014
kats7	7	7	47	2	1	442	41	1923	7	36	54150	7	449	64	1964	774742
redeco7	7	7	34	2	1	157	33	636	6	6	6269	7	200	32	74	14052
redeco8	8	8	43	2	1	204	65	2273	7	9	26642	8	457	64	205	86701
niermann1	3	3	15	5	1	59	16	384	9	3	1874	3	164	54	200	22871
cyc5mod1	4	5	22	5	1	77	25	1061	7	10	9587	8	506	61	204	93067
casmod1	4	4	45	8	1	313	18	414	4	157	55760	4	92	22	833	58901
dessin1	8	8	51	3	3	272	39	1385	5	192	102356	8	331	46	3458	979559
dessin2	10	10	40	2	3	199	45	1263	4	255	131486	10	349	42	1573	478462

Tabelle A.1: Eigenschaften der Polynommenge (†kein nulldimensionales Ideal)

Beispiele	n	Erzeugendensysteme					Red. Gröbnerbasen bzgl. $\succ_{rlex}$					Red. Gröbnerbasen bzgl. $\succ_{lex}$				
		#P	#M	Td	#Z	Bt	#P	#M	Td	#Z	Bt	#P	#M	Td	#Z	Bt
arnborg7	3	4	60	2	1	1607	12	180	5	79	1607	4	68	16	423	22420
rose	3	3	29	9	4	309	19	537	10	44	14553	3	72	34	334	13915
rose2	3	3	29	9	4	309	18	501	10	43	13707	3	105	136	903	55301
vermeer76†	5	4	17	5	1	66	20	433	6	9	3788	12	426	9	16	6643
eco7	7	7	34	3	1	203	32	747	4	8	7891	7	231	32	87	14935
hairer†	13	11	40	4	2	291	64	3704	5	17	70170	33	1066	7	4	12081
exam7	3	3	20	5	1	82	9	295	9	34	6363	3	123	40	696	58669
exam12	3	3	23	4	1	90	9	180	7	32	2846	3	75	24	304	15939
exam14	4	4	28	4	2	138	10	185	7	27	3939	4	100	24	327	25494
exam57	8	8	51	3	2	269	41	1509	5	289	229512	8	331	46	10449	2951662
exam60	3	3	12	4	1	50	13	295	8	15	2658	3	138	45	293	28027
noon4	4	4	20	3	2	168	28	443	7	6	4831	8	373	26	64	19011
noon5	5	5	30	3	2	265	72	2343	9	8	31050	15	1521	41	161	173603
noon6	6	6	42	3	2	384	187	12745	11	10	200862	29	6345	59	359	1691061
vsoro	8	8	93	2	1	701	128	18425	9	39	541639	71	8058	22	86	730478
bmarkD1	12	12	46	3	5	381	86	1487	4	775	609163	12	300	48	11543	2767791
chemkin	13	13	46	3	5	232	56	1825	4	457	646911	13	416	40	8066	2903272
m41	3	3	26	4	1	111	16	722	9	33	13731	6	347	59	719	213266
m51390	3	3	21	5	1	91	24	1960	12	33	34642	7	673	101	1310	770013
m52470	3	3	44	5	1	212	25	2115	13	117	95318	12	1248	104	2801	3238406
m52430	3	3	94	5	1	448	23	2269	13	130	120208	10	1085	108	3405	3348435
m62480	3	3	48	6	1	237	27	4445	14	71	162409	10	1869	189	5927	10039458
m63580	3	3	41	6	1	206	32	4414	12	51	160623	16	2293	141	4290	9317166

Tabelle A.2: Eigenschaften der Polynommenge (†kein nulldimensionales Ideal)

Anhang B

Gröbnerbasen im Gröbnerwalkalgorithmus

Gegeben sei ein Erzeugendensystem G eines Ideals I im Polynomring $\mathbb{Q}[x_1, \ldots, x_n]$ über dem rationalen Körper $\mathbb{Q}$. Mit Hilfe des Gröbnerwalkalgorithmus wird die reduzierte lexikographische Gröbnerbasis von I ausgerechnet. Im letzten Schritt dieses Algorithmus tritt ein „großes“ Erzeugendensystem $(G_{t-1})_{\omega_t}$ des ω_t-homogenen Ideals $\langle I_{\omega_t} \rangle$ mit $\omega_t := (1, 0, \ldots, 0) \in \mathbb{Z}^n$ auf. Die Berechnung der reduzierten lexikographischen Gröbnerbasis von $\langle (G_{t-1})_{\omega_t} \rangle$ ist allerdings sehr aufwendig (siehe Tabelle 5.3).

In diesem Anhang geben wir die Größe von G_{t-1} und $(G_{t-1})_{\omega_t}$ für mehrere Beispiele an, in denen die Berechnung wegen des Speichermangels abbricht. Außerdem sind auch die Initialterme von $(G_{t-1})_{\omega_t}$ bzgl. der lexikographischen Ordnung abgedruckt.

1. **Beispiel wang6** mit $q > c > p > d$:

   - G_{t-1} besteht aus 16 Polynomen. Ein Polynom davon hat 8 Monome und jedes der anderen besteht aus 25 Monomen.
   - $(G_{t-1})_{\omega_t}$ besteht aus 15 Polynomen und einem Monom. Ein Polynom davon hat 8 Monome und jedes der anderen besteht aus 25 Monomen. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{ -45788547433617803226564750427154640c^3, -5076204668786475314749558696507 2c^3,$$
$$350071632961852997732523390129 6c^3, -937921690579060342372937919696c^3, 3cp^3,$$
$$-66518838059019637223955156758064c^3, -59763764007343811654813651888 16c^3,$$
$$955284651688475001023856667972 8c^3, 464843800286985160276652040634 08c^3d,$$
$$276692738266062595402769071806c^3p, 3018909467211813644010305611816407 1968c^4,$$
$$206597244571993404567400906948 48q, 208493261677231439047412768007198 7597176062448c^3,$$
$$-40568775114359194453023553845 936c^3, 668985318438592288683832382721 6c^3,$$
$$-88060782938445853006373977474 512c^3\}.$$

2. **Beispiel cassoumod1** mit $a > b > c > d$:

   - G_{t-1} besteht aus 16 Polynomen. Jedes Polynom davon hat 23 Monome.

- $(G_{t-1})_{\omega_t}$ besteht aus 15 Polynomen und einem Monom. Jedes Polynom davon hat 23 Monome. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{-\underbrace{9242369\ldots864974233}_{160\text{ Ziffern}}b^3,\ \underbrace{15736552\ldots7054868 48}_{161\text{ Ziffern}}b^3,\ -\underbrace{10694981\ldots2550272}_{171\text{ Ziffern}}b^3,$$
$$-\underbrace{169815\ldots437470208}_{161\text{ Ziffern}}b^3,\ \underbrace{41162678\ldots969368064}_{158\text{ Ziffern}}b^3,\ -\underbrace{136772464\ldots704085504}_{161\text{ Ziffern}}b^3,$$
$$-\underbrace{2690195\ldots2848731136}_{170\text{ Ziffern}}b^3,\ -\underbrace{59250337\ldots6950588416}_{159\text{ Ziffern}}b^3,\ \underbrace{18396199\ldots02035456}_{161\text{ Ziffern}}b^3,$$
$$-\underbrace{329054893\ldots273426944}_{169\text{ Ziffern}}b^3,\ \underbrace{41380191\ldots31742720}_{162\text{ Ziffern}}b^3d,\ \underbrace{54009878\ldots4184100208640}_{171\text{ Ziffern}}b^3c,$$
$$\underbrace{1530279\ldots8828392448}_{171\text{ Ziffern}}b^4,\ \underbrace{2036215\ldots5622989440}_{165\text{ Ziffern}}a^2,\ -\underbrace{424086416\ldots2325652480}_{196\text{ Ziffern}}b^3,$$
$$\underbrace{149520212\ldots03366924175 36}_{166\text{ Ziffern}}b^3\}.$$

3. **Beispiel cyclic6** mit $x_1 > x_2 > x_3 > x_4 > x_5 > x_6$:

- G_{t-1} besteht aus 70 Polynomen. 15 Polynome davon haben 24 Monome, 16 davon haben 25 Momone, 17 davon haben 26 Monome, 20 davon haben 27 Monome und jedes der anderen besteht aus 6 bzw. 9 Monomen.

- $(G_{t-1})_{\omega_t}$ besteht aus 69 Polynomen und einem Monom. 15 Polynome davon haben 24 Monome, 16 davon haben 25 Momone, 17 davon haben 26 Monome, 20 davon haben 27 Monome und das andere besteht aus 9 Monomen. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{25288x_3^2x_5^2, 12644x_3^2x_4x_6, 436x_3^2x_4x_5, 25288x_3^2x_4^2, 12644x_3^3x_6, 6322x_3^3x_5, 25288x_3^3x_4, 3161x_3^4,$$
$$50576x_2x_5x_6^2, 6322x_2x_5^2x_6, 12644x_2x_5^3, 50576x_2x_4x_6^2, 25288x_2x_4x_5x_6, 12644x_2x_4x_5^2, 42785523x_3x_5^3x_6,$$
$$42785523x_3x_5^4, 5186124x_3x_4x_6^3, 19015788x_3x_4x_5x_6^2, 19015788x_3x_4x_5^2x_6, 15558372x_3x_4x_5^3,$$
$$85571046x_3x_4^2x_6^2, 85571046x_3x_4^2x_5x_6, 85571046x_3x_4^2x_5^2, 16937424x_4^4x_6^2, 16937424x_4^4x_5x_6,$$
$$16937424x_4^4x_5^2, 2419632x_4^5x_6, 16937424x_4^5x_5, 627312x_4^6, 5444172x_3x_5x_6^4, 38109204x_3x_5^2x_6^3,$$
$$263677416x_4^2x_5^5, 1582064496x_4^3x_6^4, 1582064496x_4^3x_5x_6^3, 1582064496x_4^3x_5^2x_6^2, 1582064496x_4^3x_5^3x_6,$$
$$1582064496x_4^3x_5^4, 3164128992x_3x_6^6, 3164128992x_2x_6^6, 273360245904120x_2x_3, -712206904595304x_2x_3,$$
$$-6315376062068592x_2x_3, -2647167012271560x_2x_3, -3255305371154544x_2x_3, 50467833843336x_2x_3,$$
$$3867299493132x_2x_3, -1204392357277448x_2x_3, -389056839538020x_2x_3, -300426431152638x_2x_3,$$
$$188274x_2x_4x_5, 54585234x_2x_4x_5, -3287658x_2x_4x_5, 24239886x_2x_4x_5, -23554446x_2x_4x_5,$$
$$7300338x_2x_4x_5, -31332678x_2x_4x_5, 173952x_2x_4x_5, -10875x_2x_4x_5, -84680964x_2x_4x_5,$$
$$306538x_2x_4x_5, -13142574x_2x_4x_5\}.$$

4. **Beispiel dessin1** mit $a > b > u > v > w > x > y > z$:

- G_{t-1} besteht aus 38 Polynomen. Vier Polynome davon haben 29 Monome, zwei davon haben 36 Monome, drei davon haben 37 Monome, fünf davon haben 43 Monome, acht davon haben 46 Monome, acht davon haben 47 Monome und jedes der anderen besteht aus 2, 5, 4, 9, 11, 17, 19 bzw. 39 Monomen.

- $(G_{t-1})_{\omega_t}$ besteht aus 37 Polynomen und einem Monom. Diese Polynome stimmen mit der Polynomen von G_{t-1} überein. Die Menge der Leitterme von

$(G_{t-1})_{\omega_t}$ ist:

$$\{z, 20u, 60a, -9600b, 1548bx, -9334784bv, -871890654464bv, -4729059697119068160000b^2,$$
$$-3514711927047690240000b^2, -936782178764472152064000b^2, 7137524597004657745920000b^2,$$
$$-5463258562919816913066393600b^2v, 2611162143321471722680299744279920640b^2v,$$
$$29761141406443990841386598400b^2v, -17839157122668229079962153926131712000b^2v,$$
$$-\underbrace{126952\ldots144000}_{54\text{ Ziffern}}b^3, \underbrace{174325\ldots8016000}_{49\text{ Ziffern}}b^3, \underbrace{1755932\ldots0192000}_{55\text{ Ziffern}}b^3,$$
$$29999669606369085519714096921083904 0b^2v, -\underbrace{452410\ldots9476480}_{41\text{ Ziffern}}b^2v, \underbrace{9381557\ldots53216000}_{45\text{ Ziffern}}b^3,$$
$$\underbrace{58031407\ldots22272000}_{55\text{ Ziffern}}b^3, \underbrace{74959016\ldots1856000}_{86\text{ Ziffern}}b^3, \underbrace{26493896\ldots1104000}_{86\text{ Ziffern}}b^3, \underbrace{14466592\ldots1424000}_{116\text{ Ziffern}}b^3,$$
$$-\underbrace{38109520\ldots0416000}_{86\text{ Ziffern}}b^3, -\underbrace{145485854\ldots0944000}_{87\text{ Ziffern}}b^3, \underbrace{40160978\ldots6829056000}_{114\text{ Ziffern}}b^3,$$
$$-\underbrace{3121655\ldots049728000}_{86\text{ Ziffern}}b^3, \underbrace{5082893\ldots920000000}_{87\text{ Ziffern}}b^3y, \underbrace{25237804\ldots96000000000}_{112\text{ Ziffern}}b^3w,$$
$$\underbrace{6931218\ldots080000000}_{86\text{ Ziffern}}b^3v, \underbrace{38765267\ldots600000000}_{115\text{ Ziffern}}b^4, -\underbrace{3652796\ldots072832000}_{89\text{ Ziffern}}b^3,$$
$$-\underbrace{1867158607\ldots2954368000}_{131\text{ Ziffern}}b^3, -\underbrace{95012086\ldots86144000}_{149\text{ Ziffern}}b^3, -\underbrace{259532\ldots912384000}_{141\text{ Ziffern}}b^3,$$
$$-\underbrace{5188121\ldots94016000}_{187\text{ Ziffern}}b^3\}.$$

5. **Beispiel dessin2** mit $a > b > c > d > u > v > w > x > y > z$:

- G_{t-1} besteht aus 51 Polynomen. Zwei Polynome davon haben 11 Monome, zwei davon haben 8 Monome, zwei davon haben 27 Monome, drei davon haben 31 Monome, drei davon haben 33 Monome, vier davon haben 36 Monome, vier davon haben 38 Monome, vier davon haben 40 Monome, drei davon haben 41 Monome, sechs davon haben 42 Monome, neun davon haben 43 Monome und jedes der anderen besteht aus 2, 3, 4, 6, 13, 17, 16, 20 bzw. 24 Monomen.
- $(G_{t-1})_{\omega_t}$ besteht aus 50 Polynomen und einem Monom. Es gibt nur ein Polynom $g \in G_{t-1}$ mit $in_{\omega_t}(g) \neq g$. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{z, 16d, 12c, -192b, 2880by, 7248bx, 1737780480b^2, 80a, 362400bw, -524758824bv, 270bv,$$
$$-207700383916032bu, 640bu, 847289197377957366988 8bu, 919403392452424089600bu,$$
$$-220876097661419900448985270763520bu, 124974263020627504936233420079104bu,$$
$$\underbrace{16193\ldots20992}_{47\text{ Ziffern}}bu, -\underbrace{28001\ldots60352}_{43\text{ Ziffern}}bu, \underbrace{21012\ldots18240}_{63\text{ Ziffern}}bu, \underbrace{39242\ldots77952}_{79\text{ Ziffern}}bu, -\underbrace{32750\ldots81888}_{32\text{ Ziffern}}bu,$$
$$\underbrace{17077\ldots16384}_{44\text{ Ziffern}}bu, \underbrace{17500\ldots44704}_{59\text{ Ziffern}}bu, \underbrace{17324\ldots98560}_{54\text{ Ziffern}}buw, \underbrace{24927\ldots11360}_{71\text{ Ziffern}}buv, \underbrace{42452\ldots84480}_{87\text{ Ziffern}}bu^2,$$
$$-\underbrace{72425\ldots93504}_{60\text{ Ziffern}}bu, \underbrace{36437\ldots61024}_{78\text{ Ziffern}}bu, \underbrace{19789\ldots56384}_{98\text{ Ziffern}}bu, \underbrace{75476\ldots37120}_{75\text{ Ziffern}}bu, -\underbrace{21316\ldots56384}_{97\text{ Ziffern}}bu,$$
$$\underbrace{37793\ldots73120}_{117\text{ Ziffern}}bu, \underbrace{24156\ldots1088}_{95\text{ Ziffern}}bu, -\underbrace{3554\ldots5680}_{116\text{ Ziffern}}bu, -\underbrace{30140\ldots3840}_{138\text{ Ziffern}}bu, \underbrace{494896\ldots0880}_{115\text{ Ziffern}}bu,$$
$$\underbrace{24774\ldots8160}_{136\text{ Ziffern}}bu, -\underbrace{48093\ldots7920}_{157\text{ Ziffern}}bu, -\underbrace{60822\ldots24640}_{135\text{ Ziffern}}bu, \underbrace{11349\ldots75040}_{158\text{ Ziffern}}bu, -\underbrace{52113\ldots27520}_{155\text{ Ziffern}}bu,$$
$$\underbrace{13527\ldots08320}_{174\text{ Ziffern}}bu, -\underbrace{12931\ldots89440}_{194\text{ Ziffern}}bu, -\underbrace{39739\ldots27040}_{176\text{ Ziffern}}bu, \underbrace{15012\ldots48480}_{197\text{ Ziffern}}bu, \underbrace{21804\ldots71040}_{215\text{ Ziffern}}bu,$$
$$-\underbrace{109819\ldots53120}_{215\text{ Ziffern}}bu, \underbrace{1578164\ldots96640}_{235\text{ Ziffern}}bu, -\underbrace{3409\ldots22240}_{255\text{ Ziffern}}bu, -\underbrace{4535\ldots76640}_{273\text{ Ziffern}}bu\}.$$

Anhang C

Gröbnerbasen im neuen Gröbnerwalkalgorithmus

Wir nehmen an, dass im letzten Schritt des Gröbnerwalkalgorithmus die folgenden Situationen auftreten:

1. G_{t-1} ist die reduzierte Gröbnerbasis von I bzgl. $\succ_{(\omega_{t-1}, lex)}$, wobei ω_{t-1} der vorletzte berechnete dazwischenliegende Gewichtsvektor ist.

2. $(G_{t-1})_{\omega_t}$ ist die Menge der Initialterme jedes Polynom von G_{t-1}. Diese Menge ist die reduzierte Gröbnerbasis von $\langle I_{\omega_t} \rangle$ bzgl. $\succ_{(\omega_{t-1}, lex)}$.

Dann werden die reduzierte Gröbnerbasis M_t von $\langle I_{\omega_t} \rangle = \langle (G_{t-1})_{\omega_t} \rangle$ bzgl. $\succ_{(\omega_t, lex)}$ und eine Gröbnerbasis F_t von I bzgl. $\succ_{(\omega_t, lex)}$ ausgerechnet.

Im neuen Gröbnerwalkalgorithmus ist die reduzierte Gröbnerbasis M_t nach kurzer Zeit erreicht. Die Berechnung von F_t, in der die Linearkombination jedes Polynoms von M_t bzgl. $(G_{t-1})_{\omega_t}$ ausgerechnet werden muss, benötigt allerdings viel Zeit (vgl. Tabelle 5.3).

In diesem Anhang sind die Größen von $(G_{t-1})_{\omega_t}$ und M_t für mehrere Beispiele abgedruckt. Außerdem geben wir die Initialterme von $(G_{t-1})_{\omega_t}$ bzw. M_t bzgl. $\succ_{(\omega_{t-1}, lex)}$ an.

1. **Beispiel qnt1** mit $x > y > z$:

- $(G_{t-1})_{\omega_t}$ besteht aus drei Monomen und 17 Polynomen mit 136 Monomen. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{\underbrace{9435319\ldots67225320}_{513\text{ Ziffern}}y^9z^8, \underbrace{1887063\ldots13445064}_{513\text{ Ziffern}}y^{10}z^7, \underbrace{1887063\ldots34450640}_{514\text{ Ziffern}}y^{11}z^6,$$
$$\underbrace{7548255\ldots37802560}_{514\text{ Ziffern}}y^{12}z^5, \underbrace{1207720\ldots0484096}_{515\text{ Ziffern}}y^{13}z^4, \underbrace{241544\ldots9681920}_{516\text{ Ziffern}}y^{14}z^3,$$
$$\underbrace{24492579\ldots68800}_{520\text{ Ziffern}}y^{15}z^2, \underbrace{146955478\ldots28000}_{522\text{ Ziffern}}y^{16}z, \underbrace{293910957\ldots560000}_{523\text{ Ziffern}}y^{17},$$
$$\underbrace{49868619\ldots12500}_{524\text{ Ziffern}}xy, \underbrace{383348794\ldots28320}_{529\text{ Ziffern}}yz^{17}, \underbrace{14983570\ldots96700}_{522\text{ Ziffern}}y^2z^{16},$$
$$\underbrace{19715223\ldots72325}_{520\text{ Ziffern}}y^3z^{15}, \underbrace{48403189\ldots58916}_{515\text{ Ziffern}}y^4z^{14}, \underbrace{47176597\ldots612660}_{513\text{ Ziffern}}y^5z^{13},$$
$$\underbrace{943531951\ldots7225320}_{513\text{ Ziffern}}y^6z^{12}, \underbrace{47176597\ldots612660}_{513\text{ Ziffern}}y^7z^{11}, \underbrace{94353195\ldots722532}_{512\text{ Ziffern}}y^8z^{10},$$
$$\underbrace{6462973\ldots5000000}_{527\text{ Ziffern}}xz^2, \underbrace{6521117\ldots600000}_{526\text{ Ziffern}}x^4\}.$$

- M_t besteht aus 4 Polynomen und drei Monomen. Zwei Polynome davon haben 135 Monome und jedes der anderen besteht aus 134 bzw. 136 Monomen. Die Menge der Leitterme von M_t ist:

$$\{\underbrace{523568076653\ldots33897267200000}_{5530\text{ Ziffern}}yz^{139}, 2008867838118431438883545422449868800000yz^{140},$$
$$\underbrace{1949087549483\ldots9568998400000}_{5533\text{ Ziffern}}yz^{139}, -\underbrace{1948041670001\ldots24342118400000}_{5538\text{ Ziffern}}yz^{139}, xz^2, xy, x^4\}.$$

2. **Beispiel cassoumod1** mit $a > b > c > d$:

- $(G_{t-1})_{\omega_t}$ besteht aus einem Monom und 15 Polynomen mit 23 Monomen. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{\underbrace{919559\ldots9276160}_{160\text{ Ziffern}}c^2d^2, \underbrace{827603\ldots3485440}_{161\text{ Ziffern}}c^3d, \underbrace{394211\ldots555840}_{168\text{ Ziffern}}c^4, \underbrace{558632\ldots8526720}_{162\text{ Ziffern}}bd^3$$
$$\underbrace{191574\ldots5609920}_{159\text{ Ziffern}}bcd^2, \underbrace{413801\ldots1742720}_{161\text{ Ziffern}}bc^2d, \underbrace{131403\ldots9185280}_{168\text{ Ziffern}}bc^3, \underbrace{229889\ldots7319040}_{160\text{ Ziffern}}b^2d^2,$$
$$\underbrace{413801\ldots1742720}_{161\text{ Ziffern}}b^2cd, \underbrace{219006\ldots530880}_{167\text{ Ziffern}}b^2c^2, \underbrace{413801\ldots1742720}_{161\text{ Ziffern}}b^3d, \underbrace{540098\ldots208640}_{171\text{ Ziffern}}b^3c,$$
$$\underbrace{15302\ldots8392448}_{171\text{ Ziffern}}b^4, \underbrace{2036215\ldots2989440}_{165\text{ Ziffern}}a^2, \underbrace{1238429\ldots4358400}_{194\text{ Ziffern}}d^5, \underbrace{54299\ldots971840}_{165\text{ Ziffern}}cd^4\}.$$

- M_t besteht aus 3 Polynomen und einem Monom. Jedes Polynom davon hat 23 Monome. Die Menge der Leitterme von M_t ist:

$$\{\underbrace{5015328580\ldots26803200}_{829\text{ Ziffern}}d^{21}, \underbrace{5026767\ldots02035200}_{40\text{ Ziffern}}d^{22}, \underbrace{177140711\ldots602841600}_{830\text{ Ziffern}}d^{21}, a^2\}.$$

3. **Beispiel dessin1** mit $a > b > u > v > w > x > y > z$:

- Die Große von $(G_{t-1})_{\omega_t}$ ist auf Seite 123 abgedruckt. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{z, 20u, 60a, 5504v^2, 1548bx, 1423680vy^2, 43158859200vxy, 24722887501017446400 0by^2,$$
$$12361443750508723200 0bvy, 23733972000976748544 0000y^4, 23733972000976748544 0000xy^3,$$
$$50392736511164773171200 0x^2y^2, 30518930218197392903226949632000x^3y,$$
$$20184278818362694041600 00wy^3, 67819844929327539784948776960000wxy^2,$$
$$\underbrace{12241\ldots76000}_{53\text{ Ziffern}}wx^3, \underbrace{52246\ldots40000}_{48\text{ Ziffern}}w^2y^2, \underbrace{22442\ldots560000}_{54\text{ Ziffern}}w^2xy, \underbrace{38148\ldots04000}_{31\text{ Ziffern}}vx^3,$$
$$\underbrace{32637\ldots30400}_{35\text{ Ziffern}}vwx^2, \underbrace{252643\ldots80000}_{45\text{ Ziffern}}vw^2y, \underbrace{448851\ldots20000}_{54\text{ Ziffern}}vw^2x, \underbrace{297825\ldots000000}_{85\text{ Ziffern}}vw^3,$$
$$\underbrace{68074\ldots000000}_{85\text{ Ziffern}}bw^2y, \underbrace{30285\ldots000000}_{113\text{ Ziffern}}bw^3, \underbrace{119130\ldots000000}_{86\text{ Ziffern}}bvw^2, \underbrace{23826\ldots000000}_{86\text{ Ziffern}}b^2wy,$$
$$\underbrace{32740\ldots000000}_{112\text{ Ziffern}}b^2w^2, \underbrace{423574\ldots00000}_{85\text{ Ziffern}}b^2vw, \underbrace{50828\ldots000000}_{87\text{ Ziffern}}b^3y, \underbrace{25237\ldots000000}_{112\text{ Ziffern}}b^3w,$$
$$\underbrace{693121\ldots000000}_{86\text{ Ziffern}}b^3v, \underbrace{387652\ldots0000000}_{115\text{ Ziffern}}b^4, \underbrace{343095\ldots600000}_{86\text{ Ziffern}}x^5, \underbrace{2011693\ldots000000}_{126\text{ Ziffern}}w3x2,$$
$$\underbrace{585369\ldots000000}_{143\text{ Ziffern}}w^4y, \underbrace{1007912\ldots000000}_{145\text{ Ziffern}}w^4x, \underbrace{421591\ldots0000000}_{179\text{ Ziffern}}w^5\}.$$

- M_t besteht aus 7 Polynomen und einem Monom. Sechs Polynome davon haben 47 Monome und das andere besteht aus zwei Monomen. Die Menge der Leitterme von M_t ist:

$$\{z, \underbrace{45900525\ldots0000000}_{3423\text{ Ziffern}}y^{45}, \underbrace{4363094\ldots0000000}_{86\text{ Ziffern}}y^{46}, -\underbrace{1235573974355\ldots81600000000}_{3421\text{ Ziffern}}y^{45},$$
$$-\underbrace{85012175\ldots000000}_{3422\text{ Ziffern}}y^{45}, -\underbrace{127533388\ldots000000}_{3423\text{ Ziffern}}y^{45}, -\underbrace{102756907\ldots000000}_{3420\text{ Ziffern}}y^{45}, a\}.$$

4. **Beispiel dessin2** mit $a > b > c > d > u > v > w > x > y > z$:

- Die Große von $(G_{t-1})_{\omega_t}$ ist auf Seite 123 abgedruckt. Die Menge der Leitterme von $(G_{t-1})_{\omega_t}$ ist:

$$\{z, 16d, 12c, 336y^2, 2880by, 7248bx, 1737780480b^2, 80a, 603396x^2y, 323956608x^3,$$
$$444wxy, 431451889666560w2y, 1036vxy, 9756766742541654528 0vwy, 4204936508570140672uxy,$$
$$6864048177521534837735948288 0ux^2, 22651358985821064964528629350 4uwy,$$
$$\underbrace{30905\ldots150848}_{43\text{ Ziffern}}uwx, \underbrace{376532\ldots51456}_{39\text{ Ziffern}}uvy, \underbrace{24884\ldots115840}_{58\text{ Ziffern}}u^2y, \underbrace{27918\ldots472320}_{73\text{ Ziffern}}u^2x,$$
$$643504516642643891037745152 0bw^2, 8129675124089445675318700354159449292 80bvw,$$
$$\underbrace{17053\ldots59520}_{54\text{ Ziffern}}bv^{22}, \underbrace{173240\ldots698560}_{54\text{ Ziffern}}buw, \underbrace{24927\ldots11360}_{71\text{ Ziffern}}buv, \underbrace{42452\ldots684480}_{87\text{ Ziffern}}bu^2,$$
$$\underbrace{48602\ldots96320}_{55\text{ Ziffern}}w^2x^2, \underbrace{34352\ldots159680}_{134\text{ Ziffern}}w^3x, \underbrace{534951\ldots808320}_{90\text{ Ziffern}}w^4, \underbrace{12463\ldots55680}_{71\text{ Ziffern}}vwx^2,$$
$$\underbrace{101895\ldots439680}_{90\text{ Ziffern}}vw^2x, \underbrace{135139\ldots80000}_{109\text{ Ziffern}}vw^3, \underbrace{107825\ldots306240}_{88\text{ Ziffern}}v^2x^2, \underbrace{8151265\ldots80000}_{108\text{ Ziffern}}v^2wx,$$
$$\underbrace{241101\ldots200000}_{129\text{ Ziffern}}v^2w^2, \underbrace{305672\ldots408000}_{127\text{ Ziffern}}v^3y, \underbrace{306161\ldots720000}_{127\text{ Ziffern}}v^3x, \underbrace{237719\ldots800000}_{148\text{ Ziffern}}v^3w,$$
$$\underbrace{675355\ldots600000}_{126\text{ Ziffern}}uw^3, \underbrace{1018798\ldots200000}_{148\text{ Ziffern}}uvw^2, \underbrace{215618\ldots200000}_{146\text{ Ziffern}}uv^2x, \underbrace{253804\ldots400000}_{164\text{ Ziffern}}uv^2w,$$
$$\underbrace{262204\ldots000000}_{183\text{ Ziffern}}uv^3, \underbrace{390117\ldots800000}_{165\text{ Ziffern}}u^2w^2, \underbrace{1426393\ldots2000000}_{186\text{ Ziffern}}u^2vw, \underbrace{878990\ldots000000}_{203\text{ Ziffern}}u^2v^2,$$
$$\underbrace{608824\ldots8000000}_{202\text{ Ziffern}}u^3w, \underbrace{141817\ldots6000000}_{223\text{ Ziffern}}u^3v, \underbrace{119852\ldots2000000}_{242\text{ Ziffern}}u^4, \underbrace{340163\ldots000000}_{259\text{ Ziffern}}v^5\}.$$

- M_t besteht aus 9 Polynomen und einem Monom. Sieben Polynome davon haben 43 Monome und jedes der anderen besteht aus zwei bzw. drei Monomen. Die Menge der Leitterme von M_t ist:

$$\{z, -\underbrace{2663696428113\ldots4324105385017344}_{1559\text{ Ziffern}}y^{41}, \underbrace{48414742995\ldots1788949716992}_{79\text{ Ziffern}}y^{42},$$
$$\underbrace{72908544962940\ldots1505494976954368}_{1559\text{ Ziffern}}y^{41}, -\underbrace{1194309668563361\ldots2148905914728448}_{1557\text{ Ziffern}}y^{41},$$
$$\underbrace{652293446857853\ldots97730987933696}_{1553\text{ Ziffern}}y^{41}, 16d, -\underbrace{26636964281132\ldots4324105385017344}_{1559\text{ Ziffern}}y^{41},$$
$$-\underbrace{189250341674\ldots054045302784}_{1558\text{ Ziffern}}y^{41}, a\}.$$

Anhang D

Analyse des Perturbationwalkalgorithmus

In diesem Anhang zeigen wir, dass die Effizienz des Perturbationwalkalgorithmus vom gewählten Perturbationgradpaar abhängt. Dazu stellen wir einige Tabellen vor. In diesen Tabellen sind die Anzahl der Schritte bzw. Berechnungszeiten des Pertubationwalkalgorithmus und die Große des Erzeugendensystems G_ω vom im letzten Schritt des Algorithmus aufgetretenen ω-homogenen Ideal gedruckt, falls jede Komponente der berechneten dazwischenliegenden Gewichtsvektoren kleiner als 2^{31} ist und der gestörte Zielgewichtsvektor im richtigen Gröbnerkegel liegt (siehe Bemerkungen für die anderen Fälle). Wir verwendeten in diesem Fall den Algorithmus 4.1.2 mit der Variante $4'$.

Falls nicht erwähnt, verwenden wir die folgenden Bezeichnungen:

1. † bedeutet, dass der gestörte Zielgewichtsvektor nicht im richtigen Gröbnerkegel liegt.

2. ‡ bedeutet, dass mehrere Komponenten des letzten dazwischenliegenden Gewichtsvektors größer als 2^{31} sind.

3. $x[y]$ steht für y Polynome mit jeweils x Monomen.

Bsp.	n	Perturbationgradpaar						
		(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
issac97	4	526.75	36.42	0.31	0.30†	—	—	—
ubikker	4	>6433.29	$>_z$10 M	7.29	7.18†	—	—	—
wang6	4	>8987.56	244.99	0.48	0.35†	—	—	—
casmod1	4	>17459.65	550.74	4.38	10.89†	—	—	—
cyc5mod1	5	>2621.67	>1634.50	>3745.99	21.90	7.83†	—	—
noon5	5	>1354.00	159.26	31.79	46.19	$>_z$320 M‡	—	—
noon6	6	>28 M	>2778.33	3066.57	1776.37	$>_z$368 M	$>_z$75 M‡	—
cyc6	6	>51 M	>24 M	>35 M	54.86	2.64	2.67	—
kats6	6	>3364.77	>3238.38	>10619.42	4902.30	1.12	1.17†	—
s7	7	>141 M	>1166.89	>71 M	>1363.77	4.79	5.77‡	0.70‡
kats7	7	>2585.78	>1188.30	>3491.47	>5014.57	>11083.95	112.31‡	$>_z$21 M‡
redeco7	7	3.90	4.15	3.70	250.43	253.72	0.36	0.36†

Tabelle D.1: Berechnungszeiten des Perturbationwalkalgorithmus.

Bsp.	n	Perturbationgradpaar						
		(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
issac97	4	3	10	21	13†	—	—	—
ubikker	4	4	28	59	31†	—	—	—
wang6	4	4	16	37	15†	—	—	—
casmod1	4	4	15	31	12†	–	—	—
cyc5mod1	5	2	13	55	72	62†	—	—
noon5	5	7	57	147	195	176†	—	—
noon6	6	8	76	339	422	?	1‡	—
cyc6	6	2	9	69	124	141	135†	—
kats6	6	2	4	14	29	52	56†	—
s7	7	2	11	44	88	119	131‡	1‡
kats7	7	2	4	17	43	81	122‡	8‡
redeco7	7	2	3	8	20	22	47	30†

Tabelle D.2: Anzahl der Schritte des Perturbationwalkalgorithmus (? Der Zielgewichtsvektor ist noch nicht erreicht).

Bsp.	n	Perturbationgradpaar				
		(1,1)	(2,2)	(3,3)	(4,4)	(5,5)
issac97	4	16[5], 1, 17[6]	1[2], 17[6]	1[3], 17	†	—
ubikker	4	32[4], 1, 36[9] 37[4]	37[8], 1[2]	1[3], 36	†	— —
wang6	4	25[14], 8, 1	1[2], 25[7]	1[3], 25	†	—
casmod1	4	23[15], 1	23[6], 1[2]	1[3], 23	†	—
cyc5mod1	5	1, 6, 17, 14 16, 32[2], 30[2] 33, 42, 40, 45, 46[2], 58[5], 65[11]	35, 36, 39[3] 49[4], 1[2], 51, 65[11], 50[2]	2[2], 64[11], 1[3]	2, 29, 1[6]	†
noon5	5	4[6], 1, 49[5], 46, 47[4], 48[3], 63, 93[8], 89, 92[2], 165, 121, 175,51, 180,143[3], 142, 94,139, 187	4[3], 95[2], 118, 127, 109, 30[6], 149[2], 1[2], 91[2], 147, 148, 65, 152, 49[5], 31, 141, 135[2]	4[7], 78, 73, 76, 84, 53[2], 54[2], 57[2], 60[2], 7[2], 1[3]	42, 9[3], 34[3], 1[6], 18[2]	†
cyclic6	6	1, 9, 24[15], 25[16], 26[17], 27[20]	26[41], 2[4], 1[2]	24[29], 2[6], 1[4]	12, 16, 21[10], 22, 2[6], 1[7]	8, 2[5], 1[11]
kats6	6	1, 14[2], 15, 16[2], 19, 22, 25[3], 26[6], 29, 31[8], 32[6]	1[2], 22, 25[2], 26[7], 24, 31[4], 29, 32[5]	1[3], 31[4], 32[12]	1[4], 32[8]	1[5], 32
kats7	7	1, 18, 20, 21[2], 19, 22, 26, 31, 37, 35, 40, 41[4], 39, 42[5], 38, 44 49[2], 54[2], 53[2], 56[3], 57[10], 58, 61[2], 63[11], 64[7]	1[2], 32, 35, 40, 41[4], 37[2], 38[2], 42[5], 44, 48, 54[2], 56[2], 51, 57[3], 61[2], 58, 64[6], 63[6], 53[2]	57[13], 1[3], 60[2], 62[3], 63[11], 61[2], 64[6]	1[4], 64[28]	1[5], 64[11]
redeco7‡	7	2, 1, 7, 10[2], 11, 13[2], 16[2], 8, 14[2], 17[3], 21, 23[2], 20, 22, 19, 24, 26[3], 29, 31[2], 28, 30, 32, 33	2, 8, 10, 14[2], 16, 17[3], 21, 24[2], 1[2], 13, 19, 20, 26[3], 22, 29, 32, 28, 30, 31, 33[2],	2, 13, 24, 1[3], 26[2], 32[4], 28, 30, 29, 31, 33[2],	2, 1[4], 33[8]	1[5], 33[8]

Tabelle D.3: Anzahl der Monome des im letzten Schritt des Perturbationwalkalgorithmus ergebenden ω-homogenen Ideals (‡Für das Gradpaar $(6,6)$ ist die Anzahl der Monome: $1[6], 33$).

Bsp.	n	Perturbationgradpaar								Bmk.
		(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)	(8,8)	
schwarz8	8	2	11	54	139	194	233‡	1‡	1‡	#Schr.
		>258 M	>1057.27	>1125.74	>1821.35	>1372.00	7.84	$>_z$10 M	$>_z$16 M	Zeit
		1[8], 8[14], 7[18], 9[19], 6[8], 4, 12[30], 10[26], 13[26], 11[25], 14[25], 16[22], 15[24], 18[5], 19, 17[3]	8[5], 10[9], 11[12], 13[23], 9[4], 12[16], 14[19], 15[29], 7[2], 1[8], 17[8], 18[3], 19[3], 16[20]	16[41], 12[5], 13[8], 1[8], 8 15[37], 11[3], 14[17], 9[4], 17[14], 18[5]	13[3], 14[9], 15[25], 17[33], 16[5], 11[4], 1[8], 18[14], 19[6], 9	16[27], 11[3], 6, 15[5], 13[12], 2[5], 3[7], 10[2], 18, 12,1[12]	siehe Bmk.1a	siehe Bmk.1b	siehe Bmk.1b	#Mon.
redeco8	8	2	3	8	24	60	65	3‡	1‡	#Schr.
		>1390.53	>1079.80	>2304.63	>1185.39	>4469.76	>4422.99	>2723.56	> 4564.22	Zeit
		s. Bmk.2(a)i	s. Bmk.2(a)ii	s. Bmk.2(a)iii	s. Bmk.2(a)iv	2, 1[5], 65[11],	1[6], 65[11],	s. Bmk.2b	s. Bmk.2c	#Mon.
vsoro	8	2	7	30	54	80	136‡	1‡	1‡	#Schr.
		>82 M	>982.25	>2201.15	>2994.76	>4347.96	>761.65	$>_z$247 M	>5853.75	Zeit
		s. Bmk.4(a)i	s. Bmk.4(a)ii	s. Bmk.4(a)iii	s. Bmk.4(a)iv	s. Bmk.4(a)v	s. Bmk.4b	s. Bmk.4c	s. Bmk.4c	#Mon.
dessin2	10	4	5	7	9	23	36	50	86	#Schr.
		>5457.57	>2875.15	>2879.49	>2874.58	> 2879.36	> 2814.45,	>10415.66	721.52	Zeit
		2, 3, 4, 6, 11[2], 13, 17, 1, 16, 20, 24, 8, 27[2], 31[3], 33[3], 36[4], 38[4], 40[4], 41[3], 42[6], 43[9]	2, 3, 4, 1[2], 11, 13, 17, 20[2], 24[2], 27[3], 31[3], 33[3],36[4], 38[3], 40[3], 41[3], 42[6], 43[9]	2, 3, 1[3], 11, 13, 17, 14, 12, 20, 24, 27[2], 31[3], 33[3], 36[4], 38[3], 40[4], 41[3], 42[7], 43[9]	2, 1[4], 11, 13, 17 14, 12, 20, 24, 27[2], 31[3], 33[3], 36[3], 38[2], 40[2], 41[2], 42[7], 43[8]	2, 1[5], 13, 20, 17, 24, 27, 33[3], 31, 36[2], 38[3], 41[2],40[2], 42[3], 43[5]	2, 23, 26, 1[6], 31, 33[2], 36[2], 38[2], 40[3], 41, 42[4], 43[3]	2, 1[7], 43[9],	2, 1[8], 46	#Mon.
bmarkD1	12	2	2	2	6	13	21	31	46	#Schr.
		>4698.83	>3949.81	>3952.20	>3670.19	>3035.44	>5388.09	>11204.45	>13226.93	Zeit
		4[3], 1, 3[4], 8, 13[32], 6, 7, 9[2], 10, 25[45]	4[2], 1[2], 3[4], 6, 8, 13[32], 7, 19, 9[2], 10, 25[39],	1[3], 3[4], 6, 4, 8, 13[32], 7, 9, 9[2], 10, 25[39]	3[4], 4, 8, 1[4], 25[63], 14, 13[13]	3[3],1[5], 25[63]	3[3], 1[6], 25[38]	3[2],1[7], 25[37]	3[2], 7, 1[8], 10, 25[16]	#Mon.
chemkin	13	2	2	2	2	7	14	20	20	#Schr.
		>1387.905	>8416.50	>9195.89	>9196.37	>3771.89	>8885.39	>2397.48	>2336.74	Zeit
		6, 4[4], 2[2], 1, 9, 3, 10, 8[2], 18, 41[44]	6, 4[4], 2, 1[2], 9, 3, 10, 8[2], 18, 41[42]	6, 4[4], 1[3], 9, 3, 10, 8, 18, 27, 41[42]	6, 4[3], 1[4], 9, 3, 10, 8, 18, 27, 41[42]	6, 4[3], 1[5], 3, 41[37], 13[2],8	6, 4, 3, 13, 1[6], 41[33], 8	6, 4, 1[7], 41[32], 8, 22, 26	6, 1[8], 41[32], 8, 22, 26	#Mon.

Tabelle D.4: Anzahl der Schritte (#Schr.), Berechnungszeiten (Zeit) in Sekunden und Anzahl der Monome des im letzten Schritt des Perturbationwalkalgorithmus ergebenden ω-homogenen Ideals (#Mon.).

Bsp.	Perturbationgradpaar					
	(9,9)	(10,10)	(11,11)	(12,12)	(13,13)	
dessin2	45†	45†	—	—	—	#Schr.
	16.07	>21794.26	—	—	—	Zeit
	siehe Bmk. 3a	siehe Bmk. 3b	— —	— —	— —	#Mon.
bmarkD1	63	77‡	6‡	1‡	—	#Schr.
	>31054.94	2778.11	>5552.07	>4003.70	—	Zeit
	3, 25[13], 1[9]	s. Bmk. 5a	s. Bmk. 5b	s. Bmk. 5c	—	
chemkin	20	35	15‡	4‡	1‡	#Schr.
	>2429.15	>10258.64	>12029.85	>121 M	>134 M	Zeit
	1[9], 41[32], 8, 22, 26	1[10], 41[21]	siehe Bmk. 6a	siehe Bmk. 6b	siehe Bmk. 6c	#Mon.

Tabelle D.5: Anzahl der Schritte (#Schr.), Berechnungszeiten (Zeit) in Sekunden und Anzahl der Monome des im letzten Schritt des Perturbationwalkalgorithmus ergebenden ω-homogenen Ideals (#Mon.).

Bemerkungen:

1. Beispiel **schwarz8**:

   (a) Im 233-ten Schritt des Perturbationwalkalgorithmus vom Perturbationgradpaar $(6,6)$ erhält man eine reduzierte Gröbnerbasis F des gegebenen Ideals I. Die Anzahl der Monome von F ist:

   $$11[4], 15[10], 13[3], 16[9], 10, 18[3], 17[3], 14, 19[2], 20.$$

   (b) Im ersten Schritt des Perturbationwalkalgorithmus vom Perturbationgradpaar $(7,7)$ bzw. $(8,8)$ ergibt sich eine reduzierte Gröbnerbasis F des gegebenen Ideals I. Die Anzahl der Monome von F ist:

   $$5[4], 6[4], 8, 7[2], 10[2], 12[8], 11[12], 9[2], 15[8], 16[10], 14[8], 18[18], 13[6], 19[13], 17[10], 20[6], 21[4], 22.$$

2. Beispiel **redeco8**:

   (a) Die Anzahl der Monome eines Erzeugendensystems eines ω-homogenen Ideals, das im letzten Schritt des Perturbationwalkalgorithmus von den folgenden Perturbationgradpaare auftritt, ist wie folgt:

      i. Perturbationgradpaar $(1,1)$:

      $$2, 1, 8, 12[2], 14, 15, 18[3], 22[3], 9, 16, 13, 21[2], 19, 25, 26[4], 31, 35, 38[3], 11, 27, 28, 33[2], 30, 36, 34, 41, 39[3], 46, 47[3], 50[2], 54[2], 32, 42, 45, 53[2], 55, 58, 57, 61, 63[2], 56, 60, 62, 64, 65.$$

      ii. Perturbationgradpaar $(2,2)$:

      $$2, 9, 12, 13, 19, 22[2], 16, 21[2], 25, 26[4], 31, 35, 39[4], 1[2], 11, 18[2], 28, 30, 34, 27, 33[2], 36, 41, 46, 47[3], 50[2], 53[3], 58[2], 32, 38, 45, 42, 55, 57, 61, 62[2], 56, 60, 63, 65[2].$$

iii. Perturbationgradpaar $(3,3)$:

$$2, 11, 18[2], 22, 26[2], 39, 43[2], 29, 41, 44[3], 1[3], 58[3], 32, 38, 35,$$
$$50, 45, 51, 55[2], 46, 53[2], 61, 62[2], 63, 64[2], 57, 60, 65[2].$$

iv. Perturbationgradpaar $(4,4)$:

$$2, 25, 30, 37, 56[2], 54, 1[4], 64, 65[18], 59.$$

(b) Im dritten Schritt des Perturbationwalkalgorithmus vom Perturbationgradpaar $(7,7)$ ergibt sich eine reduzierte Gröbnerbasis des gegebenen Ideals mit der Anzahl ihrer Monome:

$$2, 9, 12, 13, 19, 16, 21[2], 11, 25, 26[4], 18[2], 22[2], 28, 30, 34, 27, 39[3],$$
$$33[2], 36, 41, 46, 31, 47[3], 35, 50[2], 53[3], 55[2], 32, 38, 54, 45, 42, 58[2],$$
$$57[2], 61, 63[2], 56, 60, 62, 65[2].$$

In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines Ideals durch den Buchbergeralgorithmus ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist:

$$2, 11, 18[2], 22, 28, 29, 26[2], 35[2], 33[2], 40[3], 32, 39, 38, 43[2], 47[3], 46,$$
$$50[2], 55, 53[2], 58[2], 52, 61, 56, 64[3], 60, 36, 62, 65[2], 1[2]$$

(c) Im ersten Schritt des Perturbationwalkalgorithmus vom Perturbationgradpaar $(8,8)$ ergibt sich eine reduzierte Gröbnerbasis des gegebenen Ideals mit der Anzahl ihrer Monome:

$$2, 8[2], 12[2], 14, 15, 18[3], 22[3], 9, 13, 19, 16, 21[2], 25, 31, 26[4], 35, 38[3],$$
$$11, 28, 34, 30, 39[3], 27, 33[2], 41, 36, 46, 50[2], 47[3], 54[2], 32, 45, 42, 55, 57,$$
$$53[2], 58, 61, 63[2], 56, 60, 62, 65, 64,$$

In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines Ideals durch den Buchbergeralgorithmus ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist $2, 65[11], 1[5]$.

3. Beispiel **dessin2**:

(a) Am Ende des 45-ten Schritts des Perturbationwalkalgorithmus vom Perturbationgradpaar $(9,9)$ ergibt sich eine reduzierte Gröbnerbasis F des gegebenen Ideals I. Die Anzahl der Monome von F ist

$$2, 43[13], 3, 4, 6.$$

Da der gestörte Zielgewichtsvektor nicht im richtigen Gröbnerkegel liegt, muss die reduzierte lexikographische Gröbnerbasis von $\langle F \rangle$ ausgerechnet werden. Insgesamt benötigt der Perturbationwalkalgorithmus 16.07 Sekunden.

(b) Am Ende des 45-ten Schritts des Perturbationwalkalgorithmus vom Perturbationgradpaar $(10,10)$ ergibt sich eine reduzierte Gröbnerbasis F des gegebenen Ideals I. Die Anzahl der Monome von F ist

$$2, 43[13], 3, 4, 6.$$

Wie mit dem Perturbationgradpaar $(9,9)$ liegt der gestörte Zielgewichtsvektor nicht im richtigen Gröbnerkegel. In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals durch den Buchbergeralgorithmus ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist $2,43[9]$ bzw. $1[6]$.

4. Beispiel **Virasoro**:

   (a) Die Anzahl der Monome eines Erzeugendensystems vom letzten ω-homogenen Ideal im Perturbationwalkalgorithmus vom:

      i. Perturbationgradpaar $(1,1)$ ist:

      $8[4], 18[2], 21, 2[3], 1, 31, 41, 52, 53[2], 59[2], 50, 66, 65, 69[2], 5, 14, 33, 64, 62, 78[2], 86, 96[7], 95[4], 92[2], 112[3], 84[2], 110[5], 111, 91[2], 104[2], 94[6], 126[5], 119, 9[3], 10[3], 12, 83[2], 97[3], 98[3], 99[2], 124[3], 125, 131, 132[2], 134[2], 133[2], 140[3], 153[2], 150, 143[2], 136[2], 114, 118, 145[4], 174, 11, 93[2], 102[2], 109[6], 100, 103, 108[7], 146[2], 135, 154, 169, 142, 107[4], 115[2], 137, 155, 128.$

      ii. Perturbationgradpaar $(2,2)$ ist:

      $8[5], 18[2], 4, 1[3], 31, 41, 52, 53, 10[6], 7, 9[4], 12[2], 85, 91[4], 96[3], 88, 98[2], 100[2], 101[3], 115, 104, 116[5], 13, 15, 19[3], 21, 17[2], 129[4], 133[2], 131, 138[4], 139[2], 156, 157[2], 159[4], 158, 140[5], 151[2], 153[3], 2[2], 20[2], 135[5], 126[2], 134, 145, 148, 164[2], 147, 170[3], 160, 154[2], 175[2], 5, 124, 92, 137[5], 166, 141[2], 142, 168, 11[3], 136[2].$

      iii. Perturbationgradpaar $(3,3)$ ist:

      $8[2], 18[2], 31[2], 19, 16[5], 6, 53, 57, 66, 63, 22, 24, 23[2], 25, 27, 15[2], 2[11], 1[6], 4[8], 80, 97[2], 82[3], 91, 21[4], 84, 102, 100[3], 33[6], 37, 34, 17[4], 20[3], 10[3], 7, 96, 76, 103[2], 77, 83, 105[7], 108, 106[3], 113, 32[2], 115, 112, 92, 109[4], 13, 64, 107, 94[2], 95[2], 110, 116, 35[2], 39, 14, 12, 11, 5, 9[3].$

      iv. Perturbationgradpaar $(4,4)$ ist:

      $14[2], 33[2], 53, 2[11], 77, 96[2], 98[3], 97, 99, 4[10], 73[2], 103[7], 101, 107[4], 106[3], 105[2], 104[3], 109[4], 110[3], 111[3], 108[4], 1[10], 80, 114, 115, 12[3], 27, 18[3], 29, 28, 32[3], 36[3], 37[2], 35[6], 34[2], 38, 10[5], 21[3], 19, 24, 22, 23, 7, 13, 15, 16[4], 17[2], 11, 5, 8, 9[3].$

      v. Perturbationgradpaar $(5,5)$ ist:

      $14[3], 58, 60[4], 61, 53[2], 57, 70[4], 71[4], 72[4], 73[2], 27, 22[2], 28, 42[5], 43[4], 4[11], 26, 38, 39, 2[8], 44, 1[16], 12[6], 21[2], 20, 18[5], 29[2], 30[2], 8[3], 9[4], 11, 10[6], 17[4], 16[2], 7[2], 6[3], 5.$

(b) Im 136-ten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(6,6)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist:

$$71[4], 68[7], 67, 69[4], 72[2], 73, 92[2], 95[2], 106[2], 91, 105, 89[2], 110[5],$$
$$111[3], 113, 97[2], 112, 116, 114[2], 78[2], 82, 8, 18, 86, 93, 94[3], 98[2], 100,$$
$$138[2], 132, 141[2], 146[2], 148, 109[5], 107, 108, 104, 159[5], 158, 163[2],$$
$$157, 142[3], 144[2], 136, 152, 151, 143, 161, 180, 177, 183, 181, 185[2], 153,$$
$$155, 160, 188, 190, 186, 173, 99.$$

Im ersten Schritt jeder Schleife der Variante $4'$ sind mehrere Komponenten des dazwischenliegenden Gewichtsvektors größer als 2^{31}. Deswegen wird der Buchbergeralgorithmus auf $\langle F \rangle$ im letzten Schritt der Variante $4'$ durchgeführt.

(c) Im ersten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(7,7)$ bzw. $(8,8)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist:

$$8, 18[3], 17, 13[2], 22, 19[2], 64[2], 65[2], 62, 73[3], 27, 71[2], 80, 82, 114,$$
$$115[2], 117[2], 127[5], 129[3], 130[2], 128, 131, 148[2], 118, 149[3], 151, 155[4],$$
$$165[6], 168[5], 194, 173, 179[10], 196, 156[2], 166[4], 167[5], 170[4], 195[2],$$
$$174[5], 180, 172[2], 182[4], 204, 186[2], 169, 205, 187, 206[2], 171[2], 181[8],$$
$$177, 184[3], 210, 175.$$

5. Beispiel **benchmarkD1**:

(a) Im 77-ten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(10,10)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist 3 bzw. 25[14]. In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist $3, 25[6]$ bzw. $1[8]$.

(b) Im sechsten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(11,11)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist $3[4], 4[4], 25[76], 8, 6, 7$ bzw. 10. In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist $3[3], 25[48], 8$ bzw. $1[5]$.

(c) Im ersten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(12,12)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist

$$4[4], 3[4], 6, 7, 9[2], 8, 13[32], 10, 25[39], 19.$$

In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist

$$3[4], 4, 8, 13[16], 25[63], 6, 7, 1[3], 10.$$

.

6. Beispiel **chemkin**:

   (a) Im 15-ten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(11, 11)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist:

   $$6, 4[3], 41[41], 3, 8, 9, 7, 2[3].$$

   In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist $6, 4, 1[6], 41[39], 3$ bzw. 8.

   (b) Im vierten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(12, 12)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist:

   $$6, 4[4], 9, 3, 41[39], 10, 8[2], 18, 2[3].$$

   In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist

   $$6, 4[3], 1[5], 3, 41[39], 13, 8.$$

   (c) Im ersten Schritt des Perturbationwalkalgorithmus vom Gradpaar $(13, 13)$ ergibt sich eine reduzierte Gräbnerbasis F des gegebenen Ideals. Die Anzahl der Monome von F ist:

   $$6, 4[4], 9, 2[3], 3, 10, 8, 18, 41[43].$$

   In der Variante $4'$ wird eine reduzierte Gröbnerbasis eines ω-homogenen Ideals ausgerechnet. Die Anzahl der Monome des Erzeugendensystems dieses Ideals ist

   $$6, 4[3], 9, 1[4], 3, 10, 8, 18, 41[43].$$

Anhang E

Gröbnerbasen im Tranalgorithmus

Im Tranalgorithmus wird der aktuelle Zielgewichtsvektor durch die Repräsentation des ursprünglichen Zielgewichtsvektors ersetzt, wenn der berechnete dazwischenliegende Gewichtsvektor gleich dem aktuellen Zielgewichtsvektor und die bereits berechnete Gröbnerbasis noch nicht die gesuchte Gröbnerbasis ist. In großen Beispielen, in denen die Anzahl der Variablen und der maximale Totalgrad einer Gröbnerbasis groß ist, besitzt die Repräsentation einen oder mehrere große Einträge. Aus diesem Grund kann es passieren, dass eine oder mehrere Komponenten eines der berechneten dazwischenliegenden Gewichtsvektoren oder dieser Repräsentation größer als 2^{31} sind. An dieser Stelle erhält man nur die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. einer monomialen Ordnung, die sich von der gegebenen Zielordnung $\gg$ unterscheidet. Deswegen muss die reduzierte Gröbnerbasis von $\langle F \rangle$ bzgl. $\gg$ ausgerechnet werden. Dazu wenden wir den Unteralgorithmus 4.4.2 vom Grad $n-1$ auf $\langle F \rangle$ an. In diesem Unteralgorithmus benötigt die Berechnung der reduzierten Gröbnerbasis des im letzten Schritt aufgetretenen ω-homogenen Ideals $\langle G_\omega \rangle$ bzgl der ω-gewichteten lexikographischen Ordnung $\gg_\omega$ leider noch viel Zeit und Speicher, obwohl G_ω „kleiner“ als F ist (vgl. Tabelle 5.14).

In diesem Anhang zeigen wir nun die Größe und die Initialterme von F und G_ω bzgl. der entsprechenden Ordnung mehrerer Beispiele.

1. **Beispiel katsura7 (kats7)** mit $x_0 > x_1 > x_2 > x_3 > x_4 > x_5 > x_6$:

- Im 84-ten Schritt, in dem mehrere Komponenten eines dazwischenliegenden Gewichtsvektors größer als 2^{31} sind, erhält man die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. $\gg_{\omega_{83}}$ mit

$$\omega_{83} = (641437007, 32239503, 1639919, 86655, 5111, 387, 43).$$

  F besteht aus 10 Polynomen. Ein Polynom davon hat 65 Monome und jedes der anderen hat 64 Monome. Die Menge der Leitterme $F_{\gg_{\omega_{83}}}$ von F bzgl. dieser Ordnung ist:

$$\{\underbrace{7722917\ldots2461824}_{769\text{ Ziffern}}x_6^2x_7^{11},\underbrace{2145254\ldots568384}_{768\text{ Ziffern}}x_6^3x_7^2,\underbrace{1130594\ldots642112}_{815\text{ Ziffern}}x_7^{30},\underbrace{2987820\ldots995584}_{787\text{ Ziffern}}x_6x_7^{21},$$
$$\underbrace{1029722\ldots282432}_{770\text{ Ziffern}}x_6^4,\underbrace{1668924\ldots233950}_{778\text{ Ziffern}}x_5,\underbrace{3259906\ldots813630}_{774\text{ Ziffern}}x_4,\underbrace{3671633\ldots146900}_{779\text{ Ziffern}}x_3,$$
$$\underbrace{1810058\ldots920740}_{769\text{ Ziffern}}x_2,\underbrace{3059694\ldots595575}_{778\text{ Ziffern}}x_1\}.$$

- Im Unteralgorithmus 4.4.2 vom Grad 5 wird die reduzierte Gröbnerbasis von $\langle G_\omega\rangle$ bzgl. $\gg_\omega$ mit $\omega = (331776, 13824, 576, 24, 1, 0, 0)$ ausgerechnet. G_ω besteht aus fünf Monomen und fünf Polynomen mit 64 Monomen. Die Menge der Leitterme von G_ω bzgl. dieser Ordnung stimmt mit $F_{\gg_{\omega_{83}}}$ überein.

2. **Beispiel redeco8** mit $x_1 > x_2 > x_3 > x_4 > x_5 > u_8 > x_6 > x_7$:

- Im 94-ten Schritt, in dem mehrere Komponenten eines dazwischenliegenden Gewichtsvektors größer als 2^{31} sind, erhält man die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. $\gg_{\omega_{93}}$ mit

$$\omega_{93} = (906422928, 43862838, 2205348, 124458, 9168, 978, 138, 23).$$

F besteht aus einem Monom und 10 Polynomen mit 65 Monomen. Die Menge der Leitterme $F_{\gg_{\omega_{93}}}$ von F bzgl. dieser Ordnung ist:

$$\{\underbrace{130099\ldots252736}_{186\text{ Ziffern}}x_6^4,\underbrace{667568\ldots186511}_{187\text{ Ziffern}}x_7^{25},\underbrace{667290\ldots082664}_{185\text{ Ziffern}}x_6x_7^{19},\underbrace{127050\ldots932864}_{183\text{ Ziffern}}x_6^2x_7^{13},$$
$$\underbrace{120949\ldots884032}_{180\text{ Ziffern}}x_6^3x_7^7,7u_8,\underbrace{999578\ldots212352}_{190\text{ Ziffern}}x_5,\underbrace{139941\ldots972928}_{192\text{ Ziffern}}x_4,\underbrace{666385\ldots141568}_{190\text{ Ziffern}}x_3,$$
$$\underbrace{233235\ldots495488}_{191\text{ Ziffern}}x_2,\underbrace{317757\ldots622984}_{184\text{ Ziffern}}x_1\}.$$

- Im Unteralgorithmus 4.4.2 vom Grad 6 wird die reduzierte Gröbnerbasis von $\langle G_\omega\rangle$ bzgl. $\gg_\omega$ mit $\omega = (4084101, 194481, 9261, 441, 21, 1, 0, 0)$ ausgerechnet. G_ω besteht aus sechs Monomen und fünf Polynomen mit 65 Monomen. Die Menge der Leitterme von G_ω bzgl. dieser Ordnung stimmt mit $F_{\gg_{\omega_{93}}}$ überein.

3. **Beispiel dessin2** mit $a > b > c > d > u > v > w > x > y > z$:

- Im 40-ten Schritt, in dem mehrere Komponenten der zweiten Repräsentation des ursprünglichen Zielgewichtsvektors größer als 2^{31} sind, erhält man die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. $\gg_{\omega_{39}}$ mit

$$\omega_{39} = (524290, 131073, 32769, 8193, 2049, 513, 129, 33, 9, 3).$$

F besteht aus 16 Polynomen. 12 Polynome davon hat 43 Monome und die anderen Polynome bestehen aus 2, 3, 4 bzw. 6 Monomen. Die Menge der Leitterme $F_{\gg_{\omega_{39}}}$ von F bzgl. dieser Ordnung ist:

$$\{z,\underbrace{3669209\ldots4471424}_{418\text{ Ziffern}}x^3y^4,\underbrace{2476716\ldots2682112}_{419\text{ Ziffern}}x^2y^8,\underbrace{5337359\ldots2589568}_{430\text{ Ziffern}}wy,$$
$$\underbrace{5687407\ldots0678912}_{424\text{ Ziffern}}xy^{12},\underbrace{5283661\ldots3885056}_{420\text{ Ziffern}}x^4y,\underbrace{3085939\ldots98108928}_{464\text{ Ziffern}}y^{16},$$
$$\underbrace{4447799\ldots64382464}_{429\text{ Ziffern}}wx,\underbrace{2113464\ldots55540224}_{421\text{ Ziffern}}x^5,\underbrace{8339624\ldots99571712}_{428\text{ Ziffern}}w^2,$$
$$\underbrace{5764348\ldots39673344}_{432\text{ Ziffern}}v,\underbrace{4803623\ldots53306112}_{431\text{ Ziffern}}u,16d,12c,192b,\underbrace{192144\ldots13224448}_{432\text{ Ziffern}}a\}.$$

- Im Unteralgorithmus 4.4.2 vom Grad 7 wird die reduzierte Gröbnerbasis von $\langle G_\omega \rangle$ bzgl. $\gg_\omega$ mit $\omega = (4826809, 371293, 28561, 2197, 169, 13, 1, 0, 0, 0)$ ausgerechnet. G_ω besteht aus sieben Monomen und sieben Polynomen. Ein Polynom davon hat zwei Monome und jedes der anderen hat 43 Monome. Die Menge der Leitterme von G_ω bzgl. dieser Ordnung ist:

$$\{z, \underbrace{544921\ldots000000}_{417\text{ Ziffern}} w, \underbrace{158565\ldots135360}_{399\text{ Ziffern}} x^2y^8, \underbrace{509770\ldots175104}_{403\text{ Ziffern}} xy^{12}, \underbrace{284149\ldots256512}_{401\text{ Ziffern}} x^4y,$$
$$\underbrace{138298\ldots850880}_{445\text{ Ziffern}} y^{16}, \underbrace{164438\ldots807040}_{399\text{ Ziffern}} x^3y^5, \underbrace{189432\ldots710080}_{402\text{ Ziffern}} x^5, \underbrace{544921\ldots000000}_{417\text{ Ziffern}} v,$$
$$\underbrace{336371\ldots000000}_{415\text{ Ziffern}} u, 16d, 12c, \underbrace{622767\ldots000000}_{417\text{ Ziffern}} b, \underbrace{322916\ldots2800000}_{416\text{ Ziffern}} a\}.$$

4. **Beispiel benchmarkD1** mit

$$x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > x_7 > x_8 > x_9 > x_{10} > x_{11} > x_{12} :$$

- Im 71-ten Schritt, in dem mehrere Komponenten der zweiten Repräsentation des ursprünglichen Zielgewichtsvektors größer als 2^{31} sind, erhält man die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. $\gg_{\omega_{70}}$ mit

$$\omega_{70} = (1062883, 354295, 118099, 39367, 13123, 4375, 1459, 487, 163, 55, 19, 7).$$

  F besteht aus 17 Polynomen. Ein Polynom davon hat drei Monome und jedes der anderen hat 25 Monome. Die Menge der Leitterme $F_{\gg_{\omega_{70}}}$ von F bzgl. dieser Ordnung ist:

$$\{x_{11}^2, \underbrace{2826170974\ldots12215840}_{3543\text{ Ziffern}} x_{10}x_{11}x_{12}^6, \underbrace{50376778\ldots080000}_{3302\text{ Ziffern}} x_{10}^2x_{12}, \underbrace{38111270\ldots408000}_{3613\text{ Ziffern}} x_{10}x_{12}^9,$$
$$\underbrace{49572796\ldots350000}_{3762\text{ Ziffern}} x_{12}^{17}, \underbrace{1347645909\ldots82550000}_{3532\text{ Ziffern}} x_{11}x_{12}^{15}, \underbrace{4752526243\ldots83823680}_{3296\text{ Ziffern}} x_{10}^2x_{11},$$
$$\underbrace{3205125983\ldots21452800}_{3550\text{ Ziffern}} x_9, \underbrace{6416243749\ldots10534400}_{3400\text{ Ziffern}} x_{10}^3, \underbrace{8493581914\ldots00000000}_{3305\text{ Ziffern}} x_8,$$
$$\underbrace{4886194706\ldots48000000}_{3653\text{ Ziffern}} x_7, \underbrace{7077984928\ldots00000000}_{3305\text{ Ziffern}} x_6, \underbrace{1221548676\ldots12000000}_{3653\text{ Ziffern}} x_5,$$
$$\underbrace{4246790957\ldots00000000}_{3306\text{ Ziffern}} x_4, \underbrace{4886194706\ldots48000000}_{3653\text{ Ziffern}} x_3, \underbrace{8463918370\ldots80000000}_{3210\text{ Ziffern}} x_2,$$
$$\underbrace{3943125259\ldots20000000}_{3207\text{ Ziffern}} x_1\}.$$

- Im Unteralgorithmus 4.4.2 vom Grad 8 wird die reduzierte Gröbnerbasis von $\langle G_\omega \rangle$ bzgl. $\gg_\omega$ mit

$$\omega = (170859375, 11390625, 759375, 50625, 3375, 225, 15, 1, 0, 0, 0, 0)$$

  ausgerechnet. G_ω besteht aus acht Monomen und neun Polynomen. Ein Polynom davon hat drei Monome und jedes der anderen hat 25 Monome. Die Menge der Leitterme von G_ω bzgl. dieser Ordnung stimmt mit $F_{\gg_{\omega_{70}}}$ überein.

5. **Beispiel chemkin** mit

$$c_1 > c_2 > c_3 > z_2 > z_3 > z_4 > z_5 > y_2 > y_3 > y_4 > y_5 > x_3 > x_4 :$$

- Im 40-ten Schritt, in dem mehrere Komponenten der zweiten Repräsentation des ursprünglichen Zielgewichtsvektors größer als 2^{31} sind, erhält man die reduzierte Gröbnerbasis F des gegebenen Ideals bzgl. $\gg_{\omega_{39}}$ mit

  $$\omega_{39} = (2391485, 797162, 265721, 88574, 29525, 9842, 3281, 1094, 365, 122, 41, 14, 5).$$

  F besteht aus 20 Polynomen. 13 Polynome davon hat 41 Monome, drei Polynom haben zwei Monome und die anderen Polynome bestehen aus 4, 7, 16 bzw. 17 Monomen. Die Menge der Leitterme $F_{\gg_{\omega_{39}}}$ von F bzgl. dieser Ordnung ist:

  $\{\underbrace{1287846\ldots733280}_{1791\text{ Ziffern}} y_5x_3x_4, \underbrace{1759201\ldots181845}_{1766\text{ Ziffern}} x_3^4x_4, \underbrace{6602434\ldots887420}_{1766\text{ Ziffern}} y_5x_4^4, \underbrace{3594482\ldots923130}_{1780\text{ Ziffern}} x_3^3x_4^4,$
  $\underbrace{1651963\ldots321266}_{1776\text{ Ziffern}} x_3^2x_4^7, \underbrace{8658635\ldots412480}_{1811\text{ Ziffern}} x_3x_4^{10}, \underbrace{3865336\ldots116672}_{1975\text{ Ziffern}} 0x_4^{13}, \underbrace{3863540\ldots519984}_{1790\text{ Ziffern}} y_5x_3^2,$
  $\underbrace{3008234\ldots009549}_{1768\text{ Ziffern}} 5x_3^5, \underbrace{1219787\ldots078400}_{1917\text{ Ziffern}} y_5^2, 3938244442785973241431437977856y_4,$
  $3938244442785973241431437977856y_3, 283926y_2, \underbrace{3622800\ldots886080}_{1871\text{ Ziffern}} z_5, \underbrace{4248963\ldots364096}_{1869\text{ Ziffern}} z_4,$
  $\underbrace{6883320\ldots835520}_{1872\text{ Ziffern}} z_3, 496125264095796z_2, 33461c_3, 9c_2, 36926c_1\}.$

- Im Unteralgorithmus 4.4.2 vom Grad 9 wird die reduzierte Gröbnerbasis von $\langle G_\omega \rangle$ bzgl. $\gg_\omega$ mit

  $$\omega = (214358881, 19487171, 1771561, 151, 14641, 1331, 121, 11, 1, 0, 0, 0, 0)$$

  ausgerechnet. G_ω besteht aus neun Monomen und 11 Polynomen. Ein Polynom davon hat 17 Monome und jedes der anderen hat 41 Monome. Die Menge der Leitterme von G_ω bzgl. dieser Ordnung stimmt mit $F_{\gg_{\omega_{39}}}$ überein.

Anhang F

Zeitanalysen

In diesem Anhang präsentieren wir Laufzeitanalysen von einigen Algorithmen für ausgewählte Beispiele. In jeder Tabelle sind die Anzahl der Schritte, der maximale Speicherverbrauch in Kilobyte bzw. Berechnungszeiten in Sekunden eines Algorithmus und auch Prozente des Zeitverbrauchs jeder aufgerufenen Hauptprozedur abgedruckt, nämlich *InitialForm*, *ReduziertGB*, *LiftGB*, *Reduzierung* und *Nächster-Vektor*. In der Spalte *inp* sind Zeiten (in Sekunden bzw. Prozenten) zur Berechnung der reduzierten umgekehrten lexikographischen Gröbnerbasis des gegebenen Ideals abgedruckt. Prozente des Zeitverbrauchs der anderen Prozeduren sind in der Spalte *Rest* gegeben, etwa zur Kopie eines Ideals und zum Test, ob die bereits berechnete Gröbnerbasis die gesuchte Gröbnerbasis ist.

Tabellen 5.6 bzw. 5.7 zeigen, dass im Gröbnerwalkalgorithmus die meiste Zeit zur Berechnung der reduzierten lexikographischen Gröbnerbasis eines ω-homogenen Ideals gebraucht wird. Diese Tabellen zeigen auch, dass im neuen Gröbnerwalkalgorithmus die meiste Zeit auf die im letzten Schritt ausgeführte Prozedur *LiftGB* entfällt. Deswegen geben wir nicht Laufzeitanalysen dieser beiden Algorithmen an.

Bsp.	n	Speicher	B.-Zeit (Sek.)		Berechnungszeiten (%)							
			inp.	total	inp.	InFo	sG_ω	Lift	InRd	NäVe	Summe	Rest
qnt2	3	7549	0.00	12.44	0.00	0.08	0.88	10.13	87.86	0.32	99.28	0.72
qnt3	3	762	0.01	0.05	20.00	0.00	0.00	0.00	80.00	0.00	100.00	0.00
tran1	3	1787	0.00	0.09	0.00	0.00	0.00	0.00	66.67	11.11	77.78	22.22
tran2	3	2300	0.01	0.92	1.09	0.00	2.17	9.78	84.78	1.09	98.91	1.09
ex1	3	1275	0.00	0.13	0.00	0.00	7.69	15.38	61.54	7.69	92.31	7.69
ex2	3	1828	0.02	0.82	2.44	0.00	2.44	13.41	78.05	0.00	96.34	3.66
ex3	3	2348	0.04	0.68	5.88	2.94	0.00	13.24	66.18	2.94	91.18	8.82
exam7	3	1896	0.03	2.45	1.22	0.00	0.41	11.84	84.49	0.82	98.78	1.22
exam12	3	1275	0.01	0.30	3.33	0.00	0.00	10.00	83.33	0.00	96.67	3.33
rose	3	1787	0.02	1.35	1.48	2.22	4.44	6.67	81.48	0.74	97.04	2.96
nmann1	3	2812	0.00	0.94	0.00	4.26	1.06	17.02	73.40	2.13	97.87	2.13
lblau	3	29953	3.34	726.56	0.46	0.01	0.11	6.55	92.79	0.00	99.91	0.09
issac97	4	12828	0.05	190.82	0.03	0.00	0.03	87.08	12.86	0.01	99.99	0.01
wang6	4	7991	0.10	106.67	0.09	0.01	0.04	99.55	0.30	0.00	99.99	0.0
cassou	4	780	0.25	0.28	89.29	0.00	3.57	7.14	0.00	0.00	100.00	0.00
casmod1	4	34569	2.56	1956.26	0.13	0.00	0.01	99.76	0.10	0.00	100.00	0.00
exam58	4	1897	0.06	1.76	3.41	0.57	1.14	84.09	9.09	1.14	99.43	0.57
exam59	4	13573	0.04	209.24	0.02	0.00	0.03	86.59	13.34	0.00	99.99	0.01
exam46	4	2302	0.00	0.27	0.00	0.00	3.70	48.15	44.44	0.00	96.30	3.70
exam14	4	4372	0.01	16.16	0.06	0.00	0.00	98.64	1.18	0.00	99.88	0.12
noon4	4	2814	0.01	0.42	2.38	0.00	2.38	26.19	57.14	0.00	88.10	11.90
vermeer76	5	764	0.02	0.06	33.33	0.00	0.00	0.00	0.00	0.00	33.33	66.67
cyclic5	5	1285	0.02	0.05	40.00	0.00	0.00	40.00	20.00	0.00	100.00	0.00
kats5	5	1325	0.01	0.49	2.04	0.00	0.00	97.96	0.00	0.00	100.00	0.00
noon5	5	28167	0.04	80.36	0.05	0.14	0.16	76.29	22.80	0.12	99.56	0.44
trinks1	6	1349	0.02	1.23	1.63	0.00	0.00	95.12	2.44	0.00	99.19	0.81
jhd6inf	6	764	0.16	0.17	94.12	0.00	0.00	5.88	0.00	0.00	100.00	0.00
eco6	6	1277	0.01	0.15	6.67	0.00	0.00	80.00	13.33	0.00	100.00	0.00
kats6	6	93527	0.26	1693.62	0.02	0.00	0.00	99.90	0.07	0.00	99.99	0.01
s6	6	1277	0.01	0.07	14.29	14.29	14.29	42.86	0.00	0.00	85.71	14.29
MK5	6	119412	0.31	408.02	0.08	0.00	0.00	99.27	0.62	0.00	99.97	0.03
s7	7	106816	0.06	496.20	0.01	0.00	0.78	99.17	0.02	0.00	99.99	0.01
eco7	7	67774	0.06	361.96	0.02	0.00	0.01	99.90	0.05	0.00	99.98	0.02
redeco7	7	9501	0.02	107.83	0.02	0.02	0.04	99.74	0.15	0.01	99.97	0.03

Tabelle F.1: Anzahl der Schritte, maximaler Speicherverbrauch und Berechnungszeiten des Fractalwalkalgorithmus.

Bsp.	n	Schritte	Speicher	B.-Zeit (Sek.)		Berechnungszeiten (%)							
				inp.	total	inp.	InFo	sG_ω	Lift	InRd	NäVe	Summe	Rest
qnt2	3	41 (15,26)	6010	0.00	10.46	0.00	0.00	0.96	9.27	89.29	0.29	99.81	0.19
qnt3	3	13 (5,8)	765	0.01	0.04	25.00	0.00	0.00	0.00	75.00	0.00	100.00	0.00
tran1	3	17 (5, 12)	1277	0.00	0.12	0.00	16.67	0.00	16.67	50.00	0.00	83.33	16.67
tran2	3	64 (21, 43)	1790	0.02	1.24	1.61	1.61	1.61	9.68	82.26	0.81	97.58	2.42
ex1	3	35 (12,23)	765	0.00	0.08	0.00	0.00	0.00	12.50	75.00	0.00	87.50	12.50
ex2	3	51 (20,31)	1830	0.02	0.84	2.38	0.00	1.19	9.52	83.33	1.19	97.62	2.38
ex3	3	58 (25,33)	2346	0.02	0.50	4.00	2.00	0.00	6.00	78.00	2.00	92.00	8.00
exam7	3	47 (11,34)	1896	0.03	2.62	1.15	0.38	0.76	9.54	86.64	0.00	98.47	1.53
exam12	3	28 (9,17)	762	0.01	0.30	3.33	6.67	6.67	10.00	63.33	3.33	93.33	6.67
rose	3	79 (0,79)	2300	0.02	1.80	1.11	1.11	0.56	7.22	85.00	2.22	97.22	2.78
nmann1	3	76 (24,52)	2300	0.00	0.66	0.00	0.00	4.55	10.61	77.27	3.03	95.45	4.55
lichtblau	3	48 (24,24)	67207	1.31	621.85	0.21	0.01	0.11	5.12	94.47	0.01	99.93	0.07
m51330	3	169 (21,98,50)	16926	0.48	554.48	0.09	0.02	0.76	12.60	86.41	0.01	99.88	0.12
m52470	3	136 (16,83,37)	17743	0.36	316.63	0.11	0.03	0.66	8.40	90.57	0.03	99.81	0.19
issac97	4	16 (2,8,3,2,1)	762	0.02	0.30	6.67	0.00	10.00	10.00	26.67	0.00	53.33	46.67
wang6	4	28 (4,15,5,3,1)	1275	0.04	0.55	7.27	1.82	12.73	10.91	30.91	1.82	65.45	34.55
casmod1	4	26 (3,13,5,2,2,1)	1591	3.44	5.68	60.56	0.18	4.40	5.81	17.43	0.00	88.38	11.62
ubikker	4	40 (3,21,9,4,2,1)	2052	0.08	7.02	1.14	0.14	2.71	4.42	45.30	0.00	53.70	46.30
vermeer76	5	12 (1,11)	762	0.03	0.07	42.86	14.29	0.00	28.57	0.00	14.29	100.00	0.00
cyclic5	5	20 (1,18,1)	770	0.00	0.04	0.00	0.00	50.00	25.00	25.00	0.00	100.00	0.00
kats5	5	15 (1,9,3,1)	770	0.00	0.04	0.00	0.00	0.00	0.00	50.00	0.00	50.00	50.00
noon5	5	155 (6,141,14)	32421	0.03	28.78	0.10	2.08	0.59	10.67	81.97	1.46	96.87	3.13
cyc5mod1	5	70 (1,39,19,9,2)	5523	0.12	5.17	2.32	1.35	2.32	4.26	30.75	1.93	42.94	57.06
kats6	6	41 (1,26,8,4,1)	1920	0.05	1.16	4.31	1.72	5.17	6.90	49.14	0.00	67.24	32.76
MK5	6	41 (1,26,8,4,1)	2028	0.07	1.86	3.76	1.08	3.77	6.45	53.23	0.00	68.28	31.72
cyc6	6	76 (1,67,7)	3845	0.39	1.84	21.20	1.63	22.83	25.00	22.83	1.63	95.11	4.89
s7	7	93 (1,91,1)	2820	0.01	0.53	1.89	3.77	33.97	22.64	30.19	0.00	92.45	7.55
exam57	8	63 (1,47,15)	36081	2.96	817.83	0.36	0.01	4.55	1.40	17.16	0.01	23.49	76.51
virasoro	8	73 (1,70,2)	45414	83.68	133.71	62.58	0.48	5.09	13.27	17.49	0.34	99.25	0.75
filter9	9	80 (1,77,2)	5079	92.24	96.43	95.65	0.05	1.48	0.62	2.17	0.01	99.99	0.01

Tabelle F.2: Anzahl der Schritte, maximaler Speicherverbrauch und Berechnungszeiten des Tranalgorithmus.

Bsp.	n	Schritte	Speicher	B.-Zeit (Sek.)		Berechnungszeiten (%)							
				inp.	total	inp.	InFo	sG_ω	Lift	InRd	NäVe	Summe	Rest
qnt1	3	198 (34,164)	41182	0.37	1626.87	0.02	0.02	2.10	17.69	80.05	0.02	99.90	0.10
qnt2	3	42 (15,27)	6542	0.00	10.68	0.00	0.09	0.75	9.36	88.95	0.19	99.34	0.66
tran2	3	45 (10,45)	1787	0.00	0.87	0.00	0.00	2.30	11.49	82.76	2.30	98.85	1.15
ex2	3	42 (10,32)	1315	0.01	0.60	1.67	0.00	8.33	5.00	81.67	0.00	96.67	3.33
ex3	3	46 (12,34)	1836	0.01	0.47	2.13	4.26	6.38	12.77	74.47	0.00	100.00	0.00
exam7	3	43 (6,37)	1896	0.01	1.75	0.57	0.57	1.71	8.57	87.43	0.00	98.86	1.14
exam12	3	27 (7,20)	762	0.00	0.21	0.00	0.00	9.52	0.00	85.71	0.00	95.24	4.76
rose	3	53 (51,2)	1787	0.01	0.96	1.04	1.04	4.17	12.50	77.08	1.04	96.88	3.12
nmann1	3	64 (11,53)	1787	0.00	0.64	0.00	1.56	3.12	3.12	85.94	1.56	95.31	4.69
lblau	3	56 (11,45)	27897	0.83	694.78	0.12	0.01	0.24	6.50	93.06	0.00	99.93	0.07
m52430	3	152 (18,133)	18764	0.36	376.62	0.10	0.03	0.66	9.49	89.54	0.03	99.84	0.16
m51390	3	131 (19,112)	10224	0.08	68.44	0.12	0.12	1.27	10.34	87.59	0.16	99.61	0.39
issac97	4	19 (2,17)	762	0.01	0.21	4.76	0.00	9.52	9.52	71.43	0.00	95.24	4.76
ubikker	4	53 (3,50)	2036	0.08	4.84	1.65	0.00	1.45	7.02	89.46	0.21	99.79	0.21
wang6	4	36 (4,32)	1275	0.02	0.40	5.00	2.50	10.00	7.50	75.00	0.00	100.00	0.00
cassou	4	9 (1,8)	782	0.25	0.26	96.15	0.00	0.00	0.00	3.85	0.00	100.00	0.00
casmod1	4	29 (3,26)	1591	2.07	3.55	58.31	0.00	4.51	6.76	29.86	0.28	99.72	0.28
exam58	4	35 (2,32)	1275	0.01	0.15	6.67	13.33	0.00	13.33	66.67	0.00	100.00	0.00
exam59	4	19 (2,17)	762	0.02	0.22	9.09	0.00	18.18	9.09	63.64	0.00	100.00	0.00
exam46	4	55 (5,50)	2300	0.00	0.26	0.00	7.69	7.69	19.23	42.31	3.85	80.77	19.23
exam14	4	27 (1,26)	762	0.00	0.25	0.00	4.00	8.00	0.00	80.00	4.00	96.00	4.00
noon4	4	55 (5,50)	2300	0.00	0.51	0.00	5.88	5.88	7.84	76.47	1.96	98.04	1.96
noon5	5	161 (6,155)	33959	0.02	24.77	0.08	1.98	0.36	11.06	82.80	1.41	97.70	2.30
cyc5mod1	5	86(1,85)	4494	0.08	2.31	3.46	1.73	0.87	6.06	83.98	0.87	96.97	3.03
kats6	6	44 (1,43)	1920	0.06	1.53	3.92	0.65	3.27	8.50	79.74	0.00	96.08	3.92
MK5	6	44 (1,43)	2028	0.06	1.81	3.31	1.10	3.31	5.52	83.98	2.21	99.45	0.55
cyclic6	6	69 (1,68)	3333	0.37	1.70	21.76	0.59	24.71	25.88	22.35	1.18	96.47	3.53
kats7	7	111 (1,110)	9511	0.64	68.03	0.94	0.21	0.66	7.28	90.27	0.13	99.49	0.51
s7	7	106 (1,105)	3341	0.02	0.55	3.64	3.64	27.27	32.73	21.82	3.64	92.73	7.27
eco7	7	46 (1,45)	3525	0.02	15.13	0.13	0.00	0.53	3.83	95.11	0.00	99.60	0.40
redeco7	7	48 (1,47)	1787	0.01	0.36	2.78	8.33	2.78	13.89	69.44	2.78	100.00	0.00
s8	8	202 (1,201)	14664	0.05	3.93	1.27	4.83	28.50	26.97	27.99	3.82	93.38	6.62
redeco8	8	132 (1,131)	8586	0.06	8.91	0.67	1.01	1.57	7.41	86.08	1.46	98.20	1.80
exam57	8	77 (1,76))	11637	1.84	558.04	0.33	0.01	0.58	5.15	93.85	0.01	99.93	0.07
virasoro	8	64 (1,63)	41346	116.51	181.23	64.29	0.55	3.97	11.95	18.28	0.31	99.35	0.65
dessin1	8	64 (1,45,18)	20254	0.58	437.29	0.13	0.00	0.33	2.77	96.73	0.01	99.97	0.03
filter9	9	78 (4,74)	5035	83.74	87.76	95.42	0.08	1.60	0.70	2.10	0.02	99.91	0.09
omdi	9	10 (1,9)	762	0.19	0.21	90.48	0.00	0.00	0.00	4.76	4.76	100.00	0.00
dessin2	10	52 (3,38,9,2)	8946	1.16	66.59	1.74	0.05	5.84	0.00	80.58	11.68	99.89	0.11
bmarkD1	12	94 (1,93)	12358	26.31	423.39	6.21	0.02	1.30	3.19	89.20	0.01	99.93	0.07
f744	12	29 (1,28)	5067	4.87	6.77	71.94	1.18	5.02	7.98	10.93	1.18	98.23	1.77

Tabelle F.3: Anzahl der Schritte, maximaler Speicherverbrauch und Berechnungszeiten des ersten Alternativealgorithmus.

Bsp.	n	Schritte	Speicher	B.-Zeit (Sek.)		Berechnungszeiten (%)							
				inp.	total	inp.	InFo	sG_ω	Lift	InRd	NäVe	Summe	Rest
qnt1	3	197 (34,163,0)	41082	0.37	1618.69	0.02	0.01	1.99	17.82	80.04	0.02	99.90	0.10
qnt2	3	41 (15,26,0)	5499	0.00	10.46	0.00	0.38	0.86	8.99	88.81	0.19	99.24	0.76
tran2	3	54 (10,44,0)	1287	0.01	0.87	1.15	0.00	2.30	13.79	75.86	0.00	93.10	6.90
ex2	3	41 (10,31,0)	1315	0.01	0.56	1.79	3.57	5.36	16.07	71.43	1.79	100.00	0.00
ex3	3	45 (12,33,0)	1836	0.01	0.46	2.17	0.00	2.17	8.70	82.61	0.00	95.65	4.35
exam7	3	42 (6,36,0)	1896	0.01	1.71	0.58	0.00	2.34	7.02	88.89	0.58	99.42	0.58
exam12	3	26 (7,19,0)	762	0.00	0.21	0.00	0.00	9.52	9.52	80.95	0.00	100.00	0.00
rose	3	51 (51,0)	1787	0.01	0.95	1.05	0.00	4.21	12.63	77.89	2.11	97.89	2.11
nmann1	3	63 (11,52,0)	1787	0.00	0.64	0.00	1.56	3.12	7.81	84.38	0.00	96.88	3.12
lblau	3	55 (11,44,0)	27845	0.81	684.84	0.12	0.01	0.25	6.67	92.88	0.01	99.92	0.08
m52430	3	150 (18, 132)	18804	0.37	451.22	0.08	0.06	0.63	10.01	89.04	0.02	99.84	0.16
m51390	3	130 (19,111)	10228	0.08	69.33	0.12	0.16	1.13	10.00	88.09	0.14	99.62	0.38
issac97	4	18 (2,4,12,0)	762	0.02	0.21	9.52	0.00	0.00	14.29	71.43	0.00	95.24	4.76
ubikker	4	46 (3,9,34,0)	2108	0.08	5.05	1.58	0.00	3.37	11.68	82.97	0.20	99.80	0.20
wang6	4	30 (4,7,19,0)	1275	0.02	0.38	5.26	0.00	18.42	13.16	63.16	0.00	100.00	0.00
casmod1	4	27 (3,7,17,0)	1591	2.07	3.82	54.19	0.00	5.24	10.99	29.06	0.00	99.48	0.52
exam58	4	32 (2,9,21,0)	1275	0.01	0.15	6.67	0.00	0.00	20.00	53.33	13.33	93.33	6.67
exam59	4	18 (2,4,12,0)	762	0.01	0.22	4.55	0.00	9.09	27.27	59.09	0.00	100.00	0.00
exam46	4	41 (5,13,23,0)	1787	0.01	0.31	3.23	3.23	9.68	9.68	54.84	9.68	90.32	9.68
exam14	4	25 (1,5,19,0)	762	0.01	0.34	2.94	2.94	2.94	8.82	76.47	0.00	94.12	5.88
noon4	4	41 (5,13,23,0)	1787	0.01	0.46	2.17	8.70	2.17	4.35	78.26	4.35	100.00	0.00
noon5	5	102 (6,17,29,50,0)	17527	0.03	15.84	0.19	1.45	1.64	19.00	72.98	2.15	97.41	2.59
cyc5mod1	5	80 (1,5,13,61,0)	3982	0.08	2.44	3.28	1.23	4.92	14.34	74.59	0.41	98.77	1.23
kats6	6	42 (1,2,4,9,26,0)	1920	0.08	1.76	4.55	1.14	6.82	12.50	71.02	0.57	96.59	3.41
MK5	6	42 (1,2,4,9,26,0)	2028	0.09	2.75	3.27	0.36	6.91	12.73	75.64	0.00	98.91	1.09
noon6	6	200 (7,17,31,53,92,0)	223832	0.34	724.63	0.05	0.46	2.24	38.14	57.65	0.33	98.88	1.12
cyc6	6	42 (1,2,4,8,27,0)	2308	0.58	2.63	22.05	0.76	44.11	23.57	6.84	0.76	98.10	1.90
exam54	6	42 (1,2,3,4,5,6)	2300	0.39	2.01	19.40	0.50	43.78	22.89	9.95	1.99	98.51	1.49
kats7	7	99 (1,2,4,6,17,67,0)	9735	0.64	96.19	0.67	0.11	8.22	20.02	70.56	0.07	99.66	0.34
s7	7	60 (1,4,5,9,17,24,0)	1795	0.01	0.48	2.08	2.08	39.58	16.67	20.83	6.25	87.50	12.50
eco7	7	45 (1,2,3,7,16,16,0)	1787	0.02	0.54	3.70	5.56	3.70	16.67	59.26	0.00	88.89	11.11
redeco7	7	45 (1,2,5,10,1,26,0)	1787	0.01	0.36	2.78	8.33	5.56	13.89	63.89	2.78	97.22	2.78
s8†(7)	8	75 (1,4,5,8,11,27,19)	4591	0.05	1.95	2.56	4.10	49.23	20.51	15.90	2.56	94.87	5.13
redeco8	8	102 (1,2,5,9,17,1,67,0)	6156	0.08	12.72	0.63	1.18	3.07	12.89	80.50	0.79	99.06	0.94
exam57	8	66 (1,4,1,6,1,10,43,0)	10664	2.94	603.99	0.49	0.01	5.65	12.33	81.45	0.00	99.92	0.08
virasoro	8	39 (1,1,1,2,6,9,19,0)	26003	120.24	178.58	67.33	0.25	5.34	9.26	17.04	0.19	99.42	0.58
dessin1	8	65 (1,4,1,6,10,43,0)	5255	0.92	101.01	0.91	0.03	7.70	12.89	78.21	0.08	99.82	0.18
filter9	9	27 (4,5,5,3,2,2,2,2,4,0)	5055	110.90	113.26	97.92	0.01	1.45	0.29	0.30	0.00	99.96	0.04
dessin2	10	65 (3,2,1,1,5,6,10,37,0)	4358	1.81	24.79	7.30	0.04	3.15	8.96	80.03	0.16	99.64	0.36
bmarkD1†(11)	12	46 (1,1,1,2,3,4,4,7,4,19,0)	14584	35.49	642.64	5.52	0.01	64.35	13.17	16.90	0.01	99.95	0.05
f744 †(11)	12	21 (1,1,1,2,2,2,1,3,3,4,1,0)	8486	4.58	10.62	43.13	0.56	49.92	2.64	2.82	0.47	97.55	2.45
chemkin†(12)	13	32 (1,1,1,1,2,3,3,1,1,6,12,0)	24506	9.74	456.83	2.13	0.02	70.09	16.89	10.82	0.01	99.95	0.05

Tabelle F.4: Anzahl der Schritte, maximaler Speicherverbrauch und Berechnungszeiten des zweiten Alternativealgorithmus (†Im letzten Schritt muss der Buchbergeralgorithmus angewendet werden).

Literaturverzeichnis

[1] Adams, W. and Loustaunau, P. (1994), *An Introduction to Gröbner Bases*, Graduate Studies in Mathematics **3**, AMS, Providence.

[2] Amrhein, B., Gloor, O. and Küchlin, W. (1996a), *How Fast Does the Walk Run?*, in Proceedings of the 5. Rhine Workshop on Computer Algebra (RWCA'96), Saint-Louis, France, April 1-3, 1996.

[3] Amrhein, B., Gloor, O. and Küchlin, W. (1996b), *Walking Faster*, in Proceedings of DISCO'96, Karlsruhe, Germany, Lecture Notes in Computer Science **1128**, Springer-Verlag, New York, 150-161.

[4] Amrhein, B., Gloor, O. and Küchlin, W. (1997), *On the Walk*, Preprint (to appear in TCS (Special Issue for RWCA'96)).

[5] Amrhein, B. and Gloor, O. (1998), *The Fractal Walk*, in *Gröbner Bases and Applications* (Buchberger, B. and Winkler, F. eds.), Cambridge University Press, Cambridge, 305-322.

[6] Bayer, D. and Stillman, M. (1987), *A Theorem on Refining Division Orders by the Reverse Lexicographic Order*, Duke J. Math. **55**, 321-328.

[7] Becker, T. and Weispfenning, V. (1998), *Gröbner Bases: A Computational Approach to Commutative Algebra*, Springer-Verlag, New York.

[8] Boege, W., Gebauer, R. and Kredel, H. (1986), *Some Examples for Solving Systems of Algebraic Equations by Calculating Gröbner Bases*, Journal Symbolic Computation **1**, 83-98.

[9] Buchberger, B. (1979), *A Criterion for Detecting Unnecessary Reduction of Gröbner-Bases*, in Proceedings EUROSAM'79, Lecture Notes in Computer Science **72**, Springer-Verlag, New York, 3-21.

[10] Buchberger, B. (1985), *Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory*, in *Multidimensional Systems Theory* (N.K. Bose ed.), D. Reidel Publishing Company, Dordrecht, 184-232.

[11] Collart, S., Kalkbrener, M. and Mall, D. (1997), *Converting Bases with the Gröbner Walk*, Journal Symbolic Computation **24**, 465-470.

[12] Cox, D., Little, J. and O'Shea, D. (1997), *Ideals, Varieties and Algorithms, second ed.*, Springer-Verlag, New York.

[13] Cox, D., Little, J. and O'Shea, D. (1998), *Using Algebraic Geometry*, Springer-Verlag, New York.

[14] Czapor, S. R. (1989), *Solving Algebraic Equations via Buchberger's Algorithm*, in Proceedings EUROCAL'87, Lecture Notes in Computer Science **378**, Springer-Verlag, New York, 260-269.

[15] Czapor, S. R. (1991), *A Heuristic Selection Strategy for Lexicographic Gröbner Bases?*, in Proceedings of ISSAC'91, ACM Press, New York, 39-48.

[16] Eisenbud, D. (1995), *Commutative Algebra with a View Toward Algebraic Geometry*, Springer-Verlag, New York.

[17] Ewald, G. (1996), *Combinatorial Convexity and Algebraic Geometry*, Springer-Verlag, New York.

[18] Faugére, J.C., Gianni, P., Lazard, D. and Mora, T. (1993), *Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering*, Journal Symbolic computation **16**, 329-344.

[19] Gebauer, R. and Möller, H. M. (1988), *On an Installation of Buchberger's Algorithm*, Journal Symbolic computation **6**, 275-286.

[20] Geddes, K. O., Czapor S. R. and Labahn, G. (1992), *Algorithms for Computer Algebra*, Kluwer Academic Publishers, Boston, Dordrecht, London.

[21] Giovini, A., Mora, T., Niesi, G., Robbiano, L. and Traverso, C. (1991), *„One sugar cube, please" OR Selection strategies in the Buchberger algorithm*, in Proceedings of ISSAC'91, ACM Press, New York, 49-54.

[22] Goldman, A.J. and Tucker, A.W. (1956), *Polyhedral Convec Cones*, in *Linear Inequalities and Related Systems* (Kuhn and Tucker eds.), Annals of Mathematics Studies number **38**, Princeton University Press, Princeton, New Jersey.

[23] Greuel, G.-M., Pfister, G. and Schoenemman, H. (1997), *Singular Reference Manual*, in Reports on Computer Algebra number **12**, Centre for Computer Algebra, University of Kaiserslautern, http://www.mathematik.uni-kl.de/∼zca/Singular.

[24] Greuel, G.-M. and Pfister, G. (2002), *A* SINGULAR *Introduction to Commutative Algebra*, Springer-Verlag, Berlin, Heidelberg.

[25] Gröbner, W. (1968), *Algebraische Geometrie I*, Bibligraphisches Institut Mannheim.

[26] Gröbner, W. (1970), *Algebraische Geometrie II*, Bibligraphisches Institut Mannheim.

[27] Kalkbrener, M. (1989), *Solving Systems of Algebraic Equations by Using Gröbner Bases*, in Proceedings EUROCAL'87, Lecture Notes in Computer Science **378**, Springer-Verlag, New York, 282-292.

[28] Kalkbrener, M. (1996), *Implicitization by Gröbner Basis Conversion*, Journal Bull. Math. **2**, 197-204.

[29] Kernighan, B.W. and Ritchie, D.M. (1988), *The C Programming Language*, Prentice Hall, London.

[30] Mora, T. and Robbiano, L. (1988), *The Gröbner Fan of an Ideal*, Journal Symbolic computation **6**, 183-208.

[31] Oualline, S. (1997), *Practical C Programming, third ed.*, O'Reilly, Cambridge, Köln.

[32] Robbiano, L. (1985), *Term Orderings on the Polynomial Ring*, in Proceedings of EUROCAL'85, Lecture Notes in Computer Science **204**, Springer-Verlag, 513-156.

[33] Sturmfels, B. (1996), *Gröbner Bases and Convex Polytopes*, University Lecture Series **8**, AMS, Providence, Rhode Island.

[34] Tran, Quoc-Nam (1998), *Parallel Computation and Gröbner Bases: an Application for Converting Bases with the Gröbner Walk*, in *Gröbner Bases and Applications* (Buchberger, B. and Winkler, F. eds.), Cambridge University Press, Cambridge, 519-531.

[35] Tran, Quoc-Nam (2000), *A Fast Algorithm for Gröbner Basis Conversion and its Applications*, Journal Symbolic Computation **30**, 451-467.

[36] Traverso, C. (1996), *Hilbert functions and the Buchberger algorithm*, Journal Symbolic Computation **22**, 355-376.

[37] Trinks, W. (1978), *Über B. Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen*, Journal of Number Theory **10**, 475-488.

[38] Weispfenning, V. (1987), *Admissible Orders and Linear Forms*, ACM SIGSAM Bulletin **21**, 16-18.

[39] Wichmann, T. (1997), *Der FGLM-Algorithmus: Verallgemeinert und implementiert in* SINGULAR, Dimplomarbeit, Kaiserslautern.

[40] http://www.calfor.lip6.fr/~jcf/Benchs/index.html

[41] http://www.symbolicdata.org/SD_HTML/index.html

[42] http://www.posso.lip6.fr/~jcf/

Index